Aula 00 - Prof. Diego Carvalho e Fernando Pedrosa

CORENTO (Assistente Administrativo - Informática) Engenharia de Software - 2024 (Pós-Edital)

Autor:

Diego Carvalho, Equipe Informática 2 (Diego Carvalho)

20 de Novembro de 2024

Índice

APRESENTAÇÃO DO PROFESSOR

PROF. DIEGO CARVALHO

FORMADO EM CIÊNCIA DA COMPUTAÇÃO PELA UNIVERSIDADE DE BRASÍLIA (UNB), PÓS-GRADUADO EM GESTÃO DE TECNOLOGIA DA INFORMAÇÃO NA ADMINISTRAÇÃO PÚBLICA E, ATUALMENTE, AUDITOR FEDERAL DE FINANÇAS E CONTROLE DA SECRETARIA DO TESOURO NACIONAL.

ESTRATÉGIA CONCURSOS

PROFESSOR DIEGO CARVALHO - WWW.INSTAGRAM.COM/PROFESSORDIEGOCARVALHO

Sobre o curso: galera, todos os tópicos da aula possuem Faixas de Incidência, que indicam se o assunto cai muito ou pouco em prova. Diego, se cai pouco para que colocar em aula? Cair pouco não significa que não cairá justamente na sua prova! A ideia aqui é: se você está com pouco tempo e precisa ver somente aquilo que cai mais, você pode filtrar pelas incidências média, alta e altíssima; se você tem tempo sobrando e quer ver tudo, vejam também as incidências baixas e baixíssimas. *Fechado?*

INCIDÊNCIA EM PROVA: BAIXÍSSIMA

INCIDÊNCIA EM PROVA: BAIXA

INCIDÊNCIA EM PROVA: MÉDIA

INCIDÊNCIA EM PROVA: ALTA

INCIDÊNCIA EM PROVA: ALTÍSSIMA

Além disso, essas faixas não são por banca – é baseado tanto na quantidade de vezes que caiu em prova independentemente da banca quanto nas minhas próprias avaliações sobre cada assunto.

#ATENÇÃO

Avisos Importantes

O curso abrange todos os níveis de conhecimento...

Esse curso foi desenvolvido para ser acessível a **alunos com diversos níveis de conhecimento diferentes.** Temos alunos mais avançados que têm conhecimento prévio ou têm facilidade com o assunto. Por outro lado, temos alunos iniciantes, que nunca tiveram contato com a matéria ou até mesmo que têm trauma dessa disciplina. A ideia aqui é tentar atingir ambos os públicos - iniciantes e avançados - da melhor maneira possível..

Por que estou enfatizando isso?

O **material completo** é composto de muitas histórias pessoais, exemplos, metáforas, piadas, memes, questões, desafios, esquemas, diagramas, imagens, entre outros. Já o **material simplificado** possui exatamente o mesmo núcleo do material completo, mas ele é menor e mais objetivo. *Professor, eu devo estudar por qual material?* Se você quiser se aprofundar nos assuntos ou tem dificuldade com a matéria, necessitando de um material mais passo-a-passo, utilize o material completo. Se você não quer se aprofundar nos assuntos ou tem facilidade com a matéria, necessitando de um material mais direto ao ponto, utilize o material simplificado.

Por fim...

O curso contém diversas questões espalhadas em meio à teoria. Essas questões possuem um comentário mais simplificado porque **têm o único objetivo de apresentar ao aluno como bancas de concurso cobram o assunto previamente administrado**. A imensa maioria das questões para que o aluno avalie seus conhecimentos sobre a matéria estão dispostas ao final da aula na lista de exercícios e **possuem comentários bem mais abrangentes**.

ESTRATÉGIA FLASHCARDS

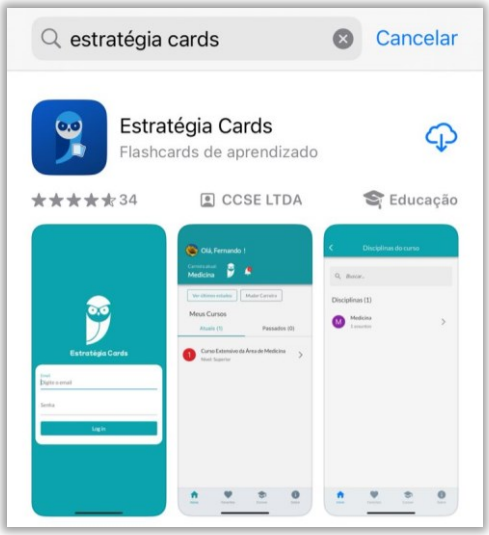
📚 Você tem dificuldade de estudar, memorizar e revisar os conteúdos que estuda em nossas aulas? Então nós temos a ferramenta perfeita para você!

Apresentamos o **Estratégia Cards**: app de flashcards que vai revolucionar sua forma de **estudar** e **revisar** conteúdos de provas de concurso público. Com nossa tecnologia inovadora e interface amigável, você dominará os tópicos mais complexos de maneira eficiente e divertida.

✨ Recursos do Estratégia Cards:

Curadoria de Flashcards	Flashcards criados e revisados por professores especializados em cada área, com qualidade e voltados para concursos públicos.
Flashcards Personalizados	Crie seus próprios flashcards, cobrindo os principais tópicos e matérias dos concursos públicos.
Repetição Espaçada	Técnica de aprendizagem que envolve revisar informações em intervalos crescentes para melhorar a retenção de longo prazo e combater o esquecimento.
Estatísticas Personalizadas	Visualize graficamente o percentual de acertos, erros ou dúvidas dos decks estudados.
Modo Offline	Estude em qualquer lugar, mesmo sem conexão à internet, fazendo o download dos decks.
Estudo por Áudio	*Está dirigindo ou fazendo esteira e quer continuar estudando?* Basta utilizar a opção "Escutar".
Decks Favoritos	Você pode escolher decks específicos como favoritos e visualizá-los em uma aba separada do app.
Opções de Estudo	Você poderá estudar todos os cards de um deck; ou apenas os que você errou; ou apenas os que você não estudou ainda; entre outras opções.

E como eu consigo baixar?



É muito fácil! Basta pesquisar por "Estratégia Cards" na loja oficial do seu smartphone.

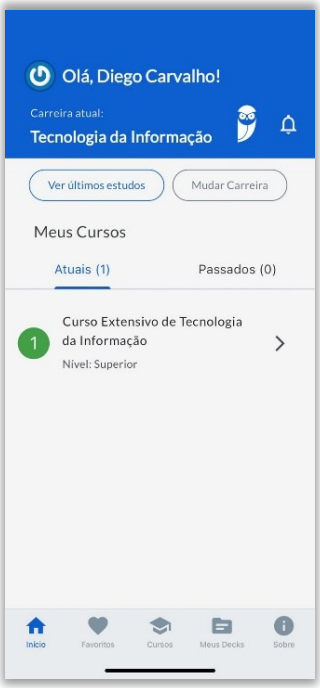
Se você tiver um Android, basta acessar a **Google Play**;

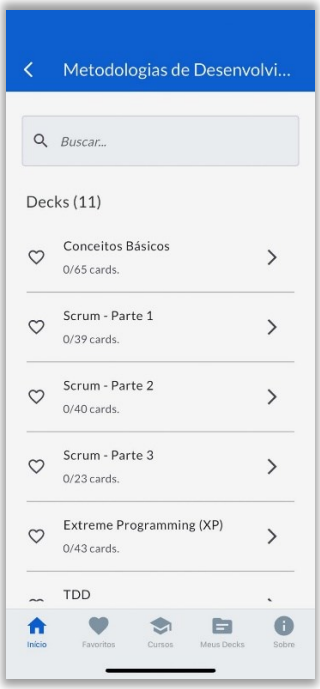
Se for tiver um iPhone, basta acessar a **App Store (iOS)**.

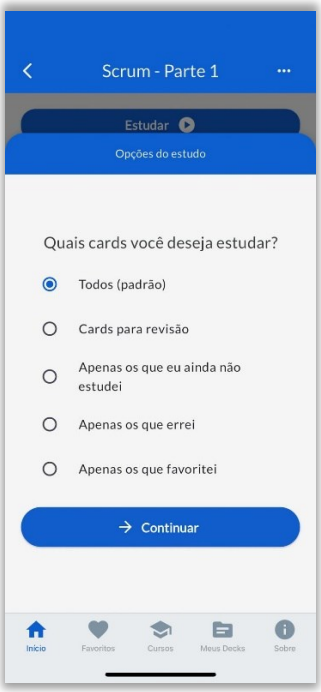
E para acessar?

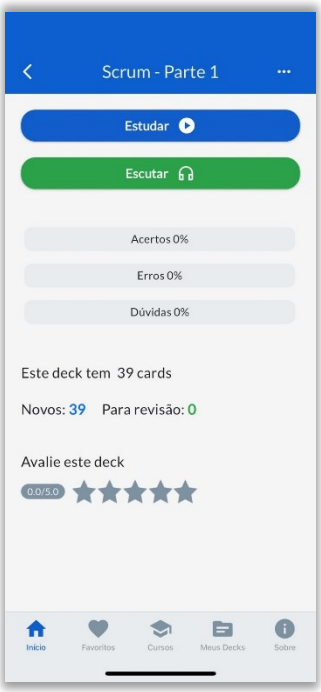
Para acessar, basta ter uma conta no Estratégia Concursos. Em seguida, utilize suas credenciais de login e senha para acessar o aplicativo. Por fim, acessa a carreira de Tecnologia da Informação.

Como utilizar o app:









PARADIGMA ORIENTADO A OBJETOS

Conceitos Básicos

INCIDÊNCIA EM PROVA: MÉDIA

Um dos maiores problemas da programação estruturada é que, muitas vezes, partes do código que servem apenas para tratar os dados se misturam com partes do código que tratam da lógica do algoritmo. Essa prática não é saudável, na medida em que diminui a reusabilidade e dificulta leitura, depuração e manutenção. A modularização da programação estruturada foi um grande avanço na busca pela reusabilidade de código, mas nem se compara ao que trouxe a programação orientada a objetos. Esse novo paradigma reflete bem mais fielmente os problemas atuais. É um paradigma que se baseia na abstração de coisas ou objetos do mundo real em um sistema de forma potencialmente reusável! A reusabilidade de classes melhora a agilidade e permite que programas sejam escritos mais rapidamente...

Galera, vocês trabalham com isso e sabem que a demanda só aumenta, logo existe uma busca por maneiras de se desenvolver sistemas de forma mais rápida! *Para quê reinventar a roda?* Nada disso!

Devemos aproveitar tudo que puder ser aproveitado. **Além disso, é importante citar a escalabilidade.** Softwares construídos seguindo os preceitos da orientação a objetos são mais escaláveis, isto é, podem crescer facilmente sem aumentar demasiadamente sua complexidade ou comprometer seu desempenho. É possível construir tanto um software da Padaria do João quanto o sistema que controla o Acelerador de Partículas da NASA.

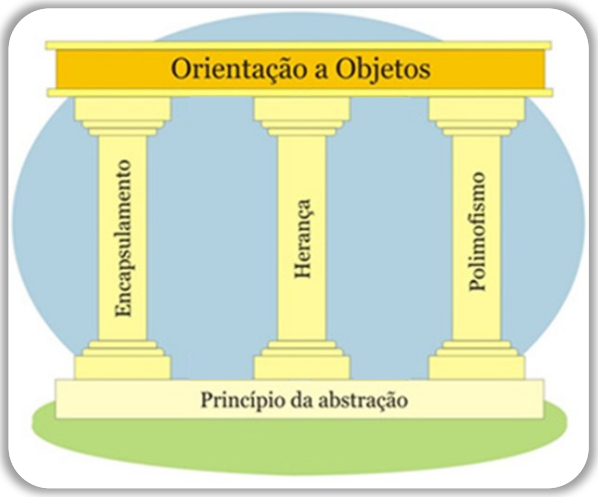
Outra grande vantagem do paradigma é o seu caráter unificador, isto é, tratar todas as etapas do desenvolvimento de sistemas e ambientes sob uma única abordagem. Nesse sentido, podemos ter análise, projeto, modelagem, implementação, banco de dados, e ambientes orientados a objetos. Isso elimina as diferenças de paradigmas utilizadas em cada um desses contextos.

O Paradigma Orientado a Objetos (POO) visualiza um sistema de software como uma coleção de agentes interconectados chamados objetos. Cada objeto é responsável por realizar tarefas específicas e é por meio da interação entre objetos que uma tarefa computacional é realizada. *Galera, vocês conseguem perceber que a sociedade utiliza o conceito de objetos cotidianamente para resolver seus problemas?*

Pois é, já é algo natural! **Ele auxilia a modelagem de sistemas, reduzindo a diferença semântica entre a realidade sendo modelada e os modelos construídos.** Um sistema orientado a objetos consiste em objetos em colaboração com o objetivo de realizar as funcionalidades desse sistema. Cada objeto é responsável por tarefas específicas e a cooperação entre eles é importante para o desenvolvimento do sistema.

O POO tem a finalidade maior é a de facilitar a vida dos programadores para que eles consigam desenvolver softwares que satisfaçam os clientes, transformando coisas do dia a dia em objetos e permitindo que sejam empregados os seus recursos de forma eficaz. O POO tem evoluído em questões voltadas para segurança e reaproveitamento de código – requisitos estes considerados importantes no desenvolvimento de qualquer aplicação moderna. Vejamos as principais vantagens:

VANTAGENS DO PARADIGMA ORIENTADO A OBJETOS
Produção de software natural. Os programas naturais são mais inteligíveis. Em vez de programar em termos de regiões de memória, o profissional pode programar usando a terminologia de seu problema em particular.
Programas orientados a objetos, bem projetados e cuidadosamente escritos são confiáveis.
Pode-se reutilizar prontamente classes orientadas a objetos bem-feitas. Assim como os módulos, os objetos podem ser reutilizados em muitos programas diferentes.
Um código orientado a objetos bem projetado é manutenível. Para corrigir um erro, o programador simplesmente corrige o problema em um lugar. Como uma mudança na implementação é transparente, todos os outros objetos se beneficiarão automaticamente do aprimoramento.
O software não é estático. Ele deve crescer e mudar com o passar do tempo, para permanecer útil. A programação orientada a objetos apresenta ao programador vários recursos para estender código. Esses recursos incluem herança, polimorfismo, sobreposição e uma variedade de padrões de projeto.
O ciclo de vida do projeto de software moderno é frequentemente medido em semanas. A programação orientada a objetos ajuda nesses rápidos ciclos de desenvolvimento. Ela diminui o tempo do ciclo de desenvolvimento, fornecendo software confiável, reutilizável e facilmente extensível.



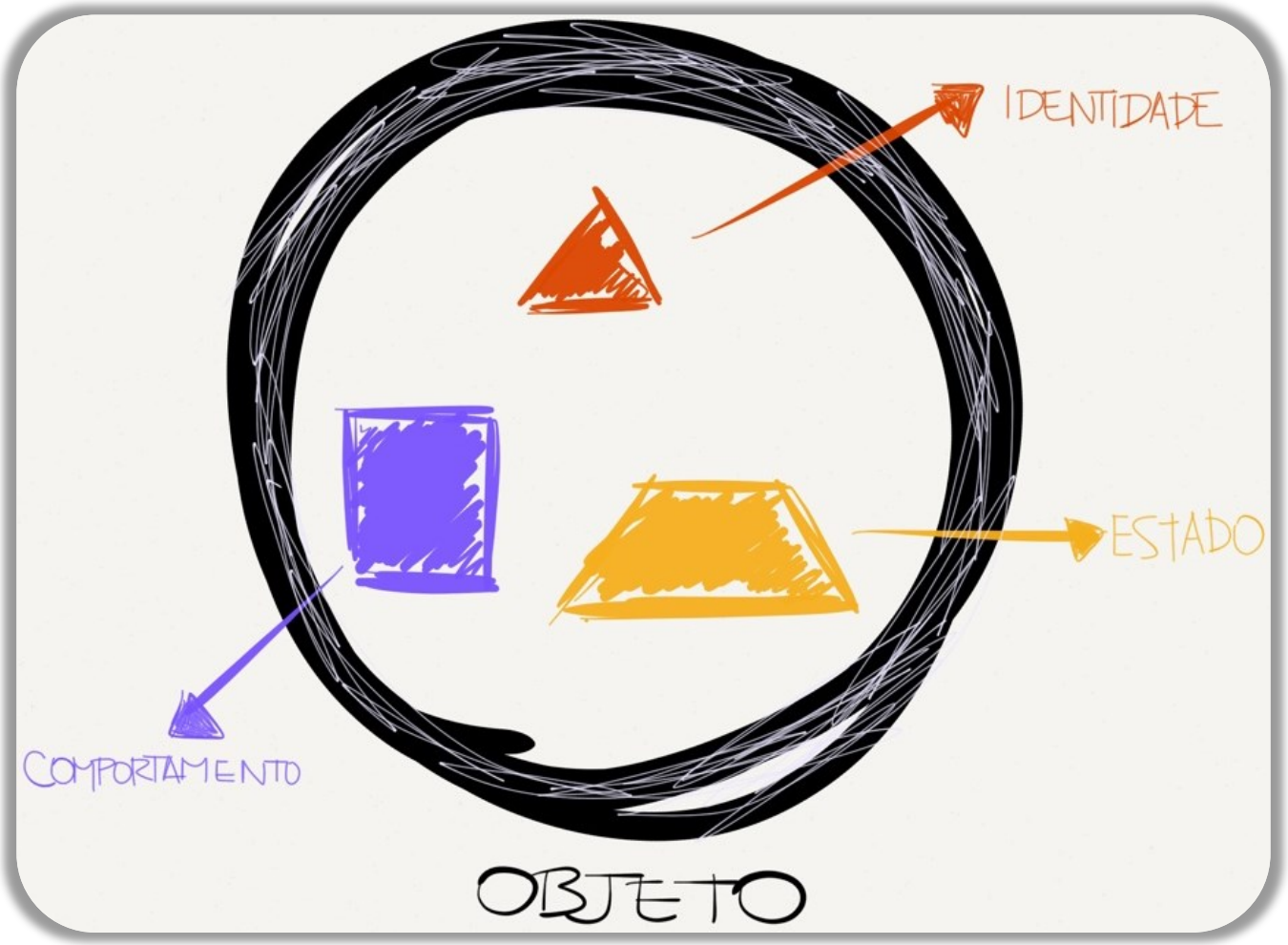
O Paradigma Orientado a Objetos (POO) possui alguns princípios básicos ou pilares fundamentais, como é possível ver na imagem ao lado. Os princípios são: Encapsulamento, Herança e Polimorfismo. *E a abstração, professor?* Galera, alguns autores afirmam que esses princípios são todos simplesmente a aplicação de um único conceito: o princípio da abstração. Vejam pela imagem que a abstração é a base que sustenta todos os outros pilares desse paradigma. Nós veremos cada um desses princípios em detalhes nos próximos tópicos porque eles despencam em prova. *Fechado?*

Classes e Objetos

INCIDÊNCIA EM PROVA: ALTÍSSIMA

Objetos são coisas (Carro, Foto, Bola, etc) e classes são um agrupamento de coisas. A classe é a descrição dos atributos e serviços comuns a um grupo de objetos (reais ou abstratos), logo podemos dizer que é um modelo a partir do qual objetos são construídos. Além disso, objetos são instâncias de classes. *O que é um carro?* Posso abstrair um carro como um objeto que tem motor, volante, porta, rodas, entre outros.

Ora, existem carros que têm dois motores (um elétrico e um à gasolina), outros têm somente duas portas. *Como nós sabemos que ambos são carros?* Porque, independentemente de pequenas diferenças entre as instâncias, nós conseguimos entender que se trata de carros. Para fins de modelagem de um sistema, somente um subconjunto de características é relevante, logo ignoramos o restante.



São componentes de um objeto: identidade, estado (propriedades) e comportamento (operações). A identidade é responsável por distinguir um objeto dos outros, isto é, eles são únicos, mesmo que sejam instâncias de uma mesma classe e que tenham os mesmos valores de variáveis.

O estado reflete os valores correntes dos atributos do objeto em um determinado momento. *Entenderam esse conceito?*

Já o comportamento se refere a como os objetos reagem em relação à mudança de estado e troca de mensagens, isto é, trata-se de um conjunto de atividades externamente observáveis do objeto. *Dito isso, vamos resumir o que vimos até agora nesse tópico?* **Identidade é o que torna o objeto único; Estado se refere aos seus atributos; e Comportamento se refere aos seus métodos e procedimentos.** *Bacana?*

Um objeto é capaz de armazenar estados por meio de seus atributos e reagir a mensagens enviadas a ele, assim como se relacionar com outros objetos e enviar mensagens. Já a classe é como um projeto, formato ou descrição geral de um objeto! São abstrações do domínio do problema, não são diretamente suportadas em todas as linguagens, mas são necessárias em linguagens orientada a objeto.

Atributos

INCIDÊNCIA EM PROVA: ALTÍSSIMA

Consiste em uma informação de estado para o qual cada objeto de uma classe tem seu próprio valor. Há dois tipos: atributos de objetos e de classes. O primeiro descreve valores mantidos em um objeto. Diferentes objetos de uma mesma classe não compartilham os atributos de objetos, isto é, cada um possui sua própria cópia do atributo. O segundo é aquele cujo valor todos os seus objetos devem compartilhar.

As mensagens enviadas a um objeto (isto é, a chamada de um método) podem mudar o valor de um ou mais atributos, alterando o estado de um objeto. Um atributo é um dado para o qual cada objeto tem seu próprio valor. **Atributos são, basicamente, a estrutura de dados que vai representar a classe.** Galera, não tem muito o que falar sobre esse tema não, é isso mesmo! Vamos ver os escopos de atributos:

TIPO DE ATRIBUTO	DESCRIÇÃO
ATRIBUTO DE CLASSE	Similar a uma variável global, trata-se de uma variável cujo valor é comum a todos os objetos membros de uma classe. Mudar o valor de uma variável de classe em um objeto membro automaticamente muda o valor para todos os objetos membros.
ATRIBUTO DE INSTÂNCIA	Trata-se de uma variável cujo valor é específico ao objeto e, não, à classe. Em geral, possui um valor diferente para cada instância. As linguagens de programação possuem palavras para definir o escopo da variável (Ex: em Java, por padrão, é de instância; para ser de classe, deve vir precedida de *static*).

Métodos

INCIDÊNCIA EM PROVA: ALTÍSSIMA

Similares a procedimentos e funções, os métodos consistem em descrições das operações que um objeto executa quando recebe uma mensagem. Existe, portanto, uma correspondência um-para-um entre mensagens e métodos que são executados quando a mensagem é recebida através de um objeto. A mesma mensagem pode resultar em métodos diferentes quanto enviada para objetos diferentes.

Existe um método que nós devemos conhecer bem: método construtor! Os métodos construtores são métodos especiais, que são chamados automaticamente quando instâncias são criadas. Seu objetivo é garantir que o objeto será instanciado de forma correta. Ele tem exatamente o mesmo nome da classe em que está inserido, não possui tipo de retorno e não é obrigatório declará-lo. *Bacana?*

Por meio da criação de construtores, podemos garantir que o código que eles contêm será executado antes de qualquer outro código de outros métodos. Eles geralmente são usados para preparar um objeto, inicializando as variáveis do objeto. Pode existir mais de um método construtor em uma classe através da sobrecarga de construtores. Em algumas linguagens, são acionados por meio do operador *New*.

TIPO DE MÉTODO	DESCRIÇÃO
MÉTODO DE CLASSE	Similar a um método global, trata-se de um método que realiza operações genéricas, isto é, não relativas a uma instância particular. Linguagens de programação possuem palavras para definir o escopo do método (Ex: em Java, deve vir precedida de *static*).
MÉTODO DE INSTÂNCIA	Similar a um método local, trata-se de um método que realiza operações específicas para um objeto e, não, à classe, isto é, são relativas a uma instância particular. Por padrão, todos os métodos de uma determinada classe são considerados métodos de instância.

Cabe salientar que um método nada mais é que uma definição, pois a ação em si somente ocorre quando o objeto é invocado através de um método – por meio de uma mensagem! **Para quem lembra de programação estruturada, métodos são similares a procedimentos ou funções.** Eles definem o comportamento a ser exibido pelas instâncias da classe associada em tempo de execução.

Em tempo de execução, eles possuem acesso aos dados armazenados em um objeto que estão associados e são, desta forma, capazes de controlar o estado da instância. A associação entre classe e método é chamada de ligação (*Binding*). Um método associado a uma classe é dito ligado (*Bound*) à classe. Existem dois tipos de ligações: Early Binding (Ligação Prematura) e Late Binding (Ligação Tardia). Vejamos...

TIPO DE LIGAÇÃO	DESCRIÇÃO
EARLY BINDING	Também conhecida como Ligação Estática, ocorre quando o método a ser invocado é definido em tempo de compilação.
LATE BINDING	Também conhecida como Ligação Dinâmica, ocorre quando o método a ser invocado é definido em tempo de execução.

Aqueles que já conhecem um pouco sobre orientação a objetos, respondam-me: *Qual desses tipos de ligação é o mais comum quando utilizamos polimorfismo?* **Late Binding**!

Mensagens

INCIDÊNCIA EM PROVA: ALTÍSSIMA

Galera, qual é a utilidade de um objeto isolado? Nesse contexto, ele é geralmente muito pouco útil! Por meio da interação entre objetos é que se torna possível obter uma grande funcionalidade ou um comportamento mais complexo. **Logo, assim como no mundo real, objetos estão sempre interagindo uns com os outros.** Bem... essa interação ocorre por meio de troca de mensagens. *Como assim, professor?*

É bom sempre pensar em nosso dia a dia! Imaginemos uma pessoa dirigindo um carro e pensemos em como modelar essa ação para o mundo orientado a objetos! Podemos imaginar um Objeto Pessoa enviando uma mensagem para o Objeto Carro, dizendo-lhe para acelerar, frear, trocar marcha, virar, ligar, desligar, etc. Um objeto manda outro objeto realizar alguma operação enviando-o uma mensagem.

Pois bem, algumas vezes o objeto receptor necessita de algumas informações para realizar o que lhe foi requisitado! Por exemplo: Objeto Carro precisa saber quanto é para acelerar, que horas frear, para qual marcha trocar, etc e o Objeto Pessoa precisa informá-lo. *Como ele pode fazer isso?* **Ele envia a mensagem acompanhada de um conjunto de parâmetros ou argumentos que podem afetar as operações**.

Uma mensagem é composta por três componentes: objeto, a quem a mensagem é endereçada; nome do método ou serviço que se deseja executar; e parâmetros necessários ao método (se existirem). Logo, podemos dizer que mensagens são requisições enviadas de um objeto para outro com o intuito de receber algo em retorno por meio da execução de uma operação. **A natureza das operações realizadas para alcançar o resultado requerido é determinada pelo objeto receptor**.

Em suma, trata-se de um ciclo completo onde uma mensagem é enviada a um objeto, operações são executadas dentro desse objeto e uma mensagem contendo o resultado da operação é enviada

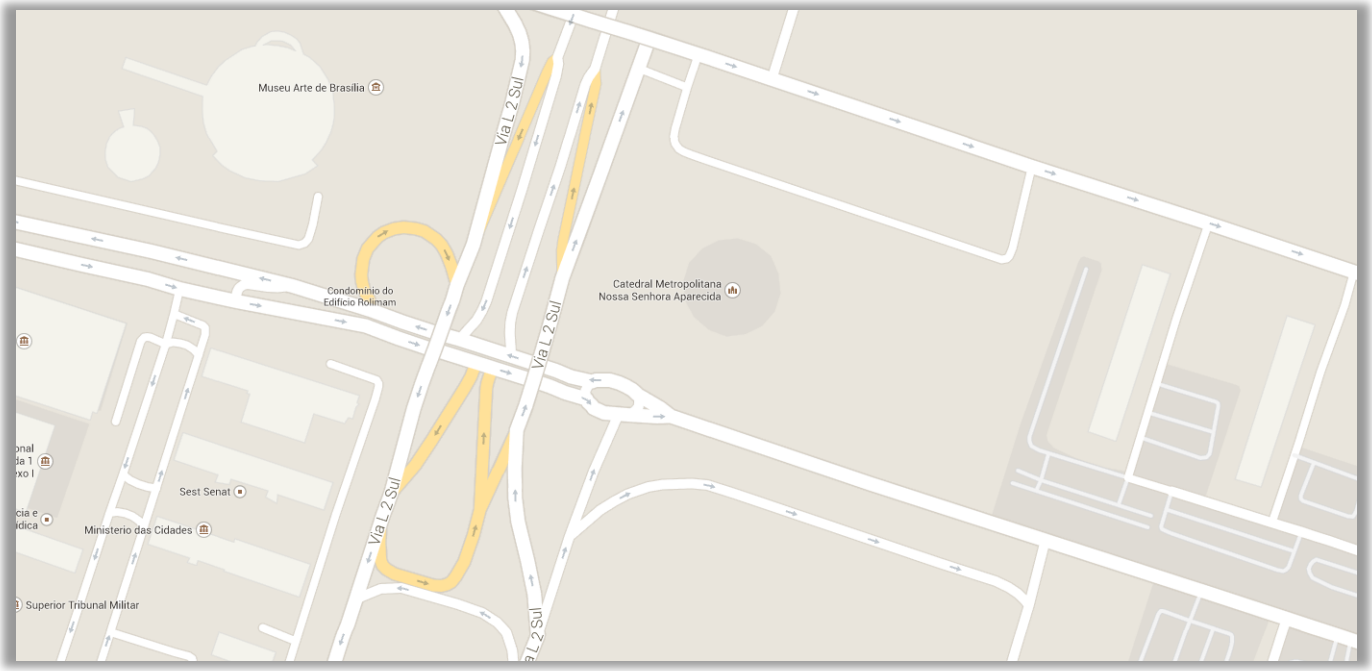
ao objeto solicitante. **Em nosso paradigma, objetos vivem dando ordens para outros objetos executarem métodos (ou realizarem serviços).** O bacana disso tudo é que objetos em processos distintos, máquinas distintas ou redes distintas podem comunicar-se através de mensagens.

E daí, professor? E daí que isso aumenta a coesão e diminui o acoplamento, melhorando substancialmente a reusabilidade! Algumas vezes, sistemas de software possuem tratadores (*handlers*) de mensagens! **Eles são responsáveis por processar mensagens de mais de um transmissor.** Em outras palavras, eles recebem mensagens de diversos objetos e encaminham para seus respectivos donos. Relaxem, nós veremos isso mais à frente com o conceito de polimorfismo!

Abstração

INCIDÊNCIA EM PROVA: ALTÍSSIMA

De modo simples e direto: abstração é a subtração de detalhes, isto é, quanto mais abstrato, há menos detalhes; e quanto menos abstrato, há mais detalhes. Observem as imagens abaixo: trata-se de uma visão aérea da Catedral de Brasília! **Na primeira imagem, há muitos detalhes; a segunda imagem é uma abstração em que se subtraiu diversos detalhes que não são relevantes para o domínio do problema.** Vejamos...



A abstração é um processo mental pelo qual nós seres humanos nos atemos aos aspectos mais importantes/relevantes de algo, ao mesmo tempo que ignoramos aspectos menos importantes. **Esse processo mental nos permite gerenciar a complexidade de um objeto ao mesmo tempo que concentramos nossa atenção em suas características essenciais.** Note que a abstração depende da perspectiva (contexto) sobre a qual ela é analisada.

Temos que falar também sobre classes abstratas e concretas. **Uma classe abstrata é desenvolvida para representar entidades e conceitos abstratos.** Ela é sempre uma superclasse que não possui instâncias. Ela define um modelo/template para uma funcionalidade e fornece uma implementação incompleta (isto é, a parte genérica dessa funcionalidade) que é compartilhada por um grupo de classes derivadas.

Cada uma das classes derivadas completa a funcionalidade da classe abstrata adicionando comportamentos específicos. Por outro lado, classes concretas implementam todos os seus métodos e permitem a criação de instâncias. Em suma: classes concretas são aquelas que podem ser instanciadas diretamente e classes abstratas, não. Há também métodos abstratos: aqueles para os quais não é definida uma forma de implementação específica[1].

[1] É possível haver uma classe abstrata contendo somente métodos concretos. No entanto, se ela tiver um único método abstrato, que seja, deverá ser declarada como abstrata!

Interface

INCIDÊNCIA EM PROVA: ALTÍSSIMA

Sabemos que classes abstratas são aquelas que são desenvolvidas para representar entidades e conceitos abstratos. Elas geralmente contêm ao menos um método abstrato (sem corpo) e não se pode criar uma instância dela. As classes abstratas são usadas para serem herdadas e funcionam como uma superclasse. Nós podemos dizer, então, que se trata de um contrato para que alguma subclasse concretize seus métodos.

No entanto, agora vamos falar de um novo conceito: interfaces. Esse é um tópico que gera muita confusão porque se trata realmente de um conceito parecido com as classes abstratas. Para entender a diferença entre uma interface e uma classe abstrata, devemos nos ater a uma parte importante da definição apresentada no parágrafo anterior: ***"(...) ela deve conter pelo menos um método abstrato (sem corpo)"***. Em uma classe qualquer, existem três possibilidades:

1. Se a classe possui pelo menos um método abstrato, será **obrigatoriamente abstrata**;
2. Se a classe possui todos os métodos abstratos, será **obrigatoriamente abstrata**;
3. Se a classe possui todos os métodos concretos, poderá ser **concreta ou abstrata**.

No primeiro caso, é fácil verificar que a classe é abstrata se ela tem pelo menos um método abstrato; no segundo caso, também é fácil notar que – se todos os métodos são abstratos – ela também deverá ser abstrata; já no terceiro caso, mesmo quando todos os métodos são concretos, ainda assim eu posso declará-la como abstrata – **isso é uma decisão do designer da classe**. Ele será responsável por defini-la da maneira que achar mais adequada.

Agora voltemos um pouco para o segundo caso! **Se todos os métodos são abstratos, a classe será abstrata!** *Professor, uma interface é uma classe?* Não, uma interface é uma entidade em que todos os métodos são obrigatoriamente abstratos. *Opa, perceberam a semelhança?* Existem classes abstratas que contêm todos os métodos abstratos exatamente como uma interface (Caso 2). Vejamos, portanto, quais são as diferenças...

CARACTERÍSTICAS	INTERFACES	CLASSE ABSTRATA
HERANÇA MÚLTIPLA	Suporta Herança Múltipla. Pode implementar diversas interfaces.	Não suporta Herança Múltipla. Não pode estender várias classes abstratas.
IMPLEMENTAÇÃO	Não pode conter qualquer método concreto, apenas abstratos.	Pode conter métodos concretos ou abstratos.
CONSTANTES	Suporta somente constantes estáticas.	Suporta constantes estáticas e de instância.
ENCAPSULAMENTO	Métodos e membros devem sempre ser públicos por padrão.	Métodos e membros podem ter qualquer visibilidade.
MEMBROS DE DADOS	Não contêm atributos de instância, apenas constantes.	Pode conter atributos de instância.
CONSTRUTORES	Não contém construtores.	Contém construtores.

VELOCIDADE	Em geral, são mais lentas que classes abstratas.	Em geral, são mais rápidas que interfaces.

Dessa forma, pode-se dizer que uma interface é similar a uma classe abstrata. Aliás, podemos dizer que uma interface é praticamente uma classe abstrata pura, isto é, todos os seus métodos são abstratos. Uma classe concreta – ao implementar uma interface – deverá escrever o corpo de todos os métodos. **Observem, portanto, que uma classe abstrata pode também implementar uma interface sem nenhum problema.**

Uma aluna certa vez me perguntou a diferença entre classe abstrata e interface e também quando se deve utilizar uma ou quando se deve utilizar a outra. **Galera, lembre-se que a classe abstrata pode conter métodos concretos, então se você quer representar métodos concretos, você pode utilizar classes abstratas, mas não poderá jamais utilizar interfaces** (que não pode conter métodos concretos).

Existe outra diferença importante: você pode declarar variáveis em classes abstratas, mas se você declarar uma variável em uma interface, ela não terá o comportamento de uma variável, mas – sim – de uma constante (será implicitamente *public static final*). **Não é papel da interface lidar com o estado interno de um objeto – é muito raro ver atributos em uma interface.** Por fim, se você usar uma classe abstrata pura, realmente não tem muitas diferenças práticas em relação a interfaces.

De todo modo, conceitualmente, uma classe abstrata especifica o que um objeto é; uma interface especifica o que um objeto pode fazer.

Encapsulamento

INCIDÊNCIA EM PROVA: ALTÍSSIMA

Galera, nós já vimos que objetos possuem comportamentos. *Correto?* **Já vimos que eles realizam operações em outros objetos, conforme recebam mensagens.** O mecanismo de encapsulamento é uma forma de restringir o acesso ao comportamento interno de um objeto. Um objeto que precise da colaboração de outro objeto para realizar alguma operação simplesmente envia uma mensagem a este último.

Segundo o mecanismo do encapsulamento, o método que o objeto requisitado utiliza para realizar a operação não é conhecido dos objetos requisitantes. Em outras palavras, o objeto remetente da mensagem não precisa conhecer a forma pela qual a operação requisitada é realizada; tudo o que importa a esse objeto remetente é obter a operação realizada, não importando como.

No entanto, o remetente da mensagem precisa conhecer pelo menos quais operações o receptor sabe realizar ou o que ele pode oferecer. Para tal, as classes descrevem seus comportamentos por meio de uma interface! Ela descreve o que o objeto sabe fazer, sem precisar detalhar como ele fará! *Vamos ver um exemplo?* Quando enviamos uma encomenda para alguém em outro país, eu pago pelo serviço de entrega internacional oferecido pelos Correios e só...

Eu não quero saber se ele vai de avião, trem, navio, submarino! Como ele fará para entregar minha encomenda não me importa, o que importa é que ele entregue a encomenda. Logo, a interface de um objeto deve definir os serviços que ele pode fornecer. **Através do encapsulamento, a única coisa que um objeto precisa saber para pedir a colaboração de outro objeto é conhecer a sua interface.** Nada mais além disso...

Isso contribui para a autonomia dos objetos, pois cada objeto envia mensagens a outros objetos para realizar certas operações, sem se preocupar em como se realizaram as operações. *Qual a vantagem disso?* A interface permite que a implementação de uma operação pode ser trocada sem que o objeto requisitante dela precise ser alterado. **Isso mantém as partes de um sistema tão independentes quanto possível.**

Daí a importância do mecanismo do encapsulamento no desenvolvimento de software orientado a objetos. Pessoal, existe o conceito de especificadores ou modificadores de acesso. Conhecidos também como visão de método ou visão de atributo, definem a visibilidade de um atributo, método ou classe. **Em geral, utilizam-se especificadores de acesso para privar o acesso direto a atributos e obrigar o usuário a fazê-lo por meio de métodos públicos.**

MODIFICADOR/ESPECIFICADOR			CLASSE	PACOTE	SUBCLASSE	TODOS
UML	PÚBLICO	+	X	X	X	X
	PROTEGIDO	#	X		X	

	PACOTE	~	X	X		
	PRIVADO	-	X			

MODIFICADOR/ESPECIFICADOR			CLASSE	PACOTE	SUBCLASSE	TODOS
JAVA	PÚBLICO	+	X	X	X	X
	PROTEGIDO	#	X	X	X	
	DEFAULT	~	X	X		
	PRIVADO	-	X			

Polimorfismo

INCIDÊNCIA EM PROVA: ALTÍSSIMA

O polimorfismo trata da capacidade de abstrair várias implementações diferentes em uma única interface. **Imaginem que vocês tenham um videocassete antigo e, em determinado dia, ele decida parar de funcionar!** Você o joga no lixo, mas mantém seu controle remoto. Meses depois, compra um blu-ray da mesma marca e, de repente, seu controle remoto antigo funciona também no aparelho novo!

Olha que maneiro! Dois objetos, um novo e um antigo, respondem à mesma mensagem! *E no mundo da orientação a objetos?* Nesse contexto, o polimorfismo diz respeito à capacidade de duas ou mais classes de objetos responderem à mesma mensagem, cada qual de seu próprio modo. **Pensem em uma coleção de formas geométricas que contenha círculos, retângulos e outras formas específicas**.

Seguindo os princípios de polimorfismo na orientação a objetos, nós podemos facilmente calcular a área de todas essas figuras geométricas! No entanto, vocês sabem que o cálculo da área de círculo é diferente do cálculo da área de um retângulo, que é diferente do cálculo da área de um trapézio, que é diferente da área de um triângulo, etc. *Pois é... vamos relembrar um pouquinho das aulas de trigonometria da escola?*

Para calcular a área do círculo, é necessário saber o raio; do retângulo, é necessário saber a base e a altura; do trapézio, é necessário saber a base maior, base menor e altura; e do triângulo, é necessário saber a base e altura. **Ora, para aplicar o polimorfismo, eu devo enviar a mesma mensagem e ele se virar para entender qual área ele deve calcular!** *E como ele faz isso, professor?* Ele vê a lista de parâmetros, isto é, a mensagem enviada!

Se eu enviei um argumento, ele sabe que é o círculo; se eu enviei três argumentos, ele sabe que é o trapézio; se eu enviei dois argumentos, pode ser o retângulo ou o triângulo! *E agora professor?* **Aí temos outro tipo de polimorfismo, que é tratado em tempo de execução.** Em outras palavras, o polimorfismo permite que a mesma mensagem seja enviada a diferentes objetos e que cada objeto execute a operação que é mais apropriada a sua classe.

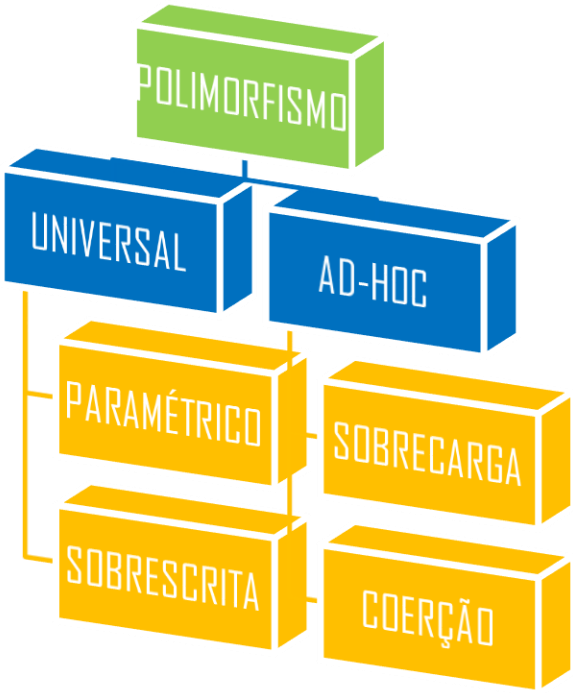
Há uma relação estreita com o conceito de abstração, na medida em que um objeto pode enviar a mesma mensagem para objetos semelhantes, mas que implementam a sua interface de formas diferentes. **O Polimorfismo pode ser Estático ou Dinâmico.** O primeiro é também conhecido como polimorfismo por sobrecarga ou *overloading*, é representado com o nome do método igual e parâmetros diferentes.

A decisão do método a ser chamado é tomada em tempo de compilação de acordo com os argumentos passados. *Professor, o que você quer dizer com parâmetros diferentes?* **Pode ser uma diferença na quantidade, tipo ou ordem dos parâmetros.** O segundo é também conhecido como

polimorfismo por sobrescrita, polimorfismo por inclusão, polimorfismo por herança, polimorfismo por subtipo, redefinição ou *overriding*.

Ele está associado ao conceito de herança e é representado com o nome e parâmetros do método iguais. Nesse caso, a subclasse redefine o método da superclasse e a decisão do método a ser chamado é tomada em tempo de execução. Alguns alunos sempre me perguntam o que é a assinatura de um método! Vejamos: dois métodos possuem a mesma assinatura se, e somente se, tiverem o mesmo nome e os mesmos parâmetros (quantidade, tipo e ordem dos parâmetros).

Algumas linguagens (Ex: Java) ignoram o tipo de retorno para verificar se métodos possuem a mesma assinatura; outras linguagens (Ex: C++) validam também o tipo de retorno para verificar se métodos possuem a mesma assinatura. **Logo, a questão do tipo de retorno não é pacífica e dependerá da linguagem de programação utilizada.** Bem... abaixo eu apresento os principais tipos de polimorfismo:



Comecemos pelo Polimorfimo Universal e Polimorfismo Ad-hoc! Grosso modo, o primeiro pode trabalhar com um número infinito de tipos; já o segundo pode trabalhar com um número finito de tipos. Dentro de cada uma dessas duas categorias, existem mais duas categorias: paramétrico e sobrescrita; sobrecarga e coerção. O polimorfismo paramétrico é também chamado por diversos autores de polimorfismo verdadeiro.

O polimorfismo paramétrico é o tipo de polimorfismo que permite que se escreva um código genérico para servir os subtipos (que só serão descobertos em tempo de execução). Em outras palavras, um mesmo objeto pode ser utilizado como parâmetro em diferentes contextos sem necessidade de quaisquer alterações. Nesse caso, um elemento (Ex: função, classe, método, etc) pode ser escrito genericamente para que possa suportar valores idênticos sem depender do tipo.

Como assim, professor? Vamos imaginar que você tenha listas de diversos tipos: você tem uma lista de carros, uma lista de pessoas, uma lista de animais e uma lista de filmes. Ok! E você sabe que pode realizar várias operações nessas listas, tais como: acessar um elemento da lista, adicionar um elemento na lista, excluir um elemento da lista, atualizar um elemento da lista. **Perceba que essas operações servem para qualquer lista, independentemente de seu tipo.**

Logo, por que criar métodos de acesso, adição, exclusão e atualização para todas as listas em vez de criar apenas um de cada? Veremos que não há necessidade disso! **Uma única função é codificada e ela trabalhará uniformemente em um intervalo de tipos (funções paramétricas também são chamadas de funções genéricas).** Bem, no Java, isso começou na versão 1.5 (com Generics). Lá, existe um tipo genérico chamado <List>. Toda vez que eu preciso instanciar uma lista, basta fazer:

```
List<TipoDaLista> NomeDaLista = new ArrayList<List>();
```

Dessa forma, caso eu queira instanciar uma lista de Strings, Integer ou um tipo criado por mim, basta fazer conforme o código a seguir:

```
List<String> listaDeString = new ArrayList<List>();

List<Integer> listaDeInteger = new ArrayList<List>();

List<TipoMeuQualquer> listaDeTipoMeuQualquer = new ArrayList<List>();
```

E olha o mais legal! Eu não preciso implementar todos os métodos de uma lista para cada tipo específico, basta fazer (por exemplo, para adição):

```
listaDeString.add("Primeira String");

listaDeInteger.add(4.000);

listaDeTipoMeuQualquer.add(TipoMeuQualquer);
```

Diga se isso não é genial e uma mão na roda! Pois é! Bem, o segundo tipo de polimorfismo universal, nós já vimos acima – **a única novidade é que ele é chamado também de polimorfismo por inclusão, polimorfismo por herança ou polimorfismo por subtipo.** Já o Polimorfismo ad-hoc se divide em sobrecarga e coerção – alguns autores mais rigorosos afirmam que polimorfismo ad-hoc não é polimorfismo. *Por que?* Porque não ocorrem em tempo de execução!

Nós já vimos o primeiro tipo anteriormente, então não vamos repetir. Nosso interesse aqui é no polimorfismo de coerção. Ele é suportado através da sobrecarga de operadores, ou seja, ocorre quando se converte um elemento de um tipo no tipo apropriado para o método (é o famoso *casting* implícito). Ele permite que um argumento seja convertido para o tipo esperado por uma função, evitando assim um erro de tipo. Imaginem uma variável do tipo inteiro e uma variável do tipo real.

É possível atribuir um valor inteiro a um tipo real (visto que ele é "maior") de forma implícita. **Nesse momento, ocorre uma coerção (também chamada conversão) de uma variável de um tipo em outro tipo.** Agora vejam que bacana! Já recebi uma dúvida algumas vezes: *Professor, um método pode sobrecarregar um método herdado?* Em nossa página no Facebook, nós já discutimos sobre essa dúvida! Vejamos dois cenários...

Primeiro, Classe Veículo possui um método dirigir(a) e uma Classe Carro (filha de Classe Veículo) possui um método dirigir(a), logo sobrescrevendo o método da classe-pai. Se eu inserir um método dirigir(a,b) na classe-filha, eu posso afirmar que esse método sobrecarrega o método dirigir(a) da classe-filha, mas não da classe-pai. E ele não sobrescreve o método da classe-pai, porque as assinaturas são diferentes.

É impossível que um método realize sobrescrita e sobrecarga simultaneamente sobre um mesmo método. Segundo cenário: imaginem que eu não tenho uma sobrescrita de dirigir(a) na classe-filha. Eu tenho apenas dirigir(a) na classe-pai e dirigir(a,b) na classe-filha. *Posso afirmar que esse método da classe-filha sobrecarrega dirigir(a)?* Sim, porque dirigir(a) é herdado na classe-filha, logo – de certo modo – há uma sobrecarga do método implícito herdado da classe-pai.

Na documentação do Java, há inclusive a seguinte referência:

> *In a subclass, you can overload the methods inherited from the superclass. Such overloaded methods neither hide nor override the superclass instance methods—they are new methods, unique to the subclasse.*

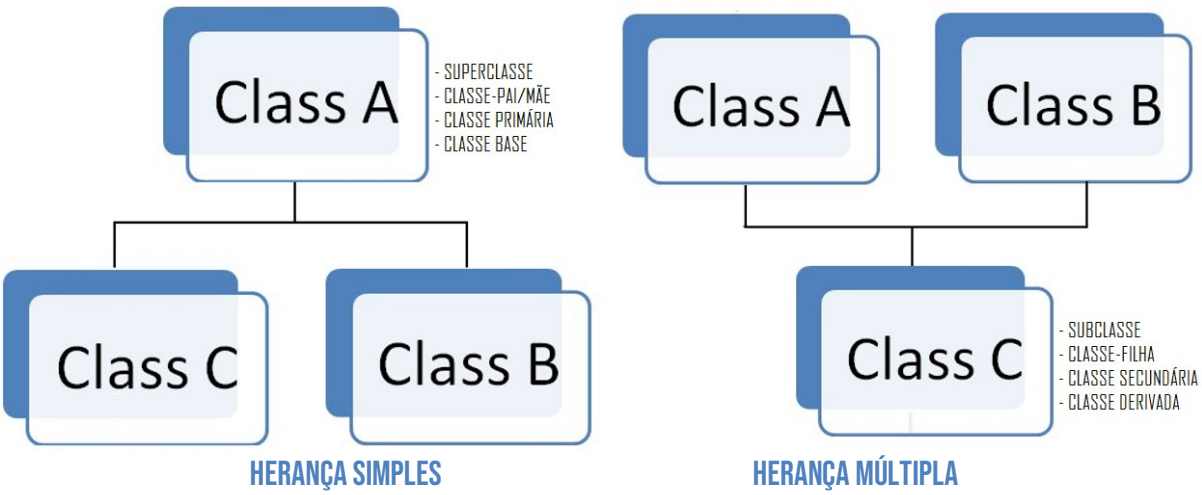
Herança (Generalização/Especialização)

INCIDÊNCIA EM PROVA: ALTÍSSIMA

A herança é outra forma de abstração utilizada na orientação a objetos que pode ser vista como um nível de abstração acima da encontrada entre classes e objetos. **Na Herança, classes semelhantes são agrupadas em uma hierarquia.** Cada nível dessa hierarquia pode ser visto como um nível de abstração. Trata-se de uma relação entre classes e, não, entre objetos. Antes de prosseguir, vamos falar um pouco sobre nomenclatura!

A Classe que herda é chamada Subclasse, Classe-Filha, Classe Secundária ou Classe Derivada. A Classe que é herdada é chamada Superclasse, Classe-Pai/Mãe, Classe Primária ou Classe Base. A semântica do código da herança é variável de acordo com a linguagem de programação utilizada (Ex: em Java, utiliza-se a palavra-chave *extends*). Cada classe em um nível de hierarquia herda as características e o comportamento das classes às quais está associada nos níveis acima dela.

Além disso, essa classe pode definir características e comportamento particulares. Dessa forma, uma classe pode ser criada a partir do reúso da definição de classes preexistentes. **A herança facilita o compartilhamento de comportamento comum entre classes.** Podemos dizer que se trata do mecanismo que permite que classes compartilhem atributos e métodos, com o intuito de reaproveitar o comportamento generalizado ou especializar operações e atributos.



Quando uma subclasse herda diretamente de duas ou mais superclasses, trata-se de herança múltipla; já quando uma subclasse herda diretamente de apenas uma superclasse, trata-se de herança simples. Pegadinha clássica de concursos de tecnologia da informação: *em orientação a objetos, é permitido herança múltipla?* É claro que sim! O lance é que algumas linguagens de programação específicas não implementam herança múltipla (Ex: Java e C#).

Já no contexto genérico de orientação a objetos, é permitida a herança múltipla! *E por que algumas linguagens não a implementam?* **Porque ela pode causar alguns problemas de ambiguidade!** Quando superclasses possuem membros homônimos e a subclasse não redefine esses membros,

no momento em que um objeto da subclasse tentar referenciar diretamente o membro homônimo das superclasses, o compilador não saberá a qual membro ele está se referindo.

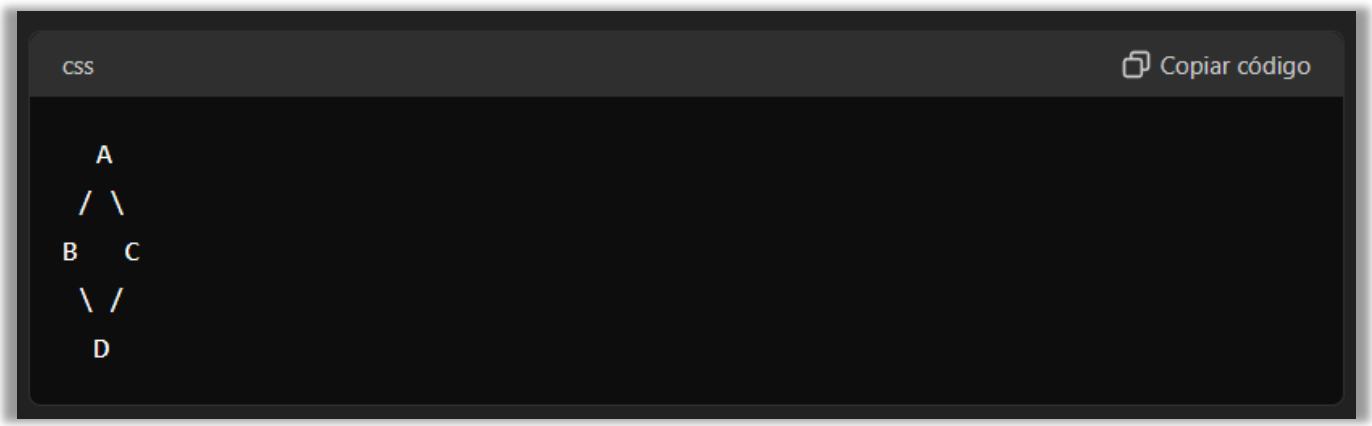
Um detalhe importante: uma subclasse sempre herdará métodos/atributos de suas superclasses – não importa se é herança simples ou herança múltipla. Você não pode dizer que você é um primata, mas não é um mamífero, porque todos os primatas são mamíferos. Vamos aproveitar para definir alguns conceitos importantes. Herdar é diferente de acessar! Se uma classe estende a outra, ela sempre herdará seus métodos/atributos.

Não importa, por exemplo, se eles são privados – a subclasse sempre os herdará, mesmo que não os acesse! *Como eu gosto de visualizar isso?* Imaginem que um tio-avô distante deixe um cofre entupido de dinheiro para vocês. **Por outro lado, esse cofre é completamente indestrutível e ele não deixou nenhuma senha. Nesse caso, vocês herdaram todo o dinheiro, mas não podem acessá-lo.** *Entendido?*

No mundo orientado a objetos acontece a mesma coisa: se uma subclasse é filha de uma superclasse, ela herdará absolutamente tudo, mesmo que ela não consiga acessar (que é o caso de membros privados). Já caiu em prova discursiva: *qual a diferença entre Polimorfismo e Herança*? **Define-se herança como um mecanismo que permite ao programador basear uma nova classe na definição de uma classe previamente existente.**

Usando herança, sua nova classe herda todos os atributos e comportamentos presentes na classe previamente existente. **Quando uma classe herda de outra, todos os métodos e atributos que surgem na interface da classe previamente existente aparecerão automaticamente na interface da nova classe.** Já o polimorfismo permite que um único nome de método represente um código diferente, selecionado por algum mecanismo automático.

Dessa forma, um nome pode assumir muitas formas e, como pode representar código diferente, o mesmo nome pode representar muitos comportamentos diferentes. Por fim, é importante falar do famoso **Problema do Diamante**. *O que é isso, Diego?* Trata-se de um problema específico que ocorre na herança múltipla, quando uma classe herda de duas classes que, por sua vez, herdam da mesma classe base.



Esse problema é assim chamado porque a estrutura de herança resultante se assemelha a um diamante, conforme imagem anterior. Note que a Classe A é a classe base; Classes B e C herdam da Classe A; e Classe D herda de ambas. O problema do diamante surge quando a classe D tenta acessar membros da classe A. Há ambiguidades sobre qual caminho seguir para acessar os membros da classe A: através da classe B ou da classe C.

Essa ambiguidade pode causar problemas de consistência e complicações na manutenção do código. Cada linguagem de programação pode tentar resolver esse problema à sua maneira.

(CESPE / TCDF – 2023) O "problema do diamante", que surge quando uma classe herda de duas classes que compartilham uma mesma classe pai, não resulta em conflitos de métodos ou ambiguidades na resolução de herança múltipla.

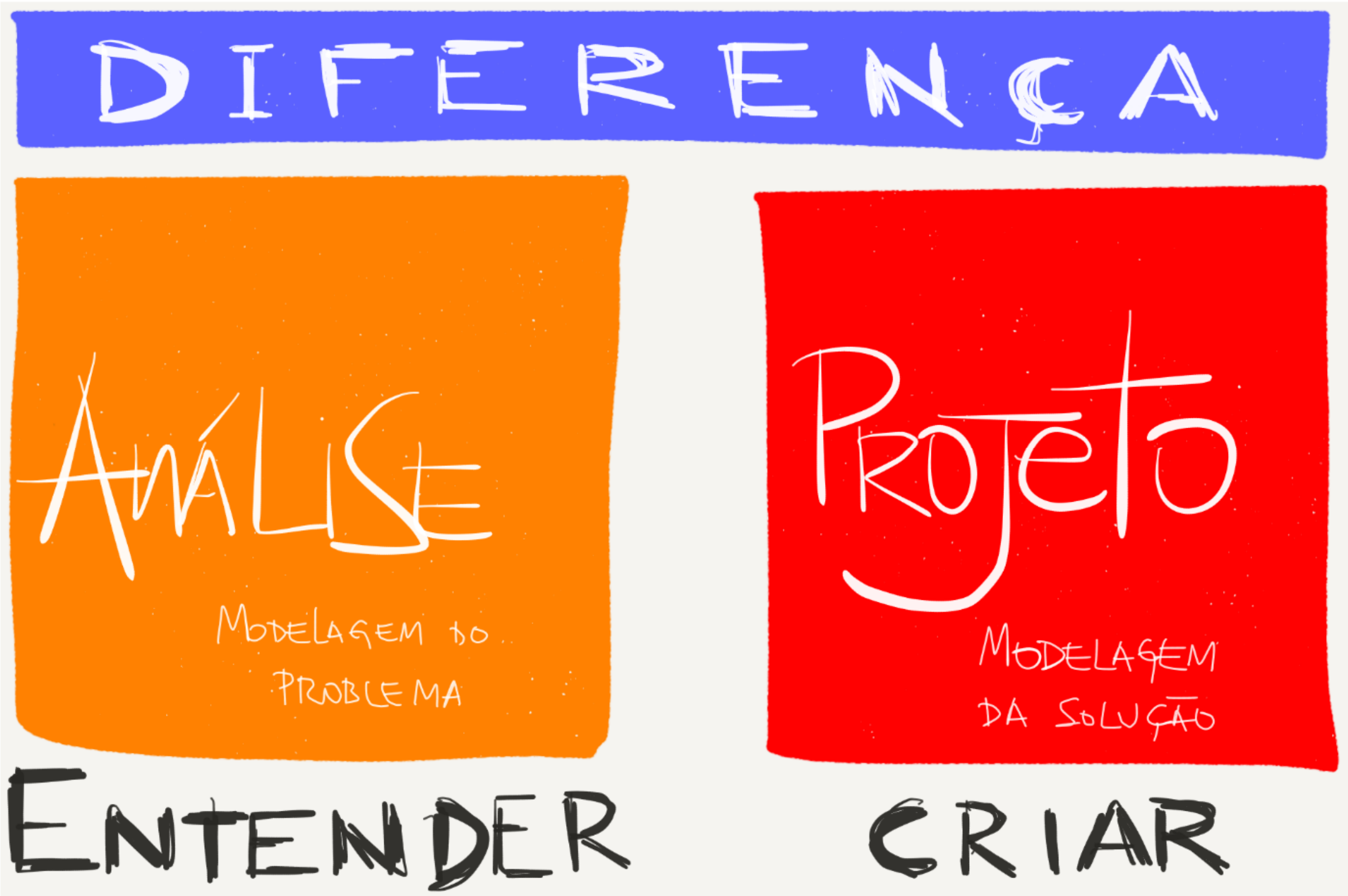
_____

Comentários: o "problema do diamante" ocorre quando uma classe herda de duas classes que têm uma classe pai em comum, e isso pode resultar em ambiguidade de métodos e conflitos na resolução de herança múltipla (Errado).

ANÁLISE E PROJETO

Conceitos Básicos

INCIDÊNCIA EM PROVA: MÉDIA



Bem, vamos começar a distinguir o que é Análise e o que é Projeto. Para tal, **preciso que vocês memorizem, decorem, tatuem, componham uma música com a frase**:

A Análise consiste em atividades necessárias para entender o domínio do problema, isto é, o que deve ser feito. É uma atividade de investigação, **com foco no cliente**. Já o Projeto consiste em atividades necessárias para entender o domínio da solução do problema, isto é, como deve ser feito. É uma atividade técnica, **com foco no programador**. Na Análise, a tecnologia de implementação e os requisitos não-funcionais não são modelados. Essa é uma tarefa do Projeto!

Também não se pensa sobre soluções técnicas, pensa-se apenas em modelar funções, dados e relacionamentos do sistema. O modelo de análise deve ser aprovado pelo cliente – pode incluir até pequenas discussões sobre a solução, como sobre Interfaces Gráficas. O Modelo de Casos de Uso representa o aspecto funcional de um domínio de negócio e, de forma similar, o Modelo de Classes representa o aspecto estrutural de um domínio de negócio. *Como é?*

Pessoal, um exemplo de Modelo de Classes é o Diagrama de Classes (da UML). Ele representa visualmente conceitos de um determinado domínio por meio de classes. É importante notar que o modelo de classes é utilizado durante a maior parte do desenvolvimento de um sistema orientado a objetos. Além disso, ele evolui durante as iterações do desenvolvimento do sistema. **À medida que o sistema é desenvolvido, o Modelo de Classes é incrementado com novos detalhes.**

Existem três estágios sucessivos de abstração: análise, especificação e implementação.

MODELO DE CLASSES DE ANÁLISE (OU DOMÍNIO)	Construído durante a atividade de análise, representa as classes de domínio do negócio. Não leva em consideração restrições inerentes à tecnologia a ser utilizada na solução de um problema.

MODELO DE CLASSES DE ESPECIFICAÇÃO (OU PROJETO)	Construído durante a atividade de projeto, estende o modelo de classes de análise e contém detalhes específicos inerentes à solução de software escolhida.

MODELO DE CLASSES DE IMPLEMENTAÇÃO

Construído durante a atividade de implementação, estende o modelo de classes de projeto e contém detalhes específicos inerentes ao desenvolvimento das classes em alguma linguagem.

```
private class Pessoa {
    public String Nome;
    public String Telefone;
    public String Endereco;
    private Empresa empresa[];

    public String getInfo(){
        return empresa.getConta();
    }

public class Empresa {
    public Number CNPJ;
    public String Endereco;

    public getConta() {
        return 0;
    }
}
```

À medida que o sistema é desenvolvido, o modelo é incrementado com novos detalhes. O Modelo de Análise enfatiza o desenho lógico, a visão externa, conceitual, abstrata, caixa-preta, de alto nível de abstração. O Modelo de Projeto enfatiza o desenho físico, a visão interna, de implementação, concreta, caixa-branca, de baixo nível de abstração. Sabe-se que o Modelo de Análise é mais estável que o Modelo de Projeto.

Sob a perspectiva organizacional, o primeiro sofre bem menos com mudanças tecnológicas, externas, regulatórias, etc. Além disso, é importante salientar que – na prática – muitas pessoas não fazem Análise e já partem para o Projeto. Inclusive, ela é uma disciplina opcional no RUP! *Professor, eu posso não fazer?* Pode, sim. *Então para que ela existe?* Imaginem se ocorre uma revolução tecnológica e todas as organizações comecem a implementar sistemas com um novo paradigma.

Com um modelo de análise pronto, torna-se muito mais fácil adaptar o modelo de projeto. Caso contrário, tem-se que refazer o modelo de projeto do início. **Em geral, as classes de análise evoluem para classes de projeto.** A meta da Análise é identificar um esboço preliminar do comportamento do sistema. A meta do Projeto é transformar esse esboço preliminar em um conjunto implementável.

O resultado é que há um refinamento detalhado e preciso quando alguém se move da Análise para o Design. Pessoal, a Análise apresenta quatro atividades principais:

Em 1992, Ivar Jacobson (aquele da UML) propôs uma técnica chamada Análise de Robustez, que propunha a categorização das classes de acordo com sua responsabilidade: **Classe de Fronteira; Classe de Controle; e Classe de Entidade.**

CLASSE DE FRONTEIRA

Classe utilizada para modelar a interação entre um ator e o sistema. Para cada ator, é identificada pelo menos uma classe de fronteira para permitir sua interação com o sistema. Então, uma classe de fronteira existe para que o sistema se comunique com o mundo exterior, logo elas são altamente dependentes do ambiente.

A Interação entre sistema e atores envolve transformar e converter eventos, bem como observar mudanças na apresentação do sistema (como a interface). As classes de fronteira modelam as partes do sistema que dependem do ambiente. As classes de entidade e de controle modelam as partes que são independentes de fatores externos ao sistema. Logo, alterar a GUI ou o protocolo de comunicação significa alterar só classes de fronteira e, não, classes de entidade e de controle.

As classes de fronteira também facilitam a compreensão do sistema, pois definem suas fronteiras. Elas ajudam no design, fornecendo um bom ponto de partida para identificar serviços relacionados. **Algumas classes de fronteira comuns são janelas, protocolos de comunicação, interfaces de impressora, sensores e terminais.** Se você estiver usando um construtor GUI, não será necessário modelar partes da interface de rotinas (botões, por exemplo) como classes de fronteira separadas.

Em geral, a janela inteira é o objeto de fronteira mais refinado. **As classes de fronteira também são úteis para capturar interfaces para APIs possivelmente não orientadas a objetos (como código mais antigo, por exemplo).** Você deve modelar as classes de fronteira de acordo com o tipo de fronteira que elas representam. A comunicação com outro sistema e a comunicação com um ator humano (através de uma interface do usuário) têm objetivos diferentes.

Durante a modelagem da interface do usuário, a principal preocupação deve ser a forma como a interface será apresentada ao usuário. Durante a modelagem da comunicação do sistema, a principal preocupação deve ser o protocolo de comunicação. **Um objeto de fronteira poderá durar mais que uma instância de caso de uso, porém costumam ter a mesma duração da instância de caso de uso.**

CLASSE DE CONTROLE

Classe utilizada para controlar a lógica de execução ou negócio correspondente a cada caso de uso. Servem como uma ponte de comunicação entre objetos de fronteira e objetos de entidade. Decidem o que o sistema deve fazer quando um evento externo relevante ocorre, agindo como coordenador para a realização de casos de uso.

Como objetos de controle (instâncias de classes de controle) geralmente controlam outros objetos, o comportamento de objetos de controle é do tipo coordenador. **As classes de controle encapsulam um comportamento específico de caso de uso.** O comportamento de um objeto de controle está estreitamente relacionado à realização de um caso de uso específico. Em muitos cenários, é possível até dizer que os objetos de controle "executam" as realizações de casos de uso.

No entanto, se as tarefas de caso de uso estiverem intrinsecamente relacionadas, alguns objetos de controle poderão participar de mais de uma realização de casos de uso. Além disso, vários objetos de controle de diferentes classes de controle podem participar de um único caso de uso. Nem todos os casos de uso exigem um objeto de controle. Se o fluxo de eventos em um caso de uso estiver relacionado a um objeto de entidade, um objeto de fronteira poderá realizar o caso de uso em cooperação com o objeto de entidade.

As classes de controle podem ajudar a entender o sistema, pois representam a dinâmica do sistema, controlando as principais tarefas e os fluxos de controle. Quando o sistema executar o caso de uso, um objeto de controle normalmente será criado. Os objetos de controle geralmente desaparecem após a execução do correspondente caso de uso. Observe que uma classe de controle não controla tudo o que é necessário em um caso de uso. Em vez disso, ela coordena as atividades de outros objetos que implementam a funcionalidade.

CLASSE DE ENTIDADE	Classe utilizada para armazenar a informação que é manipulada ou processada pelo caso de uso, partindo do domínio do negócio. Geralmente, essas classes armazenam informações persistentes. Há várias instâncias ou objetos de uma mesma classe de entidade coexistindo dentro do sistema.

Os objetos de entidade (instâncias de classes de entidade) são usados para manter e atualizar informações sobre alguns fenômenos, como um evento, uma pessoa ou algum objeto real. Esses objetos geralmente são persistentes, precisando de atributos e relacionamentos durante muito tempo, algumas vezes durante todo o ciclo de vida do sistema. Um objeto de entidade geralmente não é específico para uma realização de casos de uso.

Às vezes, um objeto de entidade não é nem mesmo específico para o próprio sistema. Os valores de seus atributos e relacionamentos costumam ser fornecidos por um ator. Um objeto de entidade também pode ajudar a executar tarefas internas do sistema. Seu comportamento pode ser tão complicado quanto o de outros estereótipos de objeto. No entanto, ao contrário de outros objetos, esse comportamento está relacionado ao fenômeno que o objeto de entidade representa.

Os objetos de entidade independem do ambiente (os atores). Os objetos de entidade representam os conceitos-chave do sistema que está sendo desenvolvido. Exemplos típicos de classes de entidade em um sistema bancário são Conta e Cliente. Em um sistema de gerenciamento de redes, os exemplos são Nó e Link. Se o fenômeno que você deseja modelar não for usado por outras classes, será possível modelá-lo como um atributo de uma classe de entidade ou mesmo como um relacionamento entre classes de entidade.

Por outro lado, se o fenômeno for usado por qualquer outra classe do modelo de design, será preciso modelá-lo como uma classe. As classes de entidade fornecem um outro ponto de vista do sistema, pois mostram a estrutura lógica dos dados, que pode ajudá-lo a compreender o que o sistema deve oferecer aos usuários. A segunda atividade se refere à identificação de responsabilidades.

Essa identificação e categorização implica que cada classe seja especialista em realizar uma tarefa específica que comporá o modelo de análise: comunicar-se com atores (Fronteira); manter as informações do sistema (Entidade); e coordenar a realização de um caso de uso (Controle). **As atividades seguintes são bastante intuitivas e fáceis de entender.** A terceira atividade se refere à identificação de atributos, em que se busca apenas descobrir quais são os atributos, sem nenhuma preocupação sobre qual seu tipo (String, Date, Time, Integer, etc).

Um bom conhecimento do domínio do problema é extremamente útil nesta fase. Por fim, identificam-se os relacionamentos como associações, agregações, composições, dependências, generalizações, especializações, entre outros. Fim da Análise, agora nós sabemos o que fazer, chegou a hora de saber como fazer. **Partamos, então, para o Projeto! Vamos falar bastante sobre arquitetura de software, estrutura e comportamento de classes.**

A Arquitetura de Software é a organização/estrutura dos componentes significativos do sistema que interagem por meio de interfaces. Uma arquitetura bem projetada e solidamente desenhada deve ser capaz de atender aos requisitos funcionais e não-funcionais e ser suficientemente flexível para suportar requisitos voláteis. A arquitetura é importante, pois permite uma comunicação efetiva entre as partes interessadas, abrangendo a compreensão, negociação e consenso.

Ademais, permite decisões tempestivas, isto é, possibilita correção e validação do sistema antes da implementação. Por fim, permite uma reutilizável em sistemas com características similares. Uma boa arquitetura deve ter componentes projetados com baixo acoplamento e alta coesão. *Como é isso, professor?* Pessoal, esse é outro mantra que eu preciso que vocês memorizem! Acoplamento trata do nível de dependência entre módulos de um software.

Já a Coesão trata do nível de responsabilidade de um módulo em relação a outros. *Professor, por que é bom ter baixo acoplamento?* **Porque se os módulos pouco dependem um do outro, modificações de um não afetam os outros, além de prejudicar o reúso**. *Professor, por que é bom ter alta coesão?* Porque se os módulos têm responsabilidades claramente definidas, eles serão altamente reusáveis independentes e simples de entender.

Uma forma de organizar a arquitetura de um sistema complexo em partes menores é por meio de camadas, em que cada uma corresponderá a um conjunto de funcionalidades de um sistema– sendo que as funcionalidades de alto nível dependerão das funcionalidades e baixo nível. A separação em camadas fornece um nível de abstração através do agrupamento lógico de subsistemas relacionados.

Parte-se do princípio de que camadas de abstração mais altas devem depender das camadas de abstração mais baixas. Isso permite que o sistema de software seja mais portável e modificável. Mudanças em uma camada mais baixa, que não afetem a sua interface, não implicarão mudanças nas camadas superiores; e mudanças em uma camada mais alta, que não impliquem a criação de um novo serviço em uma camada mais baixa, não afetarão camadas inferiores.

A arquitetura em camadas permite melhor separação de responsabilidades; decomposição de complexidade; encapsulamento de implementação; maior reúso e extensibilidade. No entanto, podem penalizar o desempenho do sistema e aumentar o esforço/complexidade de desenvolvimento do software. Arquiteturas em duas camadas já foram dominantes, mas – para minimizar o impacto de mudanças – decidiu-se separar a camada de negócio da camada de interface gráfica, gerando três camadas:

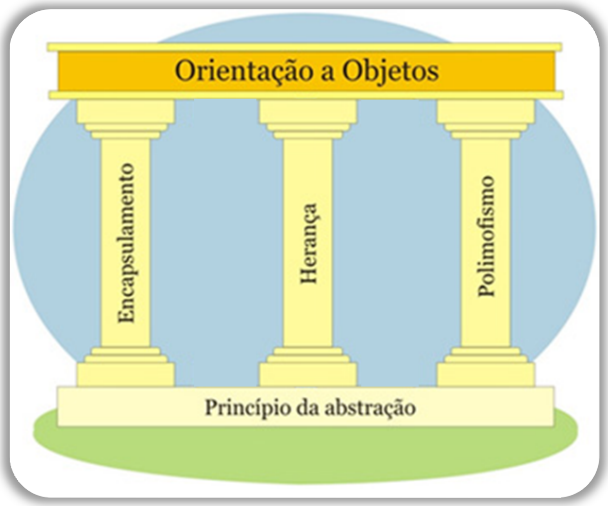
CAMADA DE APRESENTAÇÃO	Possui classes que contêm funcionalidades para visualização dos dados pelos usuários (Por exemplo: classes de fronteiras para atores humanos). E qual a real importância dela? Ela tem o objetivo de exibir informações ao usuário e traduzir ações do usuário em requisições às demais partes dos sistemas.
CAMADA LÓGICA DE NEGÓCIO	Possui classes que implementam as regras de negócio no qual o sistema será implantado. Ela realiza computações com base nos dados armazenados ou nos dados de entrada, decidindo que parte da camada de acesso deve ser ativada com base em requisições provenientes da camada de apresentação.
CAMADA DE ACESSO	Possui classes que se comunicam com outros sistemas para realizar tarefas ou adquirir informações para o sistema. Tipicamente, essa camada é implementada utilizando algum mecanismo de armazenamento persistente. Pode haver uma subcamada dentro desta camada chamada Camada de Persistência.

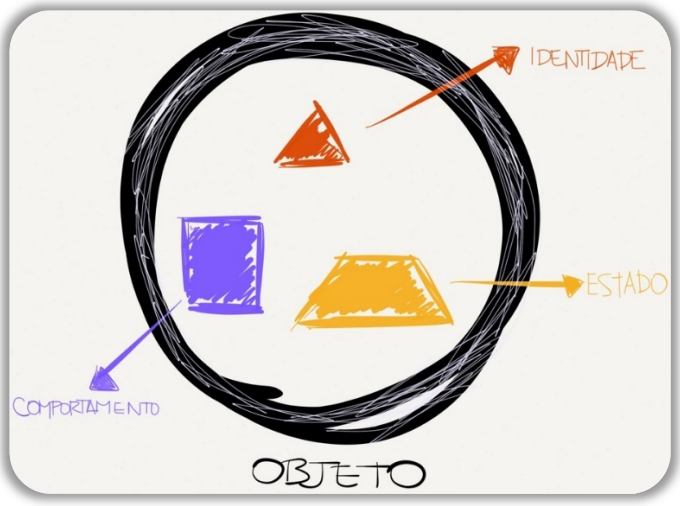
O padrão de arquitetura em três camadas mais utilizado no mercado é designado Model-View-Controler (MVC):

MODELO	Responsável por modelar os dados da aplicação e regras de negócio. Tem o foco em armazenamento, manipulação e geração de dados. Objetos do Modelo são geralmente reusáveis, distribuídos, persistentes e portáveis.
VISÃO	Responsável pela apresentação dos dados aos usuários. Ele recebe entradas de dados e apresenta resultados. Essa camada não persiste nenhum dado no sistema e também não busca dados, apenas os renderiza em tela.
CONTROLE	Responsável por definir o comportamento da aplicação. Processa e responde a eventos, geralmente ações do usuário, e pode invocar alterações no modelo. Realiza, também, a validação de dados do usuário.

RESUMO

VANTAGENS DO PARADIGMA ORIENTADO A OBJETOS
Produção de software natural. Os programas naturais são mais inteligíveis. Em vez de programar em termos de regiões de memória, o profissional pode programar usando a terminologia de seu problema em particular.
Programas orientados a objetos, bem projetados e cuidadosamente escritos são confiáveis.
Pode-se reutilizar prontamente classes orientadas a objetos bem-feitas. Assim como os módulos, os objetos podem ser reutilizados em muitos programas diferentes.
Um código orientado a objetos bem projetado é manutenível. Para corrigir um erro, o programador simplesmente corrige o problema em um lugar. Como uma mudança na implementação é transparente, todos os outros objetos se beneficiarão automaticamente do aprimoramento.
O software não é estático. Ele deve crescer e mudar com o passar do tempo, para permanecer útil. A programação orientada a objetos apresenta ao programador vários recursos para estender código. Esses recursos incluem herança, polimorfismo, sobreposição e uma variedade de padrões de projeto.
O ciclo de vida do projeto de software moderno é frequentemente medido em semanas. A programação orientada a objetos ajuda nesses rápidos ciclos de desenvolvimento. Ela diminui o tempo do ciclo de desenvolvimento, fornecendo software confiável, reutilizável e facilmente extensível.





TIPO DE ATRIBUTO	DESCRIÇÃO
ATRIBUTO DE CLASSE	Similar a uma variável global, trata-se de uma variável cujo valor é comum a todos os objetos membros de uma classe. Mudar o valor de uma variável de classe em um objeto membro automaticamente muda o valor para todos os objetos membros.
ATRIBUTO DE INSTÂNCIA	Trata-se de uma variável cujo valor é específico ao objeto e, não, à classe. Em geral, possui um valor diferente para cada instância. As linguagens de programação possuem palavras para definir o escopo da variável (Ex: em Java, por padrão, é de instância; para ser de classe, deve vir precedida de *static*).

TIPO DE MÉTODO	DESCRIÇÃO
MÉTODO DE CLASSE	Similar a um método global, trata-se de um método que realiza operações genéricas, isto é, não relativas a uma instância particular. Linguagens de programação possuem palavras para definir o escopo do método (Ex: em Java, deve vir precedida de *static*).
MÉTODO DE INSTÂNCIA	Similar a um método local, trata-se de um método que realiza operações específicas para um objeto e, não, à classe, isto é, são relativas a uma instância particular. Por padrão, todos os métodos de uma determinada classe são considerados métodos de instância.

TIPO DE LIGAÇÃO	DESCRIÇÃO
EARLY BINDING	Também conhecida como Ligação Estática, ocorre quando o método a ser invocado é definido em tempo de compilação.
LATE BINDING	Também conhecida como Ligação Dinâmica, ocorre quando o método a ser invocado é definido em tempo de execução.

MENSAGEM	Trata-se de requisições enviadas de um objeto para outro com o intuito de receber algo em retorno por meio da execução de uma operação. Uma mensagem é composta por: objeto, a quem a mensagem é endereçada; nome do método ou serviço que se deseja executar; e parâmetros necessários ao método (se existirem).
ABSTRAÇÃO	Trata-se da habilidade de concentrar nos aspectos essenciais de um contexto qualquer, ignorando características menos importantes ou acidentais. Em modelagem orientada a objetos, uma classe é uma abstração de entidades existentes no domínio do sistema de software.
INTERFACE	Trata-se de um contrato sem implementação entre dois ou mais objetos. É utilizada para reduzir o acoplamento, facilitando o reuso de classes. Elas não possuem atributos, apenas assinaturas dos métodos – sendo que todos os métodos são, por padrão, *abstract* e *public*.

CARACTERÍSTICAS	INTERFACES	CLASSE ABSTRATA
HERANÇA MÚLTIPLA	Suporta Herança Múltipla. Pode implementar diversas interfaces.	Não suporta Herança Múltipla. Não pode estender várias classes abstratas.
IMPLEMENTAÇÃO	Não pode conter qualquer método concreto, apenas abstratos.	Pode conter métodos concretos ou abstratos.
CONSTANTES	Suporta somente constantes estáticas.	Suporta constantes estáticas e de instância.
ENCAPSULAMENTO	Métodos e membros devem sempre ser públicos por padrão.	Métodos e membros podem ter qualquer visibilidade.
MEMBROS DE DADOS	Não contém atributos, apenas assinatura de métodos.	Pode conter atributos.

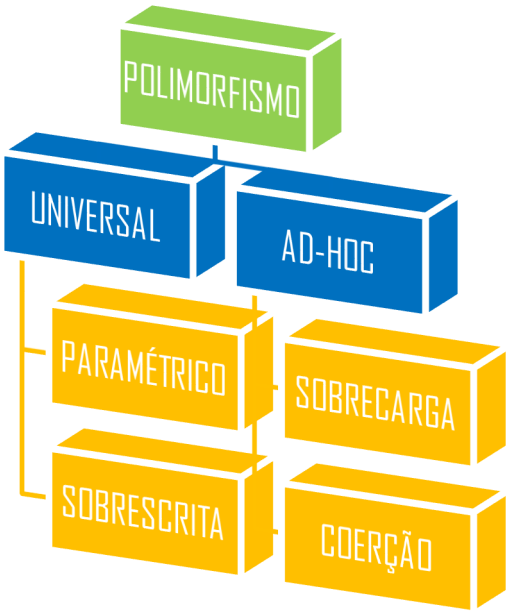
CONSTRUTORES	Não contém construtores.	Contém construtores.
VELOCIDADE	Em geral, são mais lentas que classes abstratas.	Em geral, são mais rápidas que interfaces.

ENCAPSULAMENTO	Trata-se de uma forma de restringir o acesso ao comportamento interno de um objeto de modo a evitar que eles sofram acessos indevidos. A ideia é tornar o software mais flexível, fácil de modificar e de criar novas implementações.

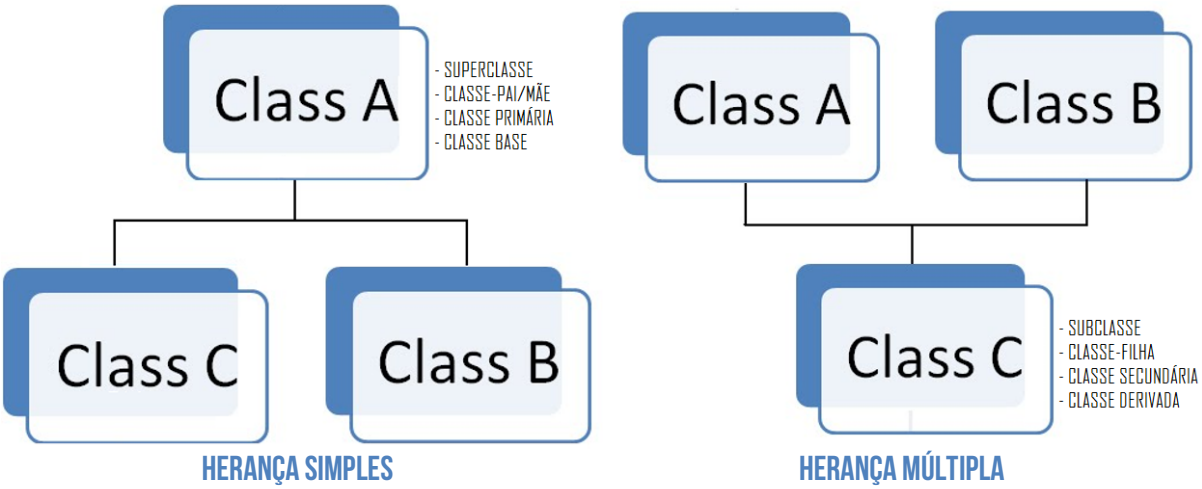
MODIFICADOR/ESPECIFICADOR			CLASSE	PACOTE	SUBCLASSE	TODOS
UML	PÚBLICO	+	X	X	X	X
	PROTEGIDO	#	X		X	
	PACOTE	~	X	X		
	PRIVADO	–	X			

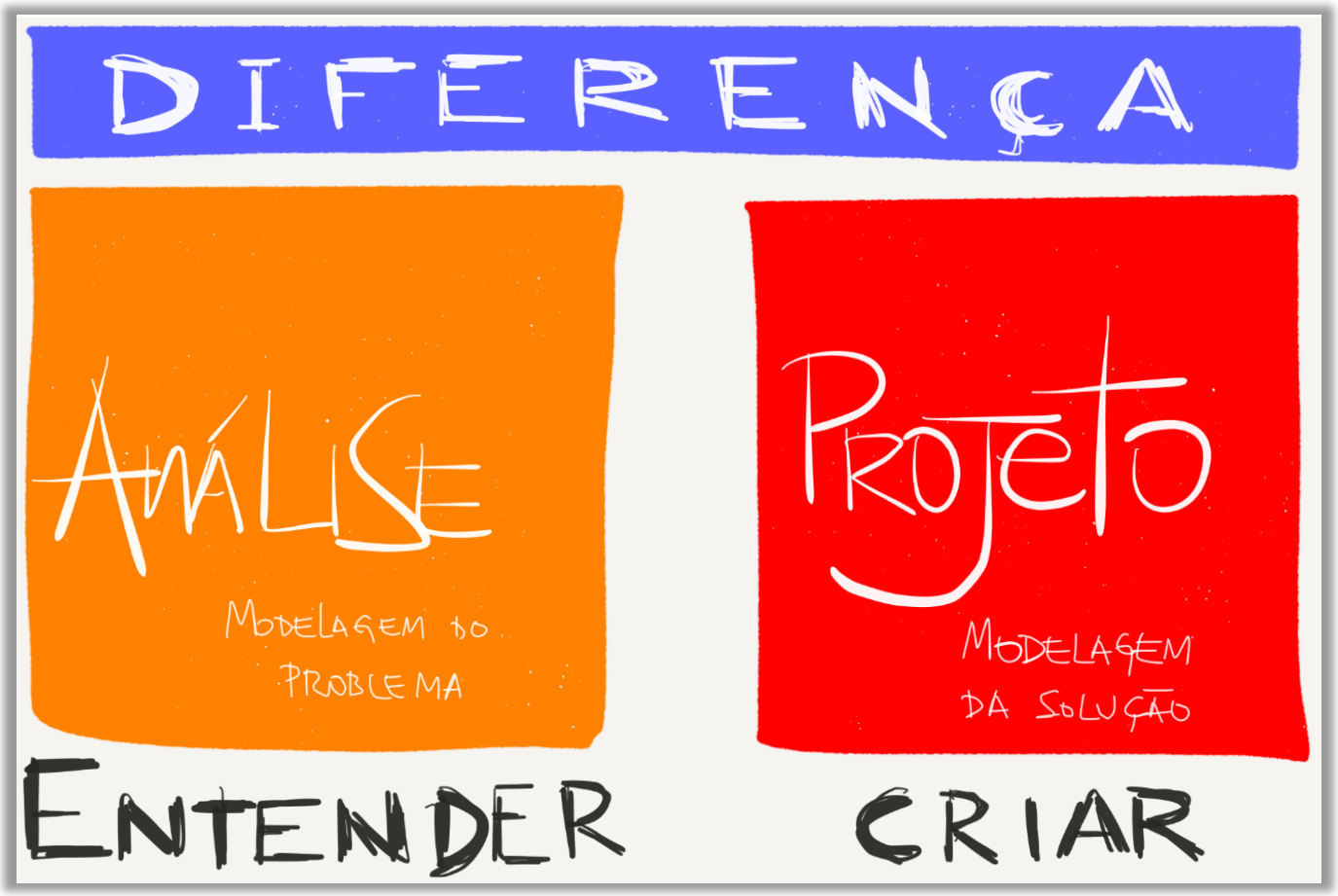
MODIFICADOR/ESPECIFICADOR			CLASSE	PACOTE	SUBCLASSE	TODOS
JAVA	PÚBLICO	+	X	X	X	X
	PROTEGIDO	#	X	X	X	
	DEFAULT	~	X	X		
	PRIVADO	-	X			

POLIMORFISMO	Trata-se capacidade de abstrair várias implementações diferentes em uma única interface. Em outras palavras, é o princípio pelo qual duas ou mais classes derivadas da mesma superclasse podem invocar métodos que têm a mesma assinatura, mas comportamentos distintos.



HERANÇA	Trata-se do princípio que permite que classes compartilhem atributos e métodos. É utilizada com a intenção de reaproveitar código ou comportamento generalizado ou especializar operações ou atributos.





MODELO DE CLASSES DE ANÁLISE (OU DOMÍNIO)

Construído durante a atividade de análise, representa as classes de domínio do negócio. Não leva em consideração restrições inerentes à tecnologia a ser utilizada na solução de um problema.

MODELO DE CLASSES DE ESPECIFICAÇÃO (OU PROJETO)

Construído durante a atividade de projeto, estende o modelo de classes de análise e contém detalhes específicos inerentes à solução de software escolhida.

MODELO DE CLASSES DE IMPLEMENTAÇÃO

Construído durante a atividade de implementação, estende o modelo de classes de projeto e contém detalhes específicos inerentes ao desenvolvimento das classes em alguma linguagem.

CLASSE DE FRONTEIRA

Classe utilizada para modelar a interação entre um ator e o sistema. Para cada ator, é identificada pelo menos uma classe de fronteira para permitir sua interação com o sistema. Então, uma classe de fronteira existe para que o sistema se comunique com o mundo exterior, logo elas são altamente dependentes do ambiente.

CLASSE DE CONTROLE

Classe utilizada para controlar a lógica de execução ou negócio correspondente a cada caso de uso. Servem como uma ponte de comunicação entre objetos de fronteira e objetos de entidade. Decidem o que o sistema deve fazer quando um evento externo relevante ocorre, agindo como coordenador para a realização de casos de uso.

CLASSE DE ENTIDADE	Classe utilizada para armazenar a informação que é manipulada ou processada pelo caso de uso, partindo do domínio do negócio. Geralmente, essas classes armazenam informações persistentes. Há várias instâncias ou objetos de uma mesma classe de entidade coexistindo dentro do sistema.

CAMADA DE APRESENTAÇÃO	Possui classes que contêm funcionalidades para visualização dos dados pelos usuários (Por exemplo: classes de fronteiras para atores humanos). E qual a real importância dela? Ela tem o objetivo de exibir informações ao usuário e traduzir ações do usuário em requisições às demais partes dos sistemas.
CAMADA LÓGICA DE NEGÓCIO	Possui classes que implementam as regras de negócio no qual o sistema será implantado. Ela realiza computações com base nos dados armazenados ou nos dados de entrada, decidindo que parte da camada de acesso deve ser ativada com base em requisições provenientes da camada de apresentação.
CAMADA DE ACESSO	Possui classes que se comunicam com outros sistemas para realizar tarefas ou adquirir informações para o sistema. Tipicamente, essa camada é implementada utilizando algum mecanismo de armazenamento persistente. Pode haver uma subcamada dentro desta camada chamada Camada de Persistência.

MODELO	Responsável por modelar os dados da aplicação e regras de negócio. Tem o foco em armazenamento, manipulação e geração de dados. Objetos do Modelo são geralmente reusáveis, distribuídos, persistentes e portáveis.
VISÃO	Responsável pela apresentação dos dados aos usuários. Ele recebe entradas de dados e apresenta resultados. Essa camada não persiste nenhum dado no sistema e também não busca dados, apenas os renderiza em tela.
CONTROLE	Responsável por definir o comportamento da aplicação. Processa e responde a eventos, geralmente ações do usuário, e pode invocar alterações no modelo. Realiza, também, a validação de dados do usuário.

QUESTÕES COMENTADAS – CESPE

1. **(CESPE / CAU-BR – 2024)** Em sistemas orientados a objetos, os objetos podem ser de natureza física, por exemplo, cadeira, ou de natureza conceitual, por exemplo, inscrição em um curso.

Comentários:

Em sistemas orientados a objetos, os objetos podem representar tanto entidades físicas, como uma cadeira, quanto entidades conceituais, como uma inscrição em um curso. Esses objetos encapsulam dados e comportamentos relacionados, permitindo uma modelagem mais próxima da realidade do problema.

Gabarito: Correto

---

2. **(CESPE / TC-DF – 2023)** O "problema do diamante", que surge quando uma classe herda de duas classes que compartilham uma mesma classe pai, não resulta em conflitos de métodos ou ambiguidades na resolução de herança múltipla.

Comentários:

O "problema do diamante" ocorre na herança múltipla quando uma classe herda de duas classes que compartilham uma mesma classe pai. Isso pode resultar em conflitos de métodos ou ambiguidades, pois a classe final pode herdar o mesmo método ou atributo de forma duplicada, causando incerteza sobre qual implementação deve ser usada. Linguagens como C++ e Python têm mecanismos específicos para resolver essas ambiguidades, enquanto linguagens como Java evitam a herança múltipla de classes para prevenir esse problema.

Gabarito: Errado

---

3. **(CESPE / TC-DF – 2023)** Considere-se que, em um encapsulamento de uma classe de nome Carro para um sistema de automóveis, exista um atributo privado de nome quilometragem. Nesse caso, ao se fornecerem métodos públicos do tipo obter_quilometragem() e atualizar_quilometragem(), protegem-se detalhes internos da classe Carro.

Comentários:

Ao encapsular o atributo privado `**quilometragem**` na classe `**Carro**` e fornecer métodos públicos `**obter_quilometragem**()` e `**atualizar_quilometragem**()`, você protege os detalhes internos da classe. Esse encapsulamento garante que o acesso e a modificação do atributo `quilometragem` sejam controlados, permitindo validações ou lógica adicional nos métodos públicos, preservando a integridade e a consistência do estado interno da classe.

Gabarito: Correto

---

4. **(CESPE / MPE-RO – 2023)** Em orientação a objetos, o conceito utilizado para descrever os vários comportamentos que um método possui, visando a um melhor aproveitamento de partes de um código, é denominado:

a) herança.
b) encapsulamento.
c) atributos.
d) polimorfismo.
e) interface.

Comentários:

(a) Errado. Herança é um conceito de orientação a objetos que permite a criação de novas classes baseadas em classes existentes, permitindo reutilização de código e especialização. Porém, não descreve o comportamento variável de um método.

(b) Errado. Encapsulamento é o princípio de restringir o acesso direto a alguns dos componentes de um objeto, protegendo os dados e a lógica interna. Não está relacionado diretamente com a variação de comportamento de métodos.

(c) Errado. Atributos são características ou propriedades de um objeto que armazenam dados. Eles não descrevem comportamentos ou métodos.

(d) Correto. Polimorfismo é a capacidade de um método ter diferentes comportamentos dependendo do objeto ou do contexto em que é chamado. Permite que métodos com o mesmo nome possam executar diferentes tarefas em classes diferentes, promovendo flexibilidade e reutilização de código.

(e) Errado. Interface define um conjunto de métodos que uma classe deve implementar, mas não é o conceito que descreve comportamentos variados de um método em si.

Gabarito: Letra D

---

5. **(CESPE / DATAPREV – 2023)** No paradigma da orientação a objetos, o polimorfismo permite que várias operações distintas possuam o mesmo nome, desacoplando, assim, os objetos uns dos outros, tornando-os mais independentes.

Comentários:

No paradigma da orientação a objetos, o polimorfismo permite que diferentes operações ou métodos compartilhem o mesmo nome, mas com implementações distintas. Isso desacopla os objetos, permitindo que sejam mais independentes e flexíveis, pois o comportamento específico é

determinado em tempo de execução. Essa característica é essencial para a extensibilidade e a manutenção de sistemas orientados a objetos.

Gabarito: Correto

---

6. **(CESPE / DATAPREV – 2023)** Herança é uma característica do paradigma orientado a objetos, a qual possibilita que haja hierarquia de classes, de forma que as alterações em uma classe-pai possam ser imediatamente propagadas para a classe-filha.

Comentários:

Herança é uma característica do paradigma orientado a objetos que permite criar uma hierarquia de classes, onde uma classe-filha herda atributos e métodos de uma classe-pai. Alterações na classe-pai, como métodos ou propriedades, podem ser imediatamente propagadas para as classes-filha, permitindo reutilização de código e facilitando a manutenção e extensão de funcionalidades.

Gabarito: Correto

---

7. **(CESPE / BANRISUL – 2022)** Nas classes abstratas, que servem de modelo para outras classes, é obrigatória a existência de pelo menos um método abstrato, sem corpo.

Comentários:

Classes abstratas em programação orientada a objetos (POO) são usadas como base para outras classes, mas não é obrigatório que tenham métodos abstratos, embora essa seja uma prática comum. Um método abstrato é um método sem implementação na classe abstrata, deixando para as subclasses a responsabilidade de sua implementação. Uma classe abstrata pode conter métodos abstratos e não abstratos. Em Java, por exemplo, não é necessário ter nenhum método abstrato para que uma classe seja considerada abstrata.

A banca deve ter utilizado como fonte o Deitel, que afirma: "*Uma classe abstrata* ***normalmente*** *contém um ou mais métodos abstratos*". Logo, discordo do gabarito oficial.

Gabarito: Correto

---

8. **(CESPE / BANRISUL – 2022)** Em um projeto orientado a objetos, a decomposição do sistema em objetos é influenciada por fatores como encapsulamento, granularidade e desempenho.

Comentários:

Esses fatores podem determinar a maneira como os objetos são modelados, assim como a arquitetura do sistema. O encapsulamento ajuda a definir quais atributos e comportamentos são associados a cada objeto, enquanto a granularidade ajuda a determinar quão complexos ou simples

os objetos são. Por fim, o desempenho influencia na quantidade de objetos criados no sistema, pois objetos muito complexos podem ter um impacto negativo na performance.

Gabarito: Correto

---

9. (CESPE / TELEBRÁS - 2021) Na orientação a objetos, o polimorfismo permite que os programadores definam comportamentos diferentes para um mesmo método. Além disso, dados os tipos de polimorfismo, é possível que, dentro de uma herança, um comportamento seja reescrito à medida que a classe se torna mais específica, possibilitando que seja criada uma abstração mais próxima do mundo real, facilitando a compreensão do código como um todo. No polimorfismo, a decisão sobre qual método deve ser selecionado é tomada em tempo de concepção.

Comentários:

Trata-se do polimorfismo por sobrescrita. Esse tipo de polimorfismo está associado ao conceito de herança e é representado com o nome e parâmetros do método iguais. Nesse caso, a subclasse redefine o método da superclasse e a decisão do método a ser chamado é tomada em tempo de execução.

Gabarito: Errado

---

10. (CESPE / Petrobrás - 2022) A POO é embasada nos conceitos de classe, objeto, encapsulamento, herança, interfaces e polimorfismo; uma característica das interfaces, por exemplo, é o fato de que elas não podem ser implementadas por uma classe, mas sim herdadas.

Comentários:

A questão começa bem: de fato, todos esses conceitos fazem parte da Programação Orientada a Objetos (POO). No entanto, a parte final está errada: as interfaces podem – sim – serem implementadas pelas classes. Lembrem-se que uma interface é um contrato para que alguma subclasse concretize seus métodos.

Gabarito: Errado

---

11. (CESPE / FUB – 2018) Na orientação a objetos, estes possuem diversos atributos e métodos, os quais são utilizados para se definir as características e ações das classes.

Comentários:

O examinador inverteu os conceitos de objeto e classes. Na verdade, as classes é que possuem atributos e métodos, os quais são utilizados para se definir as características e ações dos objetos.

Gabarito: Errado

---

12. (CESPE / BNB – 2018) O encapsulamento em uma classe garante que seus métodos e suas variáveis tenham alta coesão e baixo acoplamento, seguindo os objetivos básicos da programação orientada a objetos.

Comentários:

A coesão trata do nível de responsabilidade de um módulo em relação a outro, já acoplamento trata do nível de dependência entre os módulos. Como o encapsulamento preconiza que um objeto deve esconder sua complexidade interna não é possível dizer que ele garante alta coesão e baixo acoplamento. Não há essa relação.

Gabarito: Errado

---

13. (CESPE / BNB – 2018) As interfaces são definições a respeito de como um objeto pode ser utilizado por outros objetos, sem envolver necessariamente uma interação com o usuário.

Comentários:

Perfeito! Uma interface é uma classe abstrata pura que especifica o que um objeto pode fazer. Além disso, as interfaces não lidam com o estado interno dos objetos. Em suma: a interface de um objeto define os serviços que ele pode fornecer.

Gabarito: Correto

---

14. (CESPE / BNB – 2018) De acordo com o conceito de herança, uma classe derivada é uma implementação mais genérica da classe da qual ela deriva, o que permite a reutilização de métodos e de variáveis.

Comentários:

A herança está ligada a especialização, logo é errado dizer que uma classe derivada é uma implementação mais genérica da classe-pai quando, de fato, ela é mais específica.

Gabarito: Errado

---

15. (CESPE / BNB – 2018) Em programação orientada a objetos, a técnica utilizada para esconder detalhes internos de funcionamento de uma classe é denominada generalização.

Comentários:

Opa! Na verdade, trata-se do Encapsulamento, que é uma forma de restringir o acesso ao comportamento interno de um objeto

Gabarito: Errado

---

16. (CESPE / TCE-MG – 2018) Em uma programação orientada a objetos, a técnica de programação que mantém ocultos detalhes internos do funcionamento dos métodos de uma classe é denominada

a) encapsulamento.
b) polimorfismo.
c) generalização.
d) abstração.
e) herança.

Comentários:

Trata-se do encapsulamento, que restringe o acesso ao comportamento interno de um objeto.

Gabarito: Letra A

---

17. (CESPE / ABIN – 2018) Se, em tempo de execução de um sistema, ocorrer associação entre uma entidade e um atributo, então essa associação será considerada um acoplamento dinâmico.

Comentários:

Late Binding (também conhecida como ligação dinâmica) ocorre em tempo de execução. Ademais, uma *Early Binding* (ligação estática) ocorre em tempo de compilação.

Gabarito: Correto

---

18. (CESPE / ABIN – 2018) Considere que, em um sistema de informações, um objeto possua dados de uma pessoa, tais como: nome, endereço, data de aniversário e número do cartão de crédito. Considere, ainda, que esse sistema exponha, de forma pública, informações sobre o nome e a data de aniversário e deixe os dados do cartão de crédito protegidos em formato privado. Nesse caso, o sistema estará usando o recurso de interface.

Comentários:

Na verdade, o recurso é o de modificadores de acesso, ou seja, com o uso do modificador privado, os dados do cartão de crédito estarão visíveis apenas na classe em que foram criados.

Gabarito: Errado

---

19. (CESPE / STM – 2018) Um recurso de grande utilidade nesse tipo de programação consiste na possibilidade de um objeto exercer o comportamento de outro objeto.

Comentários:

Exato, conforme ocorre na herança! A herança facilita o compartilhamento de comportamento comum entre classes. Desse modo, a classe-filha pode exercer comportamentos da classe-pai.

Gabarito: Correto

---

20. (CESPE / STM – 2018) Os atributos de um objeto podem ser expostos tanto por meio de um enlace direto a uma variável interna quanto por meio do retorno de um valor por meio de um método.

Comentários:

É possível que o acesso aos atributos de um objeto seja direto, basta alterar o modificador de acesso. No entanto, não é recomendável que seja dessa forma – o ideal é que os modificadores de acesso não sejam definidos como público.

Gabarito: Correto

---

21. (CESPE / STM – 2018) O tipo de herança mais eficiente e indicado é a herança de implementação, pois possibilita que uma nova classe reutilize a implementação de outra classe sem a necessidade de se recortar e colar o código de forma manual, tornando o código automaticamente disponível, como parte da nova classe.

Comentários:

Herança de implementação (herança de classe) é a herança padrão que conhecemos, em que uma subclasse herda de uma superclasse. Herança de interface (ou de subtipo) vincula uma classe a uma interface. Ademais, ao fazer uma herança de implementação, automaticamente se faz uma herança de interface. Por fim, é mais indicado o uso da herança de interface.

Gabarito: Errado

---

22. (CESPE / STM – 2018) O encapsulamento permite que um programa seja dividido em várias partes menores; contudo, as partes tornam-se dependentes umas das outras em relação à implementação e em relação ao trabalho realizado.

Comentários:

Na verdade, o que o encapsulamento faz é tornar as partes menores do software mais independentes, ou seja, cada parte possui sua própria implementação e realiza suas operações independentemente.

Gabarito: Errado

---

23. (CESPE / STM – 2018) Em orientação a objetos, os membros de dados de uma classe devem ser acessados por um método específico do objeto, e não diretamente.

Comentários:

Essa uma boa prática de orientação a objetos. O acesso direto aos membros de dados (atributos) não é recomendado. É o que ocorre, por exemplo, no encapsulamento. Ele protege os dados de um objeto, impedindo o acesso direto a esses dados.

Gabarito: Correto

---

24. (CESPE / SEDF – 2017) Um objeto define atributos, comportamentos e abstrações comuns compartilhados por um tipo de classe.

Comentários:

Na verdade, uma classe define atributos, comportamentos e abstrações comuns compartilhados por um tipo de objeto.

Gabarito: Errado

---

25. (CESPE / TRE-BA – 2017) Na orientação a objetos, o conceito de polimorfismo é implementado, em algumas linguagens, por meio da técnica de sobrecarga de métodos. Sobre a aplicação desse conceito, é possível afirmar que:

a) um mesmo método pode estar em classes diferentes, com a mesma assinatura, sem que isso prejudique a sua identificação pelo compilador.

b) a identificação da assinatura do método corresponde ao seu nome.

c) uma classe derivada da classe-mãe tem vários métodos com a mesma assinatura, e o compilador realiza o primeiro deles.

d) a classe derivada da classe-mãe herda os métodos da sua classe-mãe na forma como são implementados.

e) cada método tem um nome único na classe derivada da classe-mãe a ser identificado pelo compilador.

Comentários:

Vejam a pegadinha! O enunciado fala sobre o conceito de polimorfismo e diz que ele pode ser implementado por meio da técnica de sobrecarga de métodos. No entanto, ao final ele diz "sobre esse **conceito**" e, não, sobre essa **técnica**. Logo, estamos falando do polimorfismo como um todo e, não, sobre a técnica de implementação de polimorfismo por meio da sobrecarga de dados. *Ok?*

(a) Correto, um mesmo método pode estar em classes diferentes, isto é, sobrescrita de métodos – com exatamente a mesma assinatura e isso não prejudica em nada o compilador; (b) Errado, nós vimos em aula que a assinatura é composta pelo nome e lista de parâmetros (tipo, ordem, quantidade); (c) Errado, não necessariamente tem vários métodos com a mesma assinatura e não existe isso de o compilador realizar o primeiro deles; (d) Errado, se ele herda na forma como são implementados, não temos polimorfismo, temos apenas herança; (e) Errado, métodos podem ter o mesmo nome – isso é polimorfismo.

Gabarito: Letra A

---

26. (CESPE / TRE-BA – 2017) A partir de uma classe derivada de uma superclasse, podem-se invocar métodos que tenham a mesma assinatura, mas comportamentos distintos, ou seja, em que haja alteração do funcionamento interno de um método herdado de um objeto pai. Na orientação a objetos, isso é possível por meio de:

a) polimorfismo.
b) abstração.
c) encapsulamento.
d) namespaces.
e) atributos.

Comentários:

Método com a mesma assinatura do método da classe-pai, mas funcionamento interno diferente. *Qual seria o nome disso?* Fácil, é polimorfismo! Vamos deixar a questão mais difícil: *De que tipo? É Estático ou Dinâmico? É Sobrescrita ou Sobrecarga? É Overriding ou Overloading? Ocorre em tempo de compilação ou de execução? Ou seria uma Redefinição?* A questão trata de polimorfismo dinâmico, porque o método a ser executado será definido em tempo de execução. Está intimamente ligado ao conceito de herança e também é chamado de Sobrescrita, Redefinição ou Overriding.

Gabarito: Letra A

---

27. (CESPE / TCE-PR – 2016) Em se tratando de orientação a objetos, o polimorfismo refere-se:

a) ao reconhecimento do comportamento variado de um método, o que melhora o aproveitamento do código.

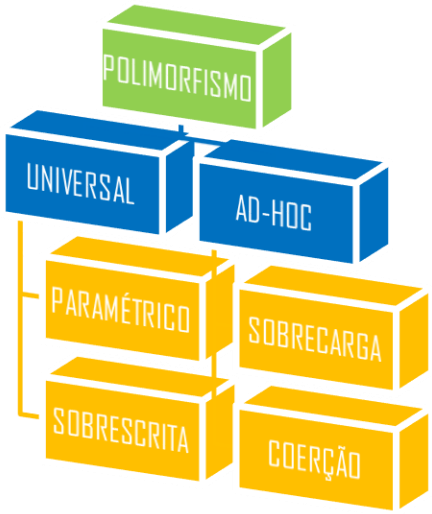
b) à transmissão dos métodos e atributos de uma classe para suas subclasses, quando ad hoc.

c) à variação das saídas de um método com relação às entradas recebidas, caso em que é considerado universal.

d) ao uso que um objeto faz dos recursos de outro objeto.

e) à utilização de métodos específicos para manipular dados com proteção por meio de encapsulamento.

Comentários:



(a) Correto, métodos de mesmo nome com comportamentos variados; (b) Errado, a questão trata de sobrescrita que é polimorfismo universal e, não, ad-hoc; (c) Errado, saída diferente, logo assinatura diferente, portanto é ad-hoc e, não, universal; (d) Errado, isso é um relacionamento de associação e, não, polimorfismo; (e) Errado, não são métodos específicos e o encapsulamento é irrelevante.

Gabarito: Letra A

28.(CESPE / FUB – 2016) Uma das vantagens de se utilizar encapsulamento em orientação a objetos é impedir o acesso direto aos atributos de um objeto.

Comentários:

Perfeito! O encapsulamento realmente protege os dados do objeto, impedindo acesso direto a eles.

Gabarito: Correto

---

29. (CESPE / FUB – 2016) O conjunto de valores das características de determinado objeto é denominado estado.

Comentários:

Perfeito! O estado de um objeto é o conjunto de valores das características (atributos) desse objeto.

Gabarito: Correto

---

30. (CESPE / FUB – 2016) Quando uma classe é subclasse de várias superclasses, mas somente herda características de uma classe, a herança é do tipo simples.

Comentários:

A questão já está conceitualmente errada porque é impossível uma subclasse de várias superclasses herdar apenas de uma de suas superclasses. Você não pode dizer que você é um primata, mas não é um mamífero, porque todos os primatas são mamíferos. Vamos aproveitar para definir alguns conceitos importantes. Primeiro: herdar é diferente de acessar! Se uma classe estende a outra, ela sempre herdará seus métodos/atributos.

Não importa, por exemplo, se eles são privados - a subclasse sempre os herdará, mesmo que não os acesse! Como eu gosto de visualizar isso? Imaginem que um tio-avô distante deixe um cofre entupido de dinheiro para vocês. No entanto, esse cofre é completamente indestrutível e ele não deixou nenhuma senha. Nesse caso, vocês herdaram todo o dinheiro, mas não podem acessá-lo.

Segundo: há que se tomar um certo cuidado com a redação desse tipo de questão. *Por que?* Porque quando a questão afirma que uma classe é subclasse de várias superclasses, ela não disse em nenhum momento que essas superclasses não estavam em uma mesma hierarquia ou disse que a classe era subclasse direta de várias superclasses. *Ok?* Já houve questões cobrando essa pegadinha!

Esse tipo de questão é rodeado por polêmicas. *Por que?* Porque alguns dizem que o lance de herdar/acessar é válido para objetos, mas não para classes. Além disso, algumas especificações de linguagens (Ex: Java) deixam explícito que subclasses não herdam atributos/métodos privados. No entanto, aqui estamos falando do Paradigma Orientado a Objetos.

Gabarito: Errado

---

31. (CESPE / TRE-GO – 2015) Uma classe abstrata possui instâncias diretas, bem como suas classes descendentes.

Comentários:

Classe abstrata não gera instâncias diretas! Item já está errado daí...

Gabarito: Errado

---

32. (CESPE / TJDFT – 2015) Recurso de classes é a propriedade dos objetos que viabiliza a implementação de hierarquia entre objetos.

Comentários:

O que é recurso de classes? Não faço ideia, isso não existe no contexto de orientação a objetos! *E o que é a propriedade dos objetos que viabiliza a implementação de hierarquia entre objetos?* É a herança! O examinador trocou herança por um nome qualquer...

Gabarito: Errado

---

33. (CESPE / TJDFT – 2015) A combinação de dados com o código que os manipula em um único objeto é denominada encapsulamento.

Comentários:

Quando eu li essa questão, eu pensei: "*Einh?*". Achei MUITO mal escrita e, em minha opinião, essa não é uma boa definição de encapsulamento. Encapsulamento é a restrição ao acesso direto a dados de um objeto e uma forma de agrupar dados com os métodos que vão manipular esses dados – e foi isso que a questão quis dizer com essa redação estranha. Ou seja, a combinação de dados com o código que os manipula em um único objeto e esse código que os manipula são os métodos.

Gabarito: Correto

---

34. (CESPE / TCE-RN – 2015) O processo de herança permite a reutilização de código, como também o reaproveitamento de atributos e métodos. Assim, em aplicações que utilizam herança, a obtenção de polimorfismo é uma possibilidade.

Comentários:

Corretíssimo! A primeira parte está perfeita, visto que ela permite a reutilização de código, reaproveitamento de atributos e métodos. E, aplicações que utilizam herança, podem realizar polimorfismo quando em classes diferentes – a isso, designamos sobrescrita de métodos. Então, o polimorfismo dinâmico está intimamente ligado ao conceito de herança.

Gabarito: Correto

---

35. **(CESPE / MEC – 2015)** Objetos são definidos como entidades da modelagem de sistemas que armazenam estados com a utilização de atributos dos próprios objetos, sem interação com outros objetos por meio de mensagens.

Comentários:

A questão vinha bonitinha até dizer que objetos não interagem com outros objetos por meio de mensagens. Eles interagem, sim (e por meio de mensagens)!

Gabarito: Errado

---

36. **(CESPE / INPI – 2013)** Ao se utilizar o encapsulamento, não é necessário saber como ele funciona internamente, apenas como transmite os seus atributos.

Comentários:

'*Transmitir*' foi um verbo pouco adequado utilizado pelo examinador. Na verdade, dizer que o encapsulamento transmite atributos não faz nenhum sentido - encapsulamento é um conceito, ele não transmite nada. Abstraindo isso, o que a questão quis dizer é que, para um método encapsulado, é necessário exibir apenas seus parâmetros e, não, detalhes internos.

Gabarito: Correto

---

37. **(CESPE / TRE-MS – 2013)** Em programação orientada a objetos, a possibilidade de haver funções de mesmo nome, com funcionalidades similares em classes sem nenhuma relação entre elas, denomina-se:

a) encapsulamento.
b) objeto.
c) classe.
d) polimorfismo.
e) relacionamento hierárquico.

Comentários:

Imaginem que nós temos uma classe abstrata: Animal. Essa classe contém um método abstrato chamado Comer( ) que deve ser implementada pelas classes que a estenderem. Vamos supor agora que temos duas classes que a estendem e implementam esse método abstrato: Dinossauro e Minhoca. Ora, temos dois métodos/funções de mesmo nome, com funcionalidades similares em classes sem nenhuma relação uma com a outra = polimorfismo. O que talvez confunda o candidato é: na verdade, elas têm um parentesco uma com a outra, no sentido que ambas são filhas do mesmo pai, mas elas em si não possuem um relacionamento uma com a outra.

Gabarito: Letra D

---

38. (CESPE / MPU – 2013) Se uma subclasse herdar características de duas ou mais superclasses, ocorrerá uma herança múltipla.

Comentários:

Essa questão gerou muita polêmica! A herança múltipla ocorre quando uma subclasse herda características de duas ou mais superclasses que não estejam na mesma hierarquia. *Como assim, professor?* Galera, esse é o grande lance dessa questão! Ora, se A é superclasse de B e B é superclasse de C, não existe herança múltipla, porque C vai herdar características de A e B, que estão na mesma hierarquia. Logo, não necessariamente ocorrerá uma herança múltipla.

Gabarito: Errado

---

39. (CESPE / INPI – 2013) Em uma operação de sobrecarga, uma classe derivada pode redefinir operações de sua classe base.

Comentários:

Uma classe derivada pode redefinir operações de sua classe base, mas esse é o conceito de sobrescrita e, não, sobrecarga (redefinição = sobrescrita).

Gabarito: Errado

---

40. (CESPE / TRE-RJ – 2012) Cada classe pode ter implementações de operação — ou métodos — com denominações únicas. Classes diferentes podem ter métodos com denominações iguais, porém, uma classe não pode ter métodos com denominações iguais e parâmetros diferentes.

Comentários:

Claro que pode! Inclusive, o nome disso é sobrecarga...

Gabarito: Errado

---

41. (CESPE / MPE-PI – 2012) É possível que um mesmo objeto tenha mais de um método com o mesmo nome.

Comentários:

Perfeito! A questão trata de polimorfismo (especificamente da sobrecarga, dado que se trata de um mesmo objeto).

Gabarito: Correto

---

42. (CESPE / TRE-RJ – 2012) As heranças, que são princípios de orientação a objetos, permitem o compartilhamento de atributos e métodos pelas classes e são usadas com o intuito de se reaproveitar código ou comportamento generalizado ou especializar operações ou atributos.

Comentários:

Perfeito! Apesar de soar estranho especialização de atributos, pode-se interpretar como um atributo extra que especializa e diferencia a classe filha.

Gabarito: Correto

---

43. (CESPE / TCE-ES – 2012) A implementação de polimorfismo envolve o mecanismo de redefinição de métodos, assim como o conceito de ligação tardia.

Comentários:

Perfeito! Redefinição ou sobrescrita é um tipo de polimorfismo que envolve o conceito de ligação tardia, que é quando o método a ser invocado é definido em tempo de execução.

Gabarito: Correto

---

44. (CESPE / TRE-RJ – 2012) Polimorfismo consiste em focalizar nos aspectos essenciais inerentes a uma entidade e ignorar propriedades acidentais ou menos importantes. No desenvolvimento de sistemas, isso significa concentrar-se no que um objeto é e no que ele faz antes de se decidir como esse objeto será implementado.

Comentários:

Na verdade, abstração consiste em focalizar nos aspectos essenciais.

Gabarito: Errado

---

45. (CESPE / TRE-RJ – 2012) O polimorfismo de sobrecarga é realizado por meio da aplicação de parâmetros diferentes em operadores distintos com o mesmo nome e semânticas distintas.

Comentários:

Perfeito! Na sobrecarga, temos mesmo nome, mas implementações diferentes (isto é, semânticas diferentes).

Gabarito: Correto

46.(CESPE / IPEA – 2012) A análise orientada a objetos, o projeto orientado a objetos e a programação orientada a objetos compreendem atividades de engenharia de software voltadas à construção de sistemas orientados a objetos. Nesses sistemas, objetos interagem para prover serviços. No nível de programação, as interações ocorrem via interfaces das classes das quais os objetos são instâncias. Essas interfaces contêm membros públicos das classes.

Comentários:

Perfeito! Objetos são instâncias de uma classe que fornecem serviços por meio de métodos públicos contidos dentro de interfaces.

Gabarito: Correto

47.(CESPE / FUB – 2011) Um objeto possui dados internos e métodos que definem, respectivamente, seu estado atual e seu comportamento. Um objeto pode se comunicar com outros objetos passando mensagens.

Comentários:

Perfeito! Objetos se comunicam pela troca de mensagens...

Gabarito: Correto

48.(CESPE / TJ-ES – 2011) Na programação orientada a objetos, o encapsulamento representa a ação em que diversas implementações de uma operação utilizam vários tipos de parâmetros diferentes.

Comentários:

Opa... isso não é encapsulamento – trata-se de polimorfismo!

Gabarito: Errado

49.(CESPE / EBC – 2011) A herança representa uma generalização, dessa forma, por meio de herança é feito o compartilhamento de atributos e operações entre classes, com base em um relacionamento hierárquico.

Comentários:

Perfeito! Por meio de herança é realmente feito o compartilhamento de atributos e operações entre classes em um relacionamento de generalização/especialização.

Gabarito: Correto

---

50. (CESPE / TRE-ES – 2011) Em programação orientada a objetos, a herança serve para criar classes que incorporem propriedades e métodos de outras classes. Assim, é possível construir uma classe a partir de outra sem ter de reescrevê-la.

Comentários:

Perfeito! A herança, de fato, ajuda a criar classes que incorporem propriedades e métodos de outras classes. Nesse caso, a subclasse herda e não precisa reescrevê-la.

Gabarito: Correto

---

51. (CESPE / BRB – 2011) Para que a interface pública de uma classe seja considerada coesa, é necessário que todos os recursos dessa interface estejam relacionados ao conceito que a classe representa.

Comentários:

Primeiro, o que é coesão? Coesão é a divisão de responsabilidades, isto é, uma classe deve ter uma única responsabilidade e realizá-la de maneira satisfatória. Ela não deve assumir responsabilidades que não são suas em hipótese alguma. Dito isso, a questão afirma que para que a interface pública de uma classe seja considerada coesa, é necessário que todos os recursos dessa interface estejam relacionados ao conceito que a classe representa. Ora, é exatamente isso! Se nós tivermos uma classe Produto, a interface dessa classe deve estar relacionada somente a Produto. Dessa forma, não assumindo outras responsabilidades.

Gabarito: Correto

---

52. (CESPE / TRE-ES – 2011) Objetos de software interagem e comunicam-se com os outros por meio de mensagens. Por exemplo, quando o objeto A deseja que o objeto B execute um de seus métodos, envia a este uma mensagem. Algumas vezes, o objeto receptor precisa de mais informação para que saiba exatamente o que deve fazer, de modo que essa informação seja transmitida juntamente com a mensagem por meio de parâmetros.

Comentários:

Objetos se comunicam por meio de mensagens? Sim. *Quando o objeto A deseja que o objeto B execute um de seus métodos, envia a este uma mensagem?* Sim. *Algumas vezes o objeto receptor precisa de algumas informações?* Sim, o nome disso é parâmetro!

Gabarito: Correto

---

53. **(CESPE / MPE-TO – 2011)** Entre os diversos diagramas utilizados em análise e projeto orientados a objetos, o diagrama de casos de uso, por procurar representar todas as possíveis situações de utilização do sistema, é considerado o diagrama responsável por mostrar a estrutura estática do sistema.

Comentários:

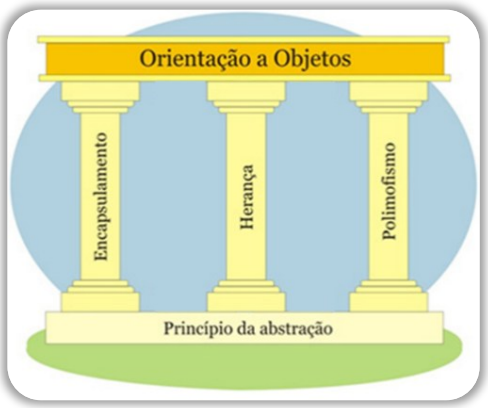
Aqui estamos falando de UML! Diagramas de Casos de Uso são responsáveis por representar a funcionalidade de um sistema, logo é dinâmica e, não, estática.

Gabarito: Errado

---

54. **(CESPE / TRT-RN – 2010)** Programa que utilize uma linguagem orientada a objetos (OO), ao incorporar corretamente ao seu funcionamento conceitos como os de encapsulamento, herança e polimorfismo, beneficia-se das características da OO.

Comentários:



Os pilares derivados da abstração: herança, polimorfismo e encapsulamento.

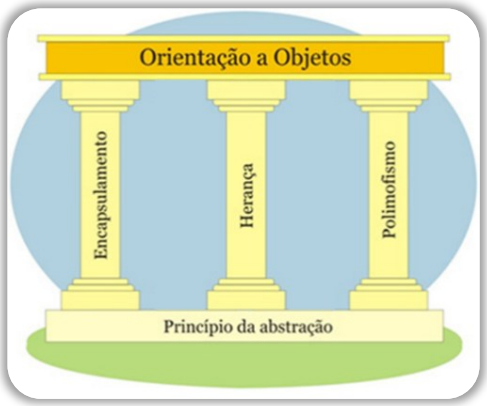
Gabarito: Correto

---

55. **(CESPE / TRT-RN – 2010)** Além dos conceitos de objeto e classe, o paradigma da orientação a objetos envolve os princípios de:

a) abstração, encapsulamento, herança e polimorfismo.
b) abstração, métodos, instâncias e herança.
c) abstração, encapsulamento, generalização e especialização.
d) generalização, especialização, herança e polimorfismo.
e) atributos, métodos, instâncias e mensagens.

Comentários:



Os pilares derivados da abstração são: herança, polimorfismo e encapsulamento.

Gabarito: Letra A

---

56. (CESPE / Banco da Amazônia – 2010) Objetos têm identidade própria. Isso garante que, mesmo tendo os mesmos valores de variáveis e pertencendo à mesma classe, dois objetos sejam considerados diferentes.

Comentários:

Os objetos possuem identidade, isto é, são únicos mesmo que sejam instâncias de uma mesma classe e que tenham os mesmos valores.

Gabarito: Correto

---

57. (CESPE / ABIN – 2010) Um objeto apresenta três características básicas, o estado, a identidade e o comportamento. A parte de dados de um objeto é definida por um conjunto de mensagens, e a porção funcional, por um conjunto de atributos.

Comentários:

A questão inverteu os conceitos: a parte de dados de um objeto é definida por um conjunto de atributos e a parte funcional é definida por um conjunto de operações.

Gabarito: Errado

---

58. (CESPE / ABIN – 2010) Objeto é o agrupamento de classes similares que apresentam os mesmos atributos e operações. Na definição de uma classe, é necessário estabelecer a que objeto ela ocorre como instância.

Comentários:

A questão inverteu os conceitos: classe é o agrupamento de objetos similares que apresentam os mesmos atributos e operações. Na definição de um objeto, é necessário estabelecer a que classe ele ocorre como instância.

Gabarito: Errado

---

59. (CESPE / TCU – 2010) Uma classe pode ser vista como uma descrição generalizada de uma coleção de objetos semelhantes.

Comentários:

Perfeito! Essa é a definição de uma classe: descrição generalizada de uma coleção de objetos semelhantes.

Gabarito: Correto

---

60. (CESPE / MPU – 2010) Uma mensagem enviada a um objeto pode levar à execução de um método que não esteja implementado na classe à qual o objeto pertence.

Comentários:

Perfeito! Para entender, vamos ver um exemplo: imaginem uma classe que possui duas subclasses. A Classe-pai pode ser instanciada, gerando um objeto que contém um método que só é implementado em suas classes-filhas. Em outras palavras, se esse objeto recebe uma mensagem, ela pode levar à execução de um método que não é implementado na classe à qual ele pertence, isto é, esse método é da classe-pai, mas é implementado na classe-filha. Captaram?

Gabarito: Correto

---

61. (CESPE / DETRAN-ES – 2010) Um dos conceitos em programação orientada a objetos é o de abstração, por meio da qual as características do mundo real podem ser modeladas, por exemplo, mediante o agrupamento de objetos e classes.

Comentários:

Perfeito! Abstração modela o mundo real subtraindo algumas características e agrupando em objetos e classes.

Gabarito: Correto

---

62. (CESPE / MPU – 2010) Considerando as características do relacionamento entre uma classe e suas subclasses, é correto afirmar que toda implementação de subclasse é polimórfica.

Comentários:

Na verdade, não é obrigatório fazer sobrescrita ou sobrecarga nos métodos herdados.

Gabarito: Errado

---

63. (CESPE / Banco da Amazônia – 2010) A herança é um conceito implementado por todas as linguagens de programação orientadas a objeto. No entanto, algumas delas somente permitem o uso de herança simples, não sendo possível a criação de classes por meio de herança múltipla.

Comentários:

Perfeito! Um exemplo é a linguagem Java...

Gabarito: Correto

64.(CESPE / TRT-RN – 2010) Uma subclasse, por ser derivada de uma superclasse e possuir todos os atributos da superclasse, além de atributos específicos, é mais especializada que a superclasse da qual foi derivada.

Comentários:

Perfeito! Além dos métodos e atributos da superclasse, ela pode ter seus métodos e atributos que a especializam.

Gabarito: Correto

65.(CESPE / MPU – 2010) Em uma hierarquia de classes na qual exista herança múltipla, não é possível fazer uso do polimorfismo na implementação das classes.

Comentários:

Claro que pode! É possível fazer uso de polimorfismo com herança múltipla, sim...

Gabarito: Errado

---

66. (CESPE / BASA – 2010) Na modelagem de classes, a hierarquia entre elas é representada por meio de um relacionamento chamado generalização.

Comentários:

Perfeito! Na verdade, podemos chamar a herança de generalização/especialização.

Gabarito: Correto

---

67. (CESPE / TRE-BA – 2010) O estado de um objeto é definido pelo conjunto de valores de suas propriedades.

Comentários:

Não tem muito o que comentar aqui! É exatamente isso, o estado é o conjunto de valores das propriedades de um objeto.

Gabarito: Correto

---

68. (CESPE / Banco da Amazônia – 2010) O comportamento de um objeto é definido em sua respectiva classe, por meio da implementação de métodos que são executados quando tal objeto recebe uma mensagem.

Comentários:

Perfeito! O comportamento de um objeto é realmente definido em sua respectiva classe e isso ocorre por meio da implementação de métodos que são executados quando tal objeto recebe uma mensagem.

Gabarito: Correto

---

69. (CESPE / Banco da Amazônia – 2010) A abstração permite, entre outras funcionalidades, identificar e compor objetos complexos e construir estruturas, na forma de classes de objetos, para organizar objetos de diferentes tipos. Porém, conceitos implementados por classes que são construídas com base na abstração não podem ser generalizados nem especializados.

Comentários:

É claro que podem ser generalizados e especializados. Além disso, ela permite compor objetos complexos e estruturas para organizar objetos do seu mesmo tipo e, não, diferentes.

Gabarito: Errado

---

70. (CESPE / TRE-BA – 2010) Em programação orientada a objetos, as propriedades que definem a estrutura e o comportamento de um objeto são especificadas para a classe da qual o objeto é instância e são válidas para todos os objetos dessa classe.

Comentários:

Perfeito! O estado de um objeto é definido pelo conjunto de valores de suas propriedades e pelo comportamento de seus métodos que são definidos a partir do valor de seus atributos.

Gabarito: Correto

---

71. (CESPE / ANAC – 2009) O uso de mais de uma super classe imediata é usualmente denominado herança múltipla; ter somente uma super classe direta é denominado herança simples.

Comentários:

Perfeito! Observem que ele falou mais de uma superclasse imediata, logo é realmente uma herança múltipla – quando há somente uma, é uma herança simples.

Gabarito: Correto

---

72. (CESPE / DETRAN-DF – 2009) Uma classe é uma descrição de um conjunto de objetos que compartilham os mesmos atributos, operações, relacionamentos e semântica.

Comentários:

A classe é a descrição dos atributos e serviços comuns a um grupo de objetos (reais ou abstratos), logo podemos dizer que é um modelo a partir do qual objetos são construídos (atributos, operações, relacionamento, semântica, etc).

Gabarito: Correto

---

73. (CESPE / TRT-BA – 2008) Os objetos permitem encapsular dados e funções, que modelam comportamentos e atributos, respectivamente.

Comentários:

A questão inverteu os conceitos: objetos encapsulam funções (comportamentos) e dados (atributos).

Gabarito: Errado

---

74. (CESPE / MPE-RR – 2008) Na orientação a objetos, um objeto tipicamente possui estado e operações definidas. O estado é representado por atributos, e as operações associadas ao objeto podem fornecer serviços a outros objetos. Os objetos são criados de acordo com classes.

Comentários:

Perfeito! Um objeto possui estado e operações definidas. Além disso, o estado é representado por atributos, e operações fornecem serviços (métodos). Objetos são instâncias de classes, logo são criados de acordo com elas.

Gabarito: Correto

---

75. **(CESPE / PETROBRÁS – 2007)** Em um modelo de análise, as classes de fronteira modelam interações entre o sistema e os atores. Cada classe de fronteira deve estar relacionada a um ou mais atores. Pode-se também ter classes de entidade, as quais tipicamente modelam dados persistentes.

Comentários:

Classes de Fronteira são utilizadas para modelar a interação entre um ator e o sistema. Para cada ator, é identificada pelo menos uma classe de fronteira para permitir sua interação com o sistema. Então, uma classe de fronteira existe para que o sistema se comunique com o mundo exterior, logo elas são altamente dependentes do ambiente.

Classe de Entidade são utilizadas para armazenar a informação que é manipulada ou processada pelo caso de uso, partindo do domínio do negócio. Geralmente, essas classes armazenam informações persistentes. Há várias instâncias ou objetos de uma mesma classe de entidade coexistindo dentro do sistema.

Gabarito: Correto

---

76. **(CESPE / PETROBRÁS – 2007)** Em um modelo de análise, as classes de controle podem encapsular controles relacionados a casos de uso e representar lógicas de negócio que não se relacionem a uma classe de entidade específica.

Comentários:

Classe de Controle são utilizadas para controlar a lógica de execução ou negócio correspondente a cada caso de uso. Servem como uma ponte de comunicação entre objetos de fronteira e objetos de entidade. Não confundam: regras de negócio ficam nas classes de entidades, mas a lógica de negócio fica nas classes de controle.

Gabarito: Correto

---

77. **(CESPE / PETROBRÁS – 2007)** Em um modelo de projeto, para que um subsistema seja coeso, seus conteúdos devem ser fortemente relacionados e, para que ele seja fracamente acoplado, é necessário que se minimizem as dependências entre subsistemas.

Comentários:

Essa questão é uma pegadinha! Ele diz que "seus conteúdos" devem ser fortemente relacionados, portanto estamos falando de dentro de um subsistema (de um objeto, por exemplo). Ele não está falando da relação entre dois subsistemas (se estivesse, estaria tratando de acoplamento e, não, coesão). Logo, dentro de um subsistema, os conteúdos devem ser fortemente relacionados para que um subsistema seja coeso, ou seja, tenha uma responsabilidade única.

Gabarito: Correto

---

78. (CESPE / ANATEL – 2006) Uma classe na análise orientada a objeto representa uma abstração que pode ser mapeada para mais de uma classe no projeto. As classes na análise podem ser fronteiras, controladoras ou entidades. Uma fronteira modela interações entre o sistema e atores, uma entidade modela apenas objetos persistentes e uma controladora só pode controlar interações entre instâncias de uma mesma classe.

Comentários:

De fato, uma classe de análise pode estar mapeada para mais de uma classe de projeto e podem ser de fronteira, controle ou entidade. No entanto, uma classe de controle pode controlar interações entre instâncias de classes diferentes.

Gabarito: Errado

---

79. (CESPE / TSE – 2006) Um modelo de análise é menos abstrato que um de projeto e as classes em um modelo de análise não podem ser conceituais. As classes na análise podem modelar objetos persistentes, mas não transientes.

Comentários:

O Modelo de Análise é mais abstrato que um Modelo de Projetos. Esse segundo já começa a pensar na solução do problema, e o primeiro pensa apenas na modelagem do problema. Ademais, as classes em um Modelo de Análise podem também ser conceituais. De modo geral, as classes de análise modelam objetos persistentes, mas não transientes – basta lembrar que, na análise, estamos apenas tratando do problema e, não, de sua solução.

Gabarito: Errado

---

80. (CESPE / TSE – 2006) Uma importante responsabilidade da análise é definir a arquitetura do sistema, dividindo-o em subsistemas. Um subsistema expõe serviços via interfaces, que devem ser especificadas na análise.

Comentários:

Quem define a Arquitetura do Sistema é o Projeto e, não, a Análise!

Gabarito: Errado

81.(CESPE / TSE – 2006) Uma classe descreve objetos com as mesmas responsabilidades, relacionamentos, operações, atributos e semântica. As instâncias de uma classe têm, portanto, os mesmos valores para os seus atributos.

Comentários:

Valores? Não, os valores podem ser diferentes.

Gabarito: Errado

---

82. (CESPE / SERPRO – 2006) Uma das vantagens dos métodos de análise e projeto orientado a objetos é o aumento do gap conceitual entre os artefatos produzidos nas fases de análise, projeto e implementação.

Comentários:

Opa... é a diminuição do gap conceitual entre os artefatos. Um Gap Conceitual é como um buraco conceitual, uma brecha, uma fresta, um vácuo conceitual. Métodos de Análise e Projeto OO ajudam a preencher e reduzir esse vácuo existente entre as fases de Análise, Projeto e Implementação.

Gabarito: Errado

83.(CESPE / TRE-AL – 2004) Uma hierarquia de classes é um mecanismo por meio do qual as modificações nos níveis inferiores da hierarquia se propagam de imediato para os níveis superiores.

Comentários:

Na verdade, a questão inverteu os conceitos: as modificações nos níveis superiores da hierarquia se propagam de imediato para os níveis inferiores.

Gabarito: Errado

84.(CESPE / TRE-AL – 2004) O polimorfismo ocorre quando uma subclasse herda atributos e operações de classes diferentes.

Comentários:

Na verdade, isso é uma herança múltipla: subclasse que herda atributos e operações de classes diferentes (em hierarquias diferentes).

Gabarito: Errado

---

85. (CESPE / STJ – 2004) Com a análise orientada a objetos, busca-se identificar entidades do domínio do problema e caracterizá-las de acordo com sua importância para o problema. Essa atividade tem consequências nas etapas de projeto de software, uma vez que as entidades identificadas darão sustentação para a definição das classes de objetos a serem implementadas.

Comentários:

Perfeito! Ela busca identificar entidades do domínio do problema e essas entidades ajudam a sustentar a definição das classes de objetos.

Gabarito: Correto

---

86. (CESPE / STJ – 2004) A definição da linguagem de programação a ser usada na implementação tem igual importância e impacto no projeto e na análise orientados a objetos.

Comentários:

Opa... definir a linguagem de programação não afeta em nada a análise, apesar de afetar o projeto.

Gabarito: Errado

QUESTÕES COMENTADAS – FCC

1. **(FCC / METRÔ-SP – 2019)** Considere as seguintes situações:

1. Um grupo foi formado por um conjunto de pessoas que têm vida própria, independente desse grupo.
2. Uma equipe de TI é formada por um conjunto de programadores com dependência de vida dessa equipe.

Na modelagem Orientação a Objetos com UML, essas situações são reconhecidas, respectivamente, como

a) composição e associação todo-parte.
b) composição e dependência funcional.
c) composição todo-parte e dependência.
d) associação todo-parte e composição.
e) associação independente e composição.

Comentários:

1. Também chamada de agregação, em uma associação todo-parte as partes têm existência própria. Em suma, o todo (grupo) independe da parte (pessoas).

2. A composição é um tipo de associação e trata do relacionamento entre um elemento (todo) e outros elementos (as partes). Ela representa um vínculo mais forte entre objetos-todo e objetos-parte. Ou seja, a equipe é o objeto todo e os programadores são as partes. Nesse sentido, há uma dependência entre o objeto todo e o objeto parte pois a equipe só existe por causa dos programadores.

Gabarito: Letra D

---

2. **(FCC / TRF3 – 2019)** O Polimorfismo, um dos Pilares da Programação Orientada a Objetos - POO,

a) ocorre quando uma classe tem um relacionamento do tipo "1 para" com outra classe e isso implica no modo como a definição das classes devem ocorrer nas aplicações.

b) consiste em esconder os atributos da classe de quem for utilizá-la. Isso se deve a: 1 - para quem for usar a classe não a use de forma errada; e 2 - para que implementação seja feita por meio dos métodos get e set.

c) permite que um mesmo método possa ter vários comportamentos e a definição de qual comportamento será executado se dá pelo valor diferente de um de seus atributos.

d) é um conceito que permite que as características bem como as operações, de um modo global, possam ser repassadas para várias funcionalidades da aplicação.

e) permite utilizar atributos e operações diferentes de uma subclasse, acrescentando ou substituindo características herdadas da classe pai.

Comentários:

(a) Errado, não se trata de polimorfismo; (b) Errado, trata-se do Encapsulamento; (c) Correto, trata-se da definição de Polimorfismo; (d) Errado, trata-se da Generalização; (e) Errado, trata-se da Herança.

Gabarito: Letra C

---

3. **(FCC / SANASA Campinas – 2019)** Considere que um Analista de TI sabe que uma classe Pessoa Física e uma classe Pessoa Jurídica possuem o atributo nome como uma informação em comum e que o CPF é um atributo específico para a Pessoa Física e o CNPJ é um atributo específico para Pessoa Jurídica. Então o Analista criou uma outra classe com o atributo nome e seu objetivo é que haja herança deste e, eventualmente, outros métodos e atributos, para as classes filhas, Pessoa Física e Pessoa Jurídica, que já existiam.

   Essa classe criada não é instanciada, apenas fornece um modelo para geração de outras classes, e é denominada:

   a) Subclasse.
   b) Classe construtora.
   c) Classe abstrata.
   d) Classe sobrescrita.
   e) Pacote.

Comentários:

Uma classe abstrata é sempre uma superclasse que não possui instancias, ou seja, não é instanciada. Ademais, ela apenas define um modelo/template para uma funcionalidade e fornece uma implementação incompleta que é compartilhada por um grupo de classes derivadas.

Gabarito: Letra C

---

4. **(FCC / SANASA Campinas – 2019)** Considere:

Os hidrômetros, relógios registradores de consumo de água, têm determinadas características. Em um sistema de computação, para processar os dados que deles provêm deve-se atentar para o fato que eles têm atributos e operações comuns e outros específicos. Usando pilares da orientação a objeto e a capacidade de reuso viabilizada por linguagens desse paradigma, um Analista usou dois conceitos fundamentais sendo um empregado no âmbito da descrição e estruturação das classes de hidrômetros e outro no âmbito da invocação dos métodos com mesma assinatura, todavia levando em consideração o comportamento distinto de operação dos hidrômetros. Tais conceitos são:

a) herança e visibilidade.
b) herança e polimorfismo.
c) composição e agregação.
d) agregação e polimorfismo.
e) visibilidade e composição.

Comentários:

Primeiramente, devemos lembrar que os pilares fundamentais do Paradigma Orientado a Objetos (POO) são: Encapsulamento, Herança e Polimorfismo. Quando se fala no enunciado em "âmbito da descrição e estruturação de classes", estamos falando de herança pois esta organiza as classes em classes-pai e classes-filhas. Ademais, quando se fala em invocação de métodos com mesma assinatura, mas com comportamento distinto, estamos falando de polimorfismo. Mais especificamente – ainda – o polimorfismo por sobrescrita.

Gabarito: Letra B

---

5. **(FCC / TRT-SC – 2013)** Na programação orientada a objetos, as classes podem conter, dentre outros elementos, métodos e atributos. Os métodos:

a) devem receber apenas parâmetros do mesmo tipo.
b) não podem ser sobrecarregados em uma mesma classe.
c) precisam possuir corpo em interfaces e classes abstratas.
d) podem ser sobrescritos em aplicações que possuem relação de herança.
e) definidos como private só podem ser acessados de classes do mesmo pacote.

Comentários:

(a) Errado, podem receber parâmetros de tipos diferentes; (b) Errado, podem ser sobrecarregados em uma mesma classe; (c) Não, não precisam necessariamente de corpo; (d) Correto, a subclasse pode sobrescrever o método da superclasse; (e) Errado, definidos como *private* só podem ser acessados pela própria classe.

Gabarito: Letra D

---

6. **(FCC / TRT-SC – 2013)** Na programação orientada a objetos, as classes podem conter, dentre outros elementos, métodos e atributos. Os métodos:

a) devem receber apenas parâmetros do mesmo tipo.
b) não podem ser sobrecarregados em uma mesma classe.
c) precisam possuir corpo em interfaces e classes abstratas.
d) podem ser sobrescritos em aplicações que possuem relação de herança.
e) definidos como private só podem ser acessados de classes do mesmo pacote.

Comentários:

(a) Errado, podem ser de tipos diferentes; (b) Errado, podem ser sobrecarregados em uma mesma classe; (c) Errado, não precisam possuir corpo em interfaces e classes abstratas; (d) Correto; (e) Errado, só podem ser acessados dentro da própria classe.

Gabarito: Letra D

---

7. **(FCC / AL-RN – 2013)** Um dos conceitos básicos de orientação a objetos é o fato de um objeto, ao tentar acessar as propriedades de outro objeto, deve sempre fazê-lo por uso de métodos do objeto ao qual se deseja atribuir ou requisitar uma informação, mantendo ambos os objetos isolados. A essa propriedade da orientação a objetos se dá o nome de:

a) herança.
b) abstração.
c) polimorfismo.
d) mensagem.
e) encapsulamento.

Comentários:

Manter objetos isolados? Opa... trata-se de uma característica típica de encapsulamento!

Gabarito: Letra E

---

8. **(FCC / TRE-SP – 2012)** Nos conceitos de orientação a objetos, ..I... é uma estrutura composta por ...II... que descrevem suas propriedades e também por ...III.... que moldam seu comportamento. ....IV.... são ....V.... dessa estrutura e só existem em tempo de execução.

Para completar corretamente o texto as lacunas devem ser preenchidas, respectivamente, por

a) objeto, métodos, assinaturas, Classes, cópias.
b) polimorfismo, funções, métodos, Herança, cópias.

c) classe, atributos, operações, Objetos, instâncias.
d) multiplicidade, símbolos, números, Classes, herdeiros.
e) domínio, diagramas, casos de caso, Diagramas de classe, exemplos.

Comentários:

Nos conceitos de orientação a objetos, classe é uma estrutura composta por atributos que descrevem suas propriedades e também por operações (ou serviços) que moldam seu comportamento. Objetos são instâncias dessa estrutura e só existem em tempo de execução.

Gabarito: Letra C

---

9. **(FCC / TJ-RJ – 2012)** No contexto de programação orientada a objetos, considere as afirmativas abaixo.

I. Objetos são instâncias de classes.
II. Herança é uma relação entre objetos.
III. Mensagens são formas de executar métodos.
IV. Classes são apenas agrupamentos de métodos.
V. Ocorre herança múltipla quando mais de um método é herdado.
VI. Herança é uma relação entre classes.

Está correto o que se afirma APENAS em:

a) I, III e IV.
b) I, III e VI.
c) III, IV e VI.
d) II, III e V.
e) II, IV e V.

Comentários:

(I) Correto, eles são instâncias de classes; (II) Errado, trata-se de uma relação entre classes; (III) Correto, elas realmente são formas de executar métodos; (IV) Errado, classes são um conjunto de coisas (ou objetos); (V) Errado, ocorre herança múltipla quando uma classe herda de uma ou mais classes; (VI) Correto, trata-se de uma relação entre classes.

Gabarito: Letra B

---

10. **(FCC / TRF2 – 2012)** Sobre orientação a objetos é correto afirmar:

a) Na hierarquia de classes, se superclasse é uma generalização de subclasses, pode-se inferir que a subclasse é uma especialização de superclasse.

b) Numa árvore genealógica de classes, a classe mais baixa herda os atributos e métodos somente da superclasse no nível imediatamente acima.

c) As variáveis de uma classe só podem ser alteradas por métodos definidos nos seus objetos.

d) O polimorfismo se caracteriza quando, para mensagens distintas, objetos diferentes responderem ou agirem de forma idêntica.

e) Os objetos de uma classe são idênticos no que se refere à sua interface e ao seu estado.

Comentários:

(a) Correto. É uma questão puramente de lógica: se a superclasse é uma generalização de subclasses, uma subclasse é uma especialização de superclasse; (b) Errado. Na verdade, ela herda de todas as classes acima – assim como um neto herda os genes da sua mãe, da sua avó, da sua bisavó, etc; (c) Errado. Variáveis (dependendo de sua visibilidade) podem ser alteradas por métodos em outros objetos que instanciem essa classe; (d) Errado. Objetos idênticos responderem ou agirem de forma diferente; (e) Errado. Objetos de uma classe só serão idênticos em relação ao seu estado, caso tenham exatamente os mesmos valores de atributos – isso não é regra, é exceção.

Gabarito: Letra A

---

11. (FCC / TRE-CE – 2012) Orientação a Objetos é um paradigma de análise, projeto e programação de sistemas de software. A respeito desse paradigma, assinale a afirmativa incorreta.

a) Um objeto pode ser considerado um conjunto de dados.
b) Os objetos possuem identidade, estado e comportamento.
c) Um evento pode existir se não houver um objeto a ele associado.
d) Um objeto pode existir mesmo que não exista nenhum evento associado a ele.
e) A orientação a objetos implementa o conceito de abstração, classe, objeto, encapsulamento, herança e polimorfismo.

Comentários:

(a) Correto; (b) Correto, são os três componentes de um objeto; (c) Errado, um objeto pode existir sem um evento associado, mas um evento só existe se estiver associado a um objeto; (d) Correto, um objeto pode existir sem que haja um evento associado; (e) Correto.

Gabarito: Letra C

---

12. (FCC / TJ-PE – 2012) Sobre orientação a objetos, considere:

I. A relação de herança permite modelar as similaridades inerentes a uma classe e também as diferenças especializadas que distinguem uma classe de outra.

II. Objetos com os mesmos atributos e operações possuem a mesma identidade, podendo ser referenciados por outros objetos.

III. A possibilidade de uma operação ter o mesmo nome, diferentes assinaturas e possivelmente diferentes semânticas dentro de uma mesma classe ou de diferentes classes é chamada de polimorfismo.

Está correto o que se afirma em:

a) I, II e III.
b) I e III, apenas.
c) II e III, apenas.
d) III, apenas.
e) II, apenas.

Comentários:

(I) Correto. A relação de herança realmente permite modelar as similaridades inerentes a uma classe e também as diferenças especializadas que distinguem uma classe de outra; (II) Errado. Até mesmo quando dois ou mais objetos contêm os mesmos atributos e operações, possuem identidades diferentes – cada objeto é único; (III) Correto. *Mesmo nome? Diferentes assinaturas?* Trata-se de uma característica típica do polimorfismo estático.

Gabarito: Errado

---

13. (FCC / TRE-CE – 2012) Sobre orientação a objetos, é INCORRETO afirmar:

a) os conceitos de generalização e especialização da orientação a objetos estão diretamente associados ao conceito de herança.

b) um objeto pode existir mesmo que não exista nenhum evento a ele associado.

c) um construtor visa inicializar os atributos e pode ser executado automaticamente sempre que um novo objeto é criado.

d) polimorfismo é o princípio pelo qual duas ou mais classes derivadas de uma mesma superclasse podem invocar métodos que têm a mesma assinatura e mesmo comportamento.

e) uma classe define o comportamento dos objetos através de seus métodos, e quais estados ele é capaz de manter através de seus atributos.

Comentários:

(a) Correto. Generalização e especialização estão intrinsecamente ligados ao conceito de herança; (b) Correto. Não é obrigatório que tenha um evento associado; (c) Correto. Esse é um dos objetivos de um método construtor; (d) Errado. Trata-se de polimorfismo (especificamente de sobrescrita); (e) Correto. Comportamento por meio de métodos e Estados por meio de atributos

Gabarito: Letra D

---

14. (FCC / TCE-AM – 2012) Sobre a orientação a objeto é correto afirmar:

a) Herança permite o reaproveitamento de atributos e métodos, porém, isso não altera o tempo de desenvolvimento, não diminui o número de linhas de código e não facilita futuras manutenções.

b) Em uma aplicação que utiliza herança múltipla, uma superclasse deve herdar atributos e métodos de diversas subclasses. Todas as linguagens de programação orientadas a objeto permitem herança múltipla.

c) O polimorfismo associado à herança trabalha com a redeclaração de métodos previamente herdados por uma classe. Esses métodos, embora semelhantes, diferem de alguma forma da implementação utilizada na superclasse, sendo necessário, portanto, reimplementá-los na subclasse.

d) A visibilidade protegida é representada pelo símbolo til (~) e significa que somente os objetos da classe detentora do atributo ou método poderão enxergá-lo ou utilizá-lo.

e) Em uma relação de herança é possível criar classes gerais, com características compartilhadas por muitas classes. Essas classes não podem possuir diferenças.

Comentários:

(a) Errado. Como há um reaproveitamento, em geral reduz-se o tempo de desenvolvimento, número de linhas de código – além de facilitar futuras manutenções; (b) Errado. A subclasse herda da superclasse e, não, o contrário! Além disso, nem toda linguagem orientada a objetos permite herança múltipla; (c) Correto. O método é herdado da superclasse e reimplementado na subclasse; (d) Errado. Protegido é (#), ademais a descrição foi de visibilidade privada; (e) Errado. É claro que elas podem possuir diferenças, caso contrário não faria sentido.

Gabarito: Letra C

---

15. (FCC / TST – 2012) Considere que a classe Pessoa possui 3 métodos que podem ser aplicados aos seus objetos: cadastrar, alterar e excluir. Considere que Aluno e Professor são classes derivadas da classe Pessoa e, por isso, herdam os métodos cadastrar, alterar e excluir, mas estes métodos são sobrescritos na classe Aluno e Professor com implementações bastante distintas, em função dos dados associados a cada um deles. O exemplo ilustra o conceito de:

a) hereditariedade.
b) polimorfismo.
c) encapsulamento.
d) abstração.
e) reusabilidade.

Comentários:

Como a questão fala que os métodos são sobrescritos, trata-se de polimorfismo.

Gabarito: Letra B

---

16. (FCC / TRT-AM – 2012) Sobre Programação Orientada a Objetos, analise:

I. A encapsulação garante que apenas as interfaces necessárias para interação com o objeto estejam visíveis, e atributos internos não sejam acessíveis.

II. O polimorfismo garante que objetos possam herdar métodos e atributos de uma superclasse para a geração de uma nova classe.

III. A herança possibilita que distintas operações na mesma classe tenham o mesmo nome, desde que alterada a assinatura.

Está correto o que se afirma em:

a) III, apenas.
b) II, apenas.
c) I, apenas.
d) II e III, apenas.

Comentários:

(I) Correto. A encapsulação realmente garante que apenas as interfaces necessárias para interação com o objeto estejam visíveis, e atributos internos não sejam acessíveis; (II) Errado. A questão descreve herança e, não, polimorfismo; (III) Errado. Isso é polimorfismo e, não, herança.

Gabarito: Letra C

---

17. (FCC / TST – 2012) Na orientação a objetos:

a) a herança permite que os membros de uma classe, chamada de classe-pai, possam ser reaproveitados na definição de outra classe, chamada de classe-filha. Esta classe-filha tem acesso aos membros públicos e protegidos da classe-pai. O polimorfismo, associado à herança, permite que métodos abstratos definidos em uma classe abstrata sejam implementados nas classes-filhas, podendo estes métodos, nas classes-filhas, apresentar comportamentos distintos.

b) atributos e métodos podem ser reaproveitados através da herança, quando uma subclasse herda as características de uma superclasse. Uma subclasse pode ter acesso aos membros de uma superclasse, independente do modificador atribuído. O polimorfismo é um recurso que permite a uma subclasse reimplementar os métodos herdados de uma superclasse, sendo este método abstrato ou não.

c) a herança e o polimorfismo são complementares, ou seja, devem ser aplicados em conjunto. A herança existe a partir de classes abstratas que contêm atributos e métodos abstratos. O polimorfismo obriga que as classes-filhas implementem os métodos e atributos desta classe-pai. O acesso aos atributos da classe-pai independe do modificador utilizado.

d) o conceito de herança estabelece que uma classe possa aproveitar a implementação, definições dos atributos e métodos de uma classe-base. A classe-filha pode ter acesso aos métodos e atributos públicos e protegidos da classe-base. O polimorfismo é aplicado ao caso em que existe a necessidade de implementar métodos sobrecarregados, nos quais a classe-filha necessita implementar dois métodos com o mesmo nome e parâmetros diferentes.

e) o polimorfismo é uma técnica que permite um objeto nascer a partir do uso de sobrecarga de construtores de uma classe, ou seja, o polimorfismo permite que um objeto possa ser instanciado de diferentes maneiras. A herança permite que uma classe sirva de base para que outras classes sejam implementadas. Entretanto, os membros com modificadores públicos da classe-base podem ser acessados pela classe-filha.

Comentários:

(a) Correto. Membros da superclasse são reaproveitas na subclasse. De fato, ela tem acesso aos membros públicos e protegidos da superclasse (privados, não). E o polimorfismo realmente permite que métodos abstratos definidos em uma classe abstrata sejam implementados nas subclasses (eventualmente com comportamentos distintos); (b) Errado. *Independente do modificador atribuído?* Não! Se for *private*, a subclasse herda, mas não acessa; (c) Errado. Não é obrigatório serem utilizados em conjuntos. Classes abstratas não têm necessariamente atributos e métodos abstratos. Não há obrigatoriedade de as classes-filhas implementarem a classe-pai. O acesso aos atributos da classe-pai depende do modificador utilizado; (d) Errado. Na verdade, são métodos sobrescritos e, não, sobrecarregados; (e) Errado. Não vejo relação disso com polimorfismo.

Gabarito: Letra A

---

18. (FCC / TRT11 – 2012) No contexto de Programação Orientada a Objetos (OOP), sobre a relação de agregação e composição, ou relação todo-parte, considere:

I. A relação de agregação expressa o ato ou resultado de formar um objeto usando outros objetos como seus componentes.
II. Na relação de agregação, as partes só existem enquanto o todo existir.
III. Na relação de composição, as partes são independentes da existência do todo.

Está correto o que se afirma em:

a) I, apenas.
b) II, apenas.
c) II e III, apenas.
d) III, apenas.
e) I, II e III.

Comentários:

(I) Correto, as partes têm existência própria; (II) Errado, as partes têm existência própria; (III) Errado, as partes não têm existência própria.

Gabarito: Letra A

---

19. (FCC / INFRAERO – 2011) Sobre orientação a objetos, é correto afirmar:

a) Uma classe é o projeto do objeto. Ela informa à máquina virtual como criar um objeto de um tipo específico. Cada objeto criado a partir da classe terá os mesmos valores para as variáveis de instância da classe.

b) Um relacionamento de herança significa que a superclasse herdará as variáveis de instância e métodos da subclasse.

c) Uma interface é uma classe 100% abstrata, ou seja, uma classe que não pode ser instanciada.

d) Os objetos têm seu estado definido pelos métodos e seu comportamento definido nas variáveis de instância.

e) A principal regra prática do encapsulamento é marcar as variáveis de instância como públicas e fornecer métodos de captura e configuração privados.

Comentários:

(a) Errado. Não se pode afirmar isso! Cada objeto criado a partir da classe terá os mesmos valores para as variáveis de classe, no entanto não podemos afirmar o mesmo para variáveis de instância; (b) Errado. *A superclasse herda?* Não, a subclasse herda; (c) Correto. No entanto, eu discordo da banca! Ela afirma que "uma interface é uma classe 100% abstrata (...)", mas interfaces não são classes. Há diferenças grandes entre classes e interfaces em relação a instanciação, estado, comportamento, herança, variáveis, métodos, etc. No entanto, a banca entendeu de maneira diferente; (d) Errado. Seu comportamento é definido por métodos e seu estado é definido por atributos (Ex: variáveis de instância); (e) Errado. Opa... é o contrário! Variáveis de Instância são privadas e os métodos para acessá-las são públicos.

Gabarito: Letra C

---

20. (FCC / INFRAERO – 2011) Sobre a programação orientada a objetos, analise:

I. Neste tipo de programação, objetos executam ações, mas não suportam propriedades ou atributos.
II. Uma classe especifica o formato geral de seus objetos.
III. As propriedades e ações disponíveis para um objeto não dependem de sua classe.
IV. A tecnologia orientada a objetos permite que classes projetadas adequadamente sejam reutilizáveis em vários projetos.

Está correto o que consta em:

a) II, III e IV, apenas.
b) I e II, apenas.
c) II e IV, apenas.
d) I, II e III, apenas.
e) I, II, III e IV.

Comentários:

(I) Errado, eles suportam propriedades e atributos; (II) Correto; (III) Errado, podemos dizer que é evidente que as propriedades e ações do objeto dependem da própria classe do objeto. No entanto, essas não são as únicas propriedades e ações disponíveis para um objeto. É comum, inclusive, que um objeto tenha disponíveis, além de suas próprias propriedades e ações, as propriedades e ações de outros objetos (claro, dependendo do encapsulamento); (IV) Correto, uma das grandes vantagens na mudança de paradigmas foi que agora temos um outro nível de reusabilidade.

Gabarito: Letra C

---

21. (FCC / TRT-RS – 2011) O aumento da produtividade de desenvolvimento e a capacidade de compartilhar o conhecimento adquirido, representa uma vantagem no uso de projetos orientados a objeto, porque:

a) um objeto pode ser chamado por objetos de classe diferente da sua.
b) os objetos podem ser potencialmente reutilizáveis.
c) as classes podem ser concretas ou abstratas.
d) todo método pode ser derivado naturalmente das operações de sua classe.
e) o encapsulamento impossibilita equívocos de código.

Comentários:

Classes são potencialmente reutilizáveis! Dessa forma, aumenta-se bastante a produtividade de desenvolvimento e a capacidade de compartilhar conhecimento. No entanto, a banca afirmou que objetos são potencialmente reusáveis infelizmente.

Gabarito: Letra B

---

22. (FCC / TRT14 – 2011) Considere:

I. Uma classe define o comportamento dos objetos através de seus métodos, e quais estados ele é capaz de manter através de seus atributos.

II. Na herança cada classe derivada (subclasse) apresenta as características (estrutura e métodos) da classe base (superclasse) e acrescenta a elas o que for definido de particularidade para ela.

III. Polimorfismo é o princípio pelo qual duas ou mais classes derivadas de uma mesma superclasse podem invocar métodos que têm a mesma identificação e mesmo comportamento.

IV. Um objeto é capaz de armazenar estados através de seus atributos e reagir a mensagens enviadas a ele, assim como se relacionar e enviar mensagens a outros objetos.

Na orientação a objetos é correto o que se afirma em:

a) I, II e III, apenas.
b) I, II e IV, apenas.
c) I, III e IV, apenas.
d) II, III e IV, apenas.
e) I, II, III e IV.

Comentários:

(I) Correto. Comportamento por meio de métodos e Estados por meio de atributos; (II) Correto. Excelente definição do conceito de herança; (III) Errado. *Mesma identificação e mesmo comportamento?* Não, polimorfismo trata da mesma identificação e comportamento diferente; (IV) Correto. Atributos armazenam estados e os objetos reagem a mensagens.

Gabarito: Correto

---

23. (FCC / TRT-MT – 2011) Sobre os conceitos de orientação a objetos, considere:

I. Classe encapsula dados para descrever o conteúdo de alguma entidade do mundo real.
II. Objetos são instâncias de uma classe que herdam os atributos e as operações da classe.
III. Superclasse é uma especialização de um conjunto de classes relacionadas a ela.
IV. Operações, métodos ou serviços fornecem representações dos comportamentos de uma classe.

Está completo e correto o que consta em:

a) I, II, III e IV.
b) I, II e IV, apenas.
c) II, III e IV, apenas.
d) I e II, apenas.
e) II e IV, apenas.

Comentários:

Mais uma questão péssima da FCC! Aqui é uma tarefa de adivinhação! Além disso, observem que o item (II) está em todas as alternativas: (I) Correto, apesar de incompleto! Ela encapsula dados e operações; (II) Correto; (III) Errado, trata-se de uma generalização; (IV) Correto. Apesar de a primeira afirmação estar incompleta, não está errada – a banca foi de letra (e).

Gabarito: Letra E

---

24. (FCC / CAIXA – 2011) Na orientação a objetos, é um recurso que serve para inicializar os atributos e é executado automaticamente sempre que um novo objeto é criado:

a) método.
b) polimorfismo.
c) interface.
d) classe.
e) construtor.

Comentários:

O recurso que serve para inicializar os atributos e é executado automaticamente sempre que um novo objeto é criado é também chamado de método construtor.

Gabarito: Letra E

---

25. (FCC / TRE-RN – 2011) Método especial destinado ao preparo de novos objetos durante sua instanciação. Pode ser acionado por meio do operador new, recebendo parâmetros como métodos comuns, o que permite caracterizar os objetos já na instanciação. Trata-se de:

a) operação polimórfica.
b) construtor.
c) atributo.
d) herança polimórfica.
e) herança múltipla.

Comentários:

O método especial destinado ao preparo de novos objetos durante sua instanciação é também chamado de método construtor.

Gabarito: Letra B

---

26. (FCC / CAIXA – 2011) Na programação orientada a objetos, subprogramas (ou subrotinas) são encapsuladas nos próprios objetos e passam a designar-se:

a) atributo.
b) herança.
c) instância.
d) método.
e) encapsulamento.

Comentários:

Subprogramas ou sub-rotinas encapsuladas em objetos são os famosos métodos.

Gabarito: Letra D

---

27. (FCC / CAIXA – 2011) Objetos se comunicam por passagem de mensagem, eliminando áreas de dados compartilhados.

Comentários:

Perfeito! Primeiro, objetos se comunicam por meio de passagem de mensagem. Segundo, eliminam-se áreas de dados compartilhados. *Como assim?* Se forem utilizados corretamente os conceitos de orientação a objetos, elimina-se a necessidade de variáveis de classe (ou globais).

Gabarito: Correto

---

28. (FCC / TCE-PR – 2011) Em relação à Programação Orientada a Objetos, é INCORRETO afirmar:

a) Polimorfismo pode ser entendido como um conceito complementar ao de herança. Assim, no polimorfismo é possível enviar a mesma mensagem a diferentes objetos e cada objeto responder da maneira mais apropriada para sua classe.

b) Uma agregação representa um todo que é composto de várias partes e constitui um relacionamento de contenção; se qualquer uma das partes for destruída, as demais partes também o serão.

c) Interfaces são como as classes abstratas, mas nelas não é possível implementar nenhum método, apenas declarar suas assinaturas; uma classe ao implementar uma interface deverá escrever todos os seus métodos.

d) No contexto da herança, uma instância da subclasse é, também, uma instância da superclasse.

e) A aplicação do polimorfismo utilizando interfaces requer que o método polimórfico seja definido na classe ancestral como abstract para possibilitar sua redefinição nas classes descendentes.

Comentários:

(a) Correto. Esse item é até bonito! Perfeito, perfeito, perfeito; (b) Errado. Na agregação, as partes têm existência própria! A questão trata, na verdade, da composição; (c) Errado. Uma classe abstrata pode também implementar uma interface e não escrever seus métodos – deixando para uma subclasse; (d) Correto. Por exemplo: uma instância de carro é uma instância de veículo; (e) Errado. Questão extremamente confusa. De toda forma, é possível ter uma superclasse concreta que implementa o método de uma interface, mas seja redefinido por outro método em uma classe filha.

Gabarito: Anulada

---

29. (FCC / CAIXA – 2011) Um detalhe importante que deve ser especificado para os atributos e operações das classes é a visibilidade. Desta forma, os símbolos: + (sinal de mais), # (sinal de número), - (sinal de menos) e ~ (til) correspondem respectivamente a:

a) público, pacote, privado e protegido.

b) público, protegido, privado e pacote.
c) privado, protegido, público e pacote.
d) privado, pacote, público e protegido.
e) pacote, protegido, privado e público.

Comentários:

Público (+), Protegido (#), Privado (-), Pacote (~).

Gabarito: Letra B

---

30. (FCC / TRT-MS – 2011) Propriedade pela qual duas ou mais classes derivadas de uma mesma superclasse podem invocar métodos que têm a mesma assinatura mas comportamentos distintos. Trata-se de:

a) polimorfismo.
b) herança múltipla.
c) operação agregada.
d) multiplicidade.
e) visibilidade.

Comentários:

Métodos com a mesma assinatura e comportamentos diferentes? Trata-se de uma característica típica de polimorfismo!

Gabarito: Letra A

---

31. (FCC / TRT14 – 2011) A classe Veiculo contém alguns atributos de interesse da classe Aeronave. Todavia, as aeronaves também demonstram interesse em captar atributos e também operações da classe Elemento Turbinado. O enunciado enfatiza o conceito OO de:

a) polimorfismo.
b) herança múltipla.
c) dependência funcional.
d) realização.
e) encapsulamento.

Comentários:

A FCC disse que o gabarito é a letra (b)! Questão completamente absurda – a herança é um relacionamento é-um! *Aeronave é-um elemento turbinado?* Claro que não! Aeronave *tem-um* elemento turbinado. Logo, discordo do gabarito...

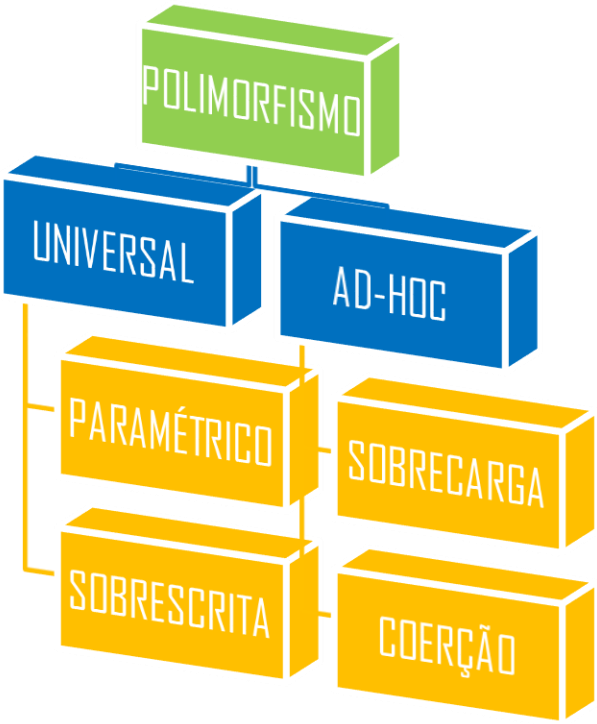
Gabarito: Letra B

---

32. (FCC / TRT-RS – 2011) Na taxonomia utilizada para as formas de polimorfismo são, respectivamente, dois tipos categorizados como universal e dois como Ad Hoc:

a) Paramétrico e Inclusão; Sobrecarga e Coerção.
b) Paramétrico e Coerção; Sobrecarga e Inclusão.
c) Paramétrico e Sobrecarga; Inclusão e Coerção.
d) Sobrecarga e Inclusão; Paramétrico e Coerção.
e) Sobrecarga e Coerção; Paramétrico e Inclusão.

Comentários:



Lembrando que sobrescrita é conhecida também como polimorfismo por inclusão.

Gabarito: Letra A

---

33. (FCC / TRE-CE – 2011) Sobre conceitos em programação orientada a objetos (OOP), analise:

I. No polimorfismo ad-hoc, métodos com o mesmo nome e pertencentes à mesma classe, podem receber argumentos distintos, consequentemente alterando a assinatura do método.

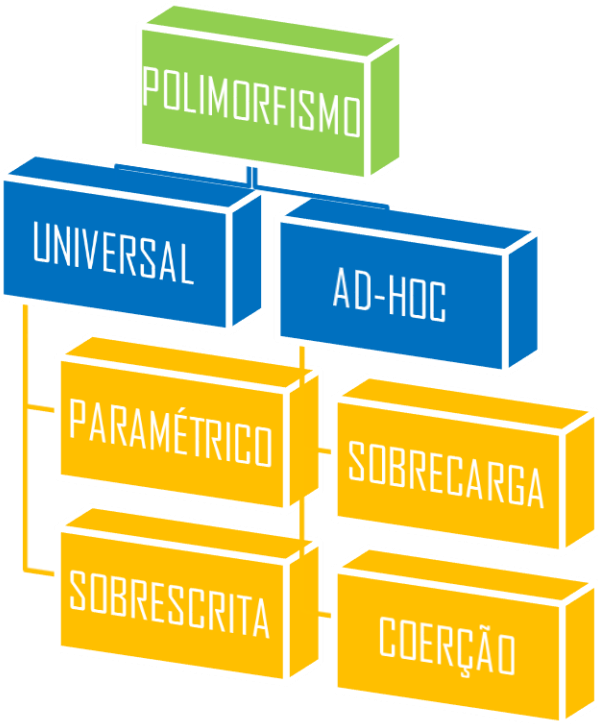
II. No polimorfismo paramétrico é possível determinar o método como atributos de objetos são acessados por outros objetos, protegendo o acesso direto aos mesmos através de operações.

III. Na restrição de multiplicidade é possível determinar o número de atributos e operações que uma classe pode herdar de uma superclasse.

Está correto o que consta em:

a) I, II e III.
b) I, apenas.
c) III, apenas.
d) II e III, apenas.
e) I e II, apenas.

Comentários:



(I) Correto, mesmo nome, mesma classe e argumentos diferentes (logo, assinaturas diferentes) é sobrecarga; (II) Errado, isso é encapsulamento – não tem nenhuma relação com polimorfismo; (III) Errado, a restrição de multiplicidade restringe o número de classes a que outra classe está associada.

Gabarito: Letra B

---

34. (FCC / DPE-SP – 2010) Classes e objetos são dois conceitos-chave da programação orientada a objetos. Com relação a estes conceitos, é correto afirmar que:

a) uma classe é uma descrição de um ou mais objetos por meio de um conjunto uniforme de atributos e serviços. Além disso, pode conter uma descrição de como criar novos objetos na classe.

b) uma classe é capaz de armazenar estados através de seus atributos e reagir a mensagens enviadas a ela, assim como se relacionar e enviar mensagens a outras classes.

c) uma classe é uma abstração de alguma coisa no domínio de um problema ou na sua implementação, refletindo a capacidade de um sistema para manter informações sobre ela, interagir com ela ou ambos.

d) um objeto em uma classe é apenas uma definição, pois a ação só ocorre quando o objeto é invocado através de um método.

e) herança é o mecanismo pelo qual um objeto pode estender outro objeto, aproveitando seus comportamentos e variáveis possíveis.

Comentários:

(a) Correto; Errado, trata-se de objetos e, não, classes; (c) Errado, não são abstrações no domínio da implementação. Ademais, o sistema não interage com classes, mas com objetos; (d) Errado, essa é a definição de um método; (e) Errado, herança é um relacionamento entre classes e, não, objetos.

Gabarito: Letra A

---

35. **(FCC / TCE-SP – 2010)** A descrição de um conjunto de entidades (reais ou abstratas) de um mesmo tipo e com as mesmas características e comportamentos. Trata-se da definição de:

a) String.
b) Método.
c) Conjunto.
d) Classe.
e) Objeto.

Comentários:

A descrição de um conjunto de entidades (reais ou abstratas) de um mesmo tipo e com as mesmas características e comportamentos é também chamada de classe.

Gabarito: Letra D

---

36. **(FCC / DPE-SP – 2010)** A cidade de São Paulo, que possuía uma população de 10.000.000 de habitantes, teve um aumento de mais 2.000.000 de novos habitantes. Na associação da frase

acima aos conceitos da modelagem orientada a objeto, é correto afirmar que São Paulo, população e aumento, referem-se, respectivamente, a:

a) classe, objeto, instância de classe.
b) objeto, atributo, implementação por um método do objeto.
c) classe, objeto, atributo.
d) objeto, instância, operação.
e) classe, objeto, associação pelo método de agregação.

Comentários:

Cidade é uma Classe; *São Paulo* é um Objeto (de Cidade); *População* é um Atributo (do Objeto); *Aumento* é o resultado da implementação de um Método (Ex: setAumento).

Gabarito: Letra B

---

37. (FCC / TRE-RS – 2010) Um objeto é, na orientação a objetos,

a) uma rotina de programação contida em uma classe que pode ser chamada diversas vezes possibilitando assim reuso de código de programação.

b) um conjunto de atributos primitivos tipados contido em uma classe.

c) uma entidade que possui um estado e um conjunto definido de operações definidas para funcionar nesse estado.

d) um elemento de uma classe que representa uma operação (a implementação de uma operação).

e) uma porção de código que resolve um problema muito específico, parte de um problema maior.

Comentários:

Todos os itens viajam bastante, exceto a letra (d)! De fato, um objeto é uma entidade que possui um estado (propriedades) e um conjunto definido de operações (comportamento) definidas para funcionar nesse estado.

Gabarito: Letra C

---

38. (FCC / TRT-PI – 2010) Em relação à orientação a objetos, considere as assertivas abaixo.

I. Um objeto pode ser real ou abstrato. Sendo uma instância de uma classe, possui informações e desempenha ações.

II. Uma classe especifica uma estrutura de dados e os métodos operacionais permissíveis que se aplicam a cada um de seus objetos. Pode ter sua própria estrutura de dados e métodos, bem como pode herdá-la de sua superclasse.

III. Todas as características de uma superclasse são reusáveis por aquelas classes que são seus subtipos. Assim, uma superclasse é um supertipo de uma ou mais classes.

IV. No polimorfismo duas ou mais classes derivadas de uma mesma superclasse podem invocar métodos que têm a mesma identificação, mas comportamentos distintos, especializados para cada classe derivada.

É correto o que se afirma em

a) I, II, III, apenas.
b) I, II e IV, apenas.
c) I, III e IV, apenas.
d) II, III e IV, apenas.
e) I, II, III e IV.

Comentários:

(I) Correto. É estranho falar em *objeto abstrato*, mas vamos ignorar (porque é FCC!). Ele, de fato, possui informações (atributos) e desempenha ações (operações); (II) Correto. São atributos e métodos de instância! Pode herdá-los por herança; (III) Correto. Atributos Private são herdados, sim. No entanto, não são acessíveis porque não são visíveis, logo não podem ser reutilizados! Como a questão disse "Todas as características...", está incorreta! Se ela tivesse dito apenas "As características...", estaria correta. No entanto, a nossa banca insiste em afirmar que o gabarito é verdadeiro; (IV) Correto. No polimorfismo, duas ou mais classes derivadas de uma mesma superclasse realmente podem invocar métodos que têm a mesma identificação, mas comportamentos distintos, especializados para cada classe derivada.

Gabarito: Letra E

---

39. (FCC / SEFAZ-SP – 2010) Os valores das propriedades de um objeto em um determinado instante, que podem mudar ao longo do tempo, representam:

a) a instância de uma classe.
b) a identidade de um objeto.
c) o estado de um objeto.
d) o comportamento de um objeto.

e) as operações de uma classe.

Comentários:

Valores de propriedades representam estados de um objeto.

Gabarito: Letra C

---

40. (FCC / TCM-PA – 2010) Não possui instâncias diretas, mas apenas classes descendentes:

a) a classe concreta.
b) o objeto.
c) a classe abstrata.
d) o caso de uso de inclusão.
e) o pacote.

Comentários:

Classes Abstratas são aquelas que não podem ser instanciadas, apenas estendidas. Em outras palavras, elas não possuem instâncias diretas, apenas classes descendentes.

Gabarito: Letra C

---

41. (FCC / TRT-PR – 2010) Uma técnica que consiste em separar aspectos externos dos internos da implementação de um objeto, isto é, determinados detalhes ficam ocultos aos demais objetos e dizem respeito apenas ao próprio objeto. Trata-se de:

a) polimorfismo.
b) generalização.
c) encapsulamento.
d) herança.
e) visibilidade.

Comentários:

Separar aspectos externos dos internos? Trata-se de uma característica típica de encapsulamento.

Gabarito: Letra C

---

42. (FCC / MPE-RN – 2010) Uma operação pode ter implementações diferentes em diversos pontos da hierarquia de classes, desde que mantenham a mesma assinatura. Na orientação a objetos, este é o conceito que embasa:

a) a multiplicidade.
b) o encapsulamento.
c) o protótipo.
d) o polimorfismo.
e) o estereótipo.

Comentários:

Implementações diferentes? Mesma assinatura? Trata-se de uma característica típica do polimorfismo dinâmico.

Gabarito: Letra D

---

43. (FCC / Sergipe Gás SA – 2010) "É o mecanismo pelo qual uma classe pode estender outra classe, aproveitando seus comportamentos e variáveis possíveis." Na programação orientada a objetos esta afirmação refere-se aos conceitos essenciais de:

a) herança, métodos e atributos.
b) subclasse, instância e associação.
c) subclasse, encapsulamento e abstração.
d) herança, abstração e associação.
e) encapsulamento, polimorfismo e interface.

Comentários:

Uma classe estende outra classe [herança], aproveitando seus comportamentos [métodos] e variáveis possíveis [atributos].

Gabarito: Letra A

---

44. (FCC / TJ-SE – 2009) Na programação orientada a objetos, são características dos objetos:

a) As classes, os métodos e as mensagens.
b) A identidade, os atributos e as operações.
c) O encapsulamento, a herança e o polimorfismo.
d) A instanciação, a generalização e a especialização.
e) A classificação, a composição e a decomposição.

Comentários:

Objetos possuem três características: identidade, propriedades (atributos ou estados) e comportamentos (métodos ou operações).

Gabarito: Letra B

---

45. (FCC / SEFAZ-SP – 2009) Uma classe é uma abstração que ajuda a lidar com a complexidade e um bom exemplo de abstração é:

a) um aluno e as disciplinas que está cursando.
b) um professor e os cursos nos quais ministra aulas.
c) um funcionário e o departamento em que trabalha.
d) uma pessoa e o número do seu CPF na Receita Federal.
e) uma casa e a empresa que a projetou e construiu.

Comentários:

(a) *Aluno* seria uma classe; e *Disciplinas* seria uma classe na forma de uma lista; (b) *Professor* seria uma classe; e *Cursos* seria uma classe na forma de uma lista; (c) *Funcionário* seria uma classe; e *departamento* seria um atributo de *Departamento*; (d) *Pessoa* seria uma classe; e *Número do CPF* seria um atributo dessa classe; (e) *Casa* seria uma classe; e *Empresa* seria também uma classe.

Péssima questão da FCC! Tivemos que abstrair bastante aqui... agora percebam que a questão pede um bom exemplo de abstração de classe. A Letra (d) é a única que apresenta uma classe e um atributo, no entanto não foi isso que a questão pediu. Enfim, essa questão não faz sentido, mas a quarta opção é a única que se distingue e, por isso, foi dada como certa.

Gabarito: Letra D

---

46. (FCC / SEFAZ-SP – 2009) Na orientação a objetos, ao nível de classe, são definidos os:

a) atributos e os valores dos atributos.
b) atributos e a invocação das operações.
c) atributos e os métodos.
d) métodos e os valores dos atributos.
e) métodos e a invocação das operações.

Comentários:

Na orientação a objetos, ao nível de classe, são definidos atributos e métodos.

Gabarito: Letra C

---

47. (FCC / SEFAZ-SP – 2009) O método utilizado para inicializar objetos de uma classe quando estes são criados é denominado:

a) void.

b) interface.
c) agregação.
d) composição.
e) construtor.

Comentários:

O método construtor é chamado assim que uma nova instância de uma classe é criada. Em geral, ele é responsável por alocar recursos necessários ao funcionamento do objeto além da definição inicial das variáveis de estado (atributos).

Gabarito: Letra E

48.(FCC / TJ-PI – 2009) Na programação orientada a objetos, é o princípio que oferece a capacidade de um método poder ser implementado de diferentes formas, ou mesmo de realizar coisas diferentes, ou seja, um único serviço pode oferecer variações, conforme se aplique a diferentes subclasses de uma superclasse. No contexto, o termo método é:

a) o mecanismo pelo qual um objeto utiliza os recursos de outro.
b) uma instância de uma classe.
c) o elemento que define as habilidades do objeto.
d) uma chamada a um objeto para invocar uma classe.
e) um objeto capaz de armazenar estados através de seus atributos.

Comentários:

Um método é um elemento que define as habilidades/comportamentos do objeto.

Gabarito: Letra C

---

49. (FCC / PGE-RJ – 2009) Sobre orientação a objetos, considere:

I. Os valores dos atributos são definidos no nível de classe.
II. Os métodos são definidos no nível de objeto.
III. A invocação de uma operação é definida no nível de objeto.

Está correto o que se afirma em:

a) II e III, apenas.
b) I, II e III.
c) III, apenas.
d) I e II, apenas.
e) I e III, apenas.

Comentários:

(I) Errado. Em geral, são definidos no nível de objeto! No entanto, atributos estáticos são definidos no nível de classe, logo discordo da FCC; (II) Errado. Métodos são definidos no nível de classe, sem dúvida; (III) Correto. Em geral, são definidos no nível de objeto! No entanto, métodos estáticos são definidos no nível de classe, logo também discordo da FCC. Só para relembrá-los:

- ***Atributo estático:*** *são compartilhados por todos os objetos da classe;*
- ***Atributo de instância****: só pode ser acessado por objetos instanciados;*
- ***Método estático:*** *são compartilhados por todos os objetos da classe;*
- ***Método de instância:*** *só pode ser acessado por objetos instanciados.*

Gabarito: Letra C

---

50. (FCC / TJ-PA – 2009) A especificação de uma comunicação entre objetos, que contém informações relacionadas ao que se espera resultar dessa atividade, é:

a) uma restrição.
b) uma mensagem.
c) uma operação.
d) um processo oculto.
e) um diálogo.

Comentários:

A comunicação entre objetos é feita por meio de mensagens...

Gabarito: Letra B

---

51. (FCC / TRE-PI – 2009) A afirmação de que o estado de um objeto não deve ser acessado diretamente, mas sim por meio de métodos de acesso, está associada ao conceito de encapsulamento.

Comentários:

Perfeito! Métodos são públicos e atributos são privados...

Gabarito: Correto

---

52. (FCC / SEFAZ-SP – 2009) Sobre a visibilidade dos métodos na orientação a objetos considere:

I. Os métodos públicos de uma classe definem a interface da classe.

II. Os métodos privativos de uma classe não fazem parte da interface da classe.
III. O nome dos métodos é a informação reconhecida como a assinatura dos métodos.

Está correto o que consta APENAS em:

a) I e II.
b) I e III.
c) II e III.
d) II.
e) I.

Comentários:

(I) Correto. Tucker afirma: *"Os métodos públicos de uma classe definem a interface da classe com o mundo externo. A maioria dos métodos de uma classe são públicos."*; (II) Correto. Métodos privados não fazem parte da interface da classe, visto que eles não podem ser acessados, *então para que serviriam?* (III) Errado. A assinatura é composta por nome do método e parâmetros (envolve quantidade, tipo e ordem de parâmetros).

Gabarito: Letra A

---

53. **(FCC / MPE-SE – 2009)** "A utilização de um sistema orientado a objetos não deve depender de sua implementação interna, mas de sua interface." Esta afirmação remete ao conceito de:

a) herança múltipla.
b) herança polimórfica.
c) prototipação.
d) encapsulamento.
e) especialização.

Comentários:

Implementação interna? Interface? Trata-se de uma característica típica de encapsulamento.

Gabarito: Letra D

---

54. **(FCC / TRT-CE – 2009)** Considere: A classe Pedido contém um método chamado obter Produtos() que retorna uma lista de produtos pertencentes a um determinado pedido. O código que usa esta classe desconhece completamente como esta lista de produtos é montada. Tudo que interessa é a lista de produtos que o método retorna. Na essência, o texto explica um dos fundamentos das linguagens OO que é:

a) polimorfismo.

b) encapsulamento.
c) dependência.
d) herança múltipla.
e) estereotipagem.

Comentários:

Desconhece completamente como a lista é mondada? Interessa apenas o retorno do método? Trata-se de uma característica típica de encapsulamento!

Gabarito: Letra B

---

55. (FCC / MPE-SE – 2009) *"...distintas implementações de uma operação de classe e que, no entanto, o nome e os parâmetros dessa operação sejam os mesmos"*. Trata-se de:

a) objeto persistente.
b) enumeração.
c) polimorfismo.
d) subclasse.
e) pseudo-estado.

Comentários:

Distintas implementações de uma operação? Trata-se de uma característica típica de polimorfismo!

Gabarito: Letra C

---

56. (FCC / TJ-PI – 2009) Na programação orientada a objetos, é o princípio que oferece a capacidade de um método poder ser implementado de diferentes formas, ou mesmo de realizar coisas diferentes, ou seja, um único serviço pode oferecer variações, conforme se aplique a diferentes subclasses de uma superclasse. O texto acima trata do Princípio de:

a) Polimorfismo.
b) Reutilização.
c) Abstração.
d) Herança.
e) Encapsulamento.

Comentários:

Poder ser implementado de diferentes formas? Realizar coisas diferentes? Trata-se de uma característica típica do polimorfismo.

Gabarito: Letra A

---

57. (FCC / TRT-MA – 2009) Um analista desenvolveu métodos de impressão de dados com a mesma assinatura para três classes de impressoras (jato de tinta, laser e matricial) derivadas de uma mesma superclasse impressora. Tal prática:

a) aplica o conceito de herança múltipla.
b) aplica o conceito de polimorfismo.
c) constitui-se em ferimento à regra de herança.
d) visa ao aumento da coesão entre os atributos da superclasse.
e) não é recomendada na orientação a objetos.

Comentários:

Métodos de impressão de dados com a mesma assinatura para três classes de impressoras derivadas de uma mesma superclasse impressora, trata-se do conceito de polimorfismo.

Gabarito: Letra B

---

58. (FCC / PGE-RJ – 2009) Um comando "abrir" ao provocar diferentes ações em objetos distintos, por exemplo: em uma caixa, porta ou janela, representa figurativamente na orientação a objetos o princípio denominado:

a) persistência.
b) polimorfismo.
c) abstração.
d) agregação.
e) herança.

Comentários:

Notem que se trata de uma mesma mensagem "abrir" provocando diferentes ações (ou implementações) em objetos distintos. Só pode ser polimorfismo...

Gabarito: Letra B

---

59. (FCC / SEFAZ-SP – 2009) Compartilhamento de atributos e operações genéricas entre diversas classes descendentes de uma classe ancestral remete ao conceito de:

a) cardinalidade.
b) encapsulamento.
c) herança.
d) agregação.

e) multiplicidade.

Comentários:

Compartilhamento entre classes descendentes? Trata-se de uma característica típica da herança!

Gabarito: Letra C

---

60. (FCC / PGE-RJ – 2009) O conceito de Herança, na orientação a objetos, está especificamente associado ao significado de:

a) cardinalidade.
b) generalização.
c) multiplicidade.
d) encapsulamento.
e) composição.

Comentários:

Herança é sinônimo de generalização/especialização.

Gabarito: Letra B

---

61. (FCC / TCE-AL – 2008) Considere: *Casas ABC Ltda., Empresa e Nome da Empresa.* Na orientação a objetos, os itens acima representam, respectivamente,

a) atributo, classe e objeto.
b) classe, atributo e objeto.
c) classe, objeto e atributo.
d) objeto, atributo e classe.
e) objeto, classe e atributo.

Comentários:

Vamos pensar no que faz mais sentido: *Empresa* é uma classe; *Casas ABC Ltda* é um objeto; e *Nome da Empresa* é um atributo. Vamos abstrair um pouco: existe uma classe (Empresa) que contém um objeto (Casas ABC Ltda.), que é uma instância específica dessa classe e esse objeto contém uma propriedade (Nome da Empresa).

Gabarito: Letra E

---

62. (FCC / TRT18 – 2008) São dois tipos de relacionamento todo-parte:

a) agregação e composição.
b) generalização e composição.
c) generalização e especialização.
d) composição e dependência.
e) especialização e agregação.

Comentários:

Agregação (todo independe da parte) e Composição (todo depende da parte).

Gabarito: Letra A

---

63. (FCC / TCE-AL – 2008) Os conceitos de generalização e especialização da orientação a objetos estão diretamente relacionados ao conceito de:

a) Agregação.
b) Associação.
c) Encapsulamento.
d) Polimorfismo.
e) Herança.

Comentários:

Herança é sinônimo de generalização/especialização.

Gabarito: Letra E

---

64. (FCC / TRF4 – 2007) A proteção de atributos e operações das classes, fazendo com que estas se comuniquem com o meio externo por meio de suas interfaces, define o conceito de:

a) polimorfismo.
b) encapsulamento.
c) herança.
d) agregação.
e) especialização.

Comentários:

Proteção de atributos e operações? Trata-se de uma característica típica de encapsulamento.

Gabarito: Letra B

QUESTÕES COMENTADAS – FGV

1. **(FGV / AL-SC – 2024)** Considerando o paradigma da orientação a objetos, identifique os quatro pilares mestres que norteiam o fundamento da tecnologia.

a) Classe, Abstração, Coesão e Encapsulamento.
b) Variáveis, Classe, Objeto e Método.
c) Agregação, Acoplamento, Colaboração e Especificação.
d) Classe, Objeto, Atributo e Método
e) Acoplamento, Coesão, Colaboração e Encapsulamento.

Comentários:

Questão bizarra! Os pilares da orientação a objetos são: Encapsulamento, Abstração, Herança e Polimorfismo. A questão considerou como correta: Classe, Objeto, Atributo e Método.

Gabarito: Letra D

---

2. **(FGV / SEFAZ-MG – 2023)** Os padrões de projeto de software (design patterns) tiram proveito máximo dos pilares da orientação a objetos. Usemos como exemplo o padrão Abstract Factory, que é um padrão de projeto criacional que permite produzir famílias de objetos relacionados sem especificar suas classes concretas. O principal pilar da orientação a objetos usado nesse design pattern é:

a) o polimorfismo.
b) a herança.
c) o encapsulamento.
d) a abstração.
e) a sublevação.

Comentários:

O principal pilar da orientação a objetos usado no padrão Abstract Factory é a abstração. O padrão Abstract Factory abstrai a criação de objetos relacionados, permitindo que eles sejam criados sem especificar suas classes concretas. Isso permite que novos objetos sejam facilmente incorporados ao sistema sem muita alteração no código existente.

Gabarito: Letra D

---

3. **(FGV / SEAD-AP – 2022)** No contexto da orientação por objetos, o princípio pelo qual duas ou mais classes derivadas de uma mesma classe podem incorporar métodos que têm a mesma assinatura original, porém com comportamentos especializados, é conhecido como:

a) abstração.
b) encapsulamento.
c) herança.
d) interfaceamento.
e) polimorfismo.

Comentários:

Quando duas ou mais classes derivadas de uma mesma classe podem incorporar métodos que têm a mesma assinatura original, porém com comportamentos especializados, temos um exemplo de **polimorfismo** – um dos princípios da orientação a objetos.

Gabarito: Letra E

---

4. **(FGV / TCE-TO – 2022)** O sistema de controle automotivo SisAut foi modelado orientado a objetos. O SisAut possui a classe Veículo, que compartilha seu código-fonte com suas subclasses: Carro e Moto. As subclasses Carro e Moto aproveitam os métodos e atributos da classe Veículo.

   Em orientação a objeto, o mecanismo pelo qual uma classe pode estender outra classe ou ser estendida por outra classe é:

   a) interface;
   b) herança;
   c) pacotes;
   d) agregação;
   e) encapsulamento.

Comentários:

O mecanismo pelo qual uma classe pode estender outra classe ou ser estendida por outra classe é chamado de Herança – um dos princípios da orientação a objetos.

Gabarito: Letra B

---

5. **(FGV / IBGE – 2016)** Em Orientação a Objetos, para que uma subclasse de uma classe possa ter seu próprio comportamento, e mesmo assim compartilhar algumas das funcionalidades da classe pai, deve-se implementar:

   a) generalização;
   b) agregação;
   c) abstração;
   d) composição;

e) polimorfismo.

Comentários:

Quando uma subclasse tem seu comportamento especializado, apesar de compartilhar funcionalidades com a superclasse estendendo o comportamento original, temos um caso de polimorfismo.

Gabarito: Letra E

---

6. **(FGV / CODEBA – 2016)** Durante a fase de análise de um sistema que está sendo desenvolvido sob o paradigma de orientação a objetos, o analista Pedro quer representar, em um diagrama de classes, que uma turma é formada por alunos. Os alunos, porém, também podem ser considerados individualmente no sistema, independente da turma.

   Para representar a relação entre alunos e turma, Pedro deve utilizar:

   a) composição.
   b) agregação.
   c) herança.
   d) encapsulamento.
   e) atributo classe.

Comentários:

Eu gosto dessa questão! Note que uma turma é formada por alunos, mas alunos podem existir individualmente no sistema. *Vamos pensar em termos de cardinalidade?* O relacionamento Turma-Aluno é (1:N)(0:N)*. Logo, ao analisar a cardinalidade mínima do relacionamento, podemos ver que se trata de um relacionamento opcional (não obrigatório), dado que um aluno pode existir sem que exista uma turma. Para representar esse tipo de relacionamento, utiliza-se a Agregação.

** A questão não deixa explícito, mas pode-se inferir que um aluno pode se matricular em várias turmas.*

Gabarito: Letra B

---

7. **(FGV / TCE-SE – 2015)** Em POO (Programação Orientada a Objetos), dizer que a classe A estende a classe B é o mesmo que dizer que:

   a) a classe B é subclasse de A;
   b) a classe A é superclasse de B;
   c) a classe A é derivada de B;
   d) a classe B é derivada de A;
   e) as classes A e B são irmãs.

Comentários:

Se A estende B, A é derivada de B.

Gabarito: Letra C

---

8. **(FGV / TCE-SE – 2015)** Em POO (programação orientada a objetos), dizer que a classe A é superclasse de B é o mesmo que dizer que:

a) A é derivada de B;
b) A estende B;
c) B é derivada de A;
d) B implementa A;
e) A implementa B.

Comentários:

Essa ficou fácil! Se A é superclasse de B, B é derivada de A.

Gabarito: Letra C

---

9. **(FGV / PROCEMPA – 2014)** Definir a responsabilidade de cada classe é um aspecto muito importante que deve ser observado durante a modelagem de um projeto de sistema de software. Em relação aos princípios essenciais de boas práticas de modelagem orientada a objeto assinale a afirmativa correta.

a) O encapsulamento e a adoção de interfaces permitem diminuir o acoplamento e a coesão entre classes.

b) Cada classe deve assumir uma única responsabilidade bem definida como meio de aumentar a coesão e assegurar o encapsulamento.

c) As responsabilidades de uma classe devem estar relacionadas entre si para facilitar o entendimento e aumentar as chances de sua reutilização.

d) Para promover a flexibilidade e a facilidade de manutenção a longo prazo, as classes devem ser fortemente acopladas e encapsuladas.

e) Minimizar o acoplamento implica em classes com alta coesão.

Comentários:

(a) Errado, permitem aumentar a coesão; (b) Errado, a responsabilidade única de uma classe não assegura o encapsulamento; (c) Correto. Classes que possuem responsabilidades bem definidas e se relacionam entre si facilitam o entendimento e aumentam as chances de sua reutilização; (d) Errado, devem ser fracamente acopladas; (e) Errado, em geral a redução do acoplamento aumenta a coesão, mas isso não é uma regra.

Gabarito: Letra C

---

10. (FGV / TJ-GO – 2014) Analise as afirmativas a seguir, no contexto das linguagens de programação orientadas a objetos:

I. A herança múltipla é a possibilidade de uma classe estender uma ou mais classes simultaneamente.

II. A herança múltipla é a possibilidade de uma classe implementar uma ou mais inferfaces simultaneamente.

III. A herança múltipla é a possibilidade de, numa dada classe, coexistirem métodos homônimos com múltiplas assinaturas, desde que distintas.

É verdadeiro somente o que se afirma em:

a) I;
b) II;
c) III;
d) I e II;
e) II e III.

Comentários:

(I) Errado, é a possibilidade de uma subclasse estender duas ou mais superclasses imediatas simultaneamente; (II) Errado, isso é possível, mas não é herança múltipla; (III) Errado, isso é polimorfismo, não é herança múltipla.

A banca considerou o primeiro item como correto, mas eu divirjo e – para mim – a questão não possui resposta e deveria ter sido anulada.

Gabarito: Letra A

---

11. (FGV / TJ-AM – 2013) No que diz respeito à programação orientada a objetos, um recurso refere-se ao poder que os objetos de classes distintas têm de invocar um mesmo método e obter comportamento diferente. Esse recurso é conhecido por:

a) polimorfsimo.
b) encapsulamento.
c) abstração.
d) herança.
e) coesão.

Comentários:

A capacidade de objetos de classes distintas invocarem um mesmo método e obterem comportamentos diferentes é chamado de polimorfismo.

Gabarito: Letra A

---

12. **(FGV / FIOCRUZ – 2010)** Num banco de dados orientado a objetos, a informação é armazenada na forma de objetos. Este tipo de banco de dados possui três bases principais. Assinale a alternativa que atende ao paradigma de Orientação a Objetos.

a) Herança; isomorfismo; multilateralidade.
b) Herança; polimorfismo; encapsulamento.
c) Identidade; isomorfismo; armazenamento.
d) Geomorfismo; identidade; herança.
e) Identidade; classificação; multilateralidade.

Comentários:

O Paradigma Orientado a Objetos (POO) se apoia sobre os pilares da herança, polimorfismo e encapsulamento (há autores que inserem outros pilares, mas esses três necessariamente existirão).

Gabarito: Letra B

---

13. **(FGV / SEAD-AP - 2010)** Em conformidade com a metodologia orientada a objetos, com a finalidade de evitar que partes de um programa se tornem tão independentes que uma pequena alteração tenha grandes efeitos em cascata, é aplicado um recurso que separa os aspectos externos e acessíveis de um objeto dos detalhes internos de implementação.

Esse recurso utiliza um princípio da Orientação a Objetos que propõe ocultar determinados elementos de uma classe das demais classes. O objetivo ao colocar uma proteção ao redor é prevenir contra os efeitos colaterais indesejados ao ter essas propriedades modificadas de forma inesperada.

Este recurso é conhecido por:

a) coesão.

b) acoplamento.
c) polimorfismo.
d) modularidade.
e) encapsulamento.

Comentários:

O recurso que permite separar aspectos externos e acessíveis dos detalhes internos de implementação, ocultando os últimos e expondo apenas os primeiros para prevenir efeitos colaterais indesejados é o encapsulamento.

Gabarito: Letra E

---

14. **(FGV / MEC - 2009)** Na Análise Orientada a Objetos, o princípio pelo qual duas ou mais classes derivadas de uma mesma superclasse podem invocar métodos que têm a mesma identificação (assinatura) mas comportamentos distintos, especializados para cada classe derivada, usando para tanto uma referência a um objeto do tipo da superclasse é denominado:

a) encapsulamento.
b) independência.
c) modularidade.
d) polimorfismo
e) herança.

Comentários:

Classes com mesma identificação, mas comportamentos distintos especializados para cada classe derivada (subclasse) é um caso típico de polimorfismo.

Gabarito: Letra D

QUESTÕES COMENTADAS – DIVERSAS BANCAS

1. **(CESGRANRIO / IPEA – 2024)** Em um diagrama de classes criado por uma equipe, há uma classe P que possui um relacionamento de associação com a classe Q. Qual situação, em código, representa, de maneira adequada, tal ideia de relacionamento entre essas classes?

   a) A classe P possui um método que instancia um objeto da classe Q.
   b) A classe P possui um método que recebe como parâmetro um objeto da classe Q.
   c) A classe P possui uma propriedade de objeto da classe Q.
   d) A classe P herda da classe Q.
   e) A classe Q herda da classe P.

Comentários:

(a) Errado. Um método que instancia um objeto da classe Q sugere uma dependência temporária ou local, não uma associação direta e contínua entre as classes P e Q;

(b) Errado. Um método que recebe um objeto da classe Q como parâmetro indica uma dependência temporária no contexto desse método, não uma associação direta e contínua;

(c) Correto. Uma propriedade de objeto da classe Q dentro da classe P representa uma associação, pois indica que a classe P tem um relacionamento direto e contínuo com um objeto da classe Q;

(d) Errado. Herança indica uma relação "é-um" (P é um Q), não uma associação entre duas classes distintas;

(e) Errado. Herança na direção oposta também não representa uma associação, mas sim uma relação de especialização.

Gabarito: Letra C

---

2. **(CESGRANRIO / TRANSPETRO – 2023)** Um desenvolvedor pretende criar três métodos com o mesmo nome em uma mesma classe, mas com parâmetros diferentes. Para essa tarefa, utilizará uma técnica que faz parte do paradigma de orientação a objetos. Qual o nome da técnica a ser usada pelo desenvolvedor?

   a) Herança
   b) Interface
   c) Instanciação
   d) Sobrecarga
   e) Sobrescrita

Comentários:

(a) Errado. Herança é o mecanismo pelo qual uma classe pode herdar propriedades e métodos de outra classe, mas não se refere a criar métodos com o mesmo nome na mesma classe;

(b) Errado. Interface define um contrato que outras classes podem implementar, mas não está diretamente relacionada com a criação de métodos com o mesmo nome na mesma classe;

(c) Errado. Instanciação refere-se ao processo de criar uma instância (objeto) de uma classe, não à criação de métodos com o mesmo nome na mesma classe;

(d) Correto. Sobrecarga (Overloading) é a técnica que permite criar vários métodos com o mesmo nome na mesma classe, mas com diferentes parâmetros;

(e) Errado. Sobrescrita (Overriding) refere-se a redefinir um método de uma classe base em uma classe derivada, mas não à criação de métodos com o mesmo nome na mesma classe.

Gabarito: Letra D

---

3. **(VUNESP / TCM-SP - 2023)** Considere que, na modelagem de dados utilizando a orientação a objetos, há três classes: a classe Filme e as classes Suspense e Comédia, ambas herdando as características da classe Filme. Portanto, está-se fazendo uma aplicação do conceito de herança, segundo o qual:

a) a classe Filme assume o papel de uma subclasse, e as classes Suspense e Comédia assumem o papel de superclasses.

b) a classe Filme só pode ter atributos do tipo literal, enquanto que as classes Suspense e Comédia só podem conter atributos do tipo numérico.

c) as classes Suspense e Comédia herdam atributos e métodos da classe Filme, podendo, por exemplo, acrescentar atributos peculiares à classe Filme.

d) as classes Suspense e Comédia assumem exatamente os mesmos métodos e atributos da classe Filme, sem poder alterar qualquer um deles.

e) as quantidades de objetos oriundos das classes Suspense e Comédia devem ser exatamente as mesmas.

Comentários:

(a) Errado. A classe Filme é a superclasse, e as classes Suspense e Comédia são as subclasses; (b) Errado. A classe Filme pode ter atributos de qualquer tipo, não apenas do tipo literal; (c) Correto. a

classe Filme é a superclasse, enquanto que as classes Suspense e Comédia são as subclasses. Isso significa que as classes Suspense e Comédia herdam todos os atributos e métodos da classe Filme, e podem, ainda, acrescentar atributos peculiares à classe Filme. Assim, por exemplo, uma classe Suspense pode ter um atributo "suspense", enquanto que uma classe Comédia pode ter um atributo "humor". No entanto, ambas as classes herdarão todos os atributos e métodos da classe Filme, como "título", "ano", "diretor", "elenco", etc; (d) Errado. As classes Suspense e Comédia podem alterar os métodos e atributos da classe Filme, desde que não alterem a assinatura dos métodos; (e) Errado. As quantidades de objetos oriundos das classes Suspense e Comédia não precisam ser exatamente as mesmas.

Gabarito: Letra C

---

4. **(ACCESS / Câmara de Mangaratiba-RJ – 2020)** No que diz respeito à Orientação a Objetos, dois princípios são caracterizados a seguir:

   I. faz com que detalhes internos do funcionamento dos métodos de uma classe permaneçam ocultos para os objetos. O conhecimento a respeito da implementação interna da classe é desnecessário do ponto de vista do objeto, uma vez que isso passa a ser responsabilidade dos métodos internos da classe.

   II. indica a capacidade de abstrair várias implementações diferentes em uma única interface. As classes derivadas de uma única classe base são capazes de invocar os métodos que, embora apresentem a mesma assinatura, comportam-se de maneira diferente para cada uma das classes derivadas.

   Os princípios caracterizados em I e II são respectivamente denominados

   a) encapsulamento e coesão.
   b) acoplamento e coesão.
   c) encapsulamento e acoplamento.
   d) acoplamento e polimorfismo.
   e) encapsulamento e polimorfismo.

Comentários:

(I) Pessoal, trata-se do princípio do encapsulamento, que consiste na separação de aspectos internos e externos de um objeto. Ademais, o encapsulamento protege os dados de um objeto, impedindo o acesso direto a eles; (II) Trata-se do princípio do polimorfismo, que busca obter um comportamento específico através de uma mesma interface da classe.

Gabarito: Letra E

---

5. **(IBFC / TRE-PA – 2020)** Assinale a alternativa que identifica incorretamente um conceito básico das linguagens orientadas a objetos.

a) interface
b) herança
c) encapsulamento
d) subprogramação

Comentários:

Desses conceitos apresentados, apenas o de subprogramação não está relacionado a linguagens orientadas a objetos. Na verdade, esse conceito está ligado a linguagens com paradigma procedimental.

Gabarito: Letra D

---

6. **(IBFC / Prefeitura de Cruzeiro do Sul - AC – 2019)** Um dos conceitos do paradigma orientado a objetos consiste na alteração do funcionamento interno de um método herdado de um objeto pai. Assinale a alternativa correta que apresenta este conceito.

a) Polimorfismo
b) Pai e filho
c) Encapsulamento
d) Abstração

Comentários:

O polimorfismo está associado ao conceito de herança, logo trata-se da capacidade de abstrair várias implementações diferentes em uma única interface, ou seja, é justamente o que diz o enunciado a alteração do funcionamento de um método que foi herdado.

Gabarito: Letra A

---

7. **(CONSULPLAN/ Prefeitura de Suzano - SP – 2019)** Na orientação a objetos, todo objeto está relacionado a uma classe que o representa e que serve como forma de modelo. O objeto terá atributos e métodos definidos na classe. Polimorfismo se refere a:

a) Classes com vários métodos com o mesmo nome, mas com parâmetros diferentes.
b) Mecanismo que permite definir uma nova classe (subclasse) a partir de uma já existente (superclasse).
c) Classes com vários tipos diferentes de objetos instanciados; porém, cada um com sua própria variável.

d) Superclasses e classes relacionadas, com interfaces idênticas; porém, com implementações diferentes.

Comentários:

(a) A alternativa retrata o polimorfismo estático, porém o item foi considerado errado pela banca. Discordo do gabarito; (b) Errado, na verdade, trata-se do princípio da herança; (c) Errado, trata-se do conceito de objeto no contexto de orientação a objetos; (d) Correto, a definição refere-se ao princípio do polimorfismo.

Gabarito: Letra D

---

8. **(AOCP / IBGE – 2019)** As classes, bem como os seus objetos, contêm atributos e métodos que estão intimamente relacionados. Os objetos podem se comunicar entre si, mas eles, em geral, não sabem como outros objetos são implementados, uma vez que os detalhes de implementação permanecem ocultos dentro dos próprios objetos. Assinale a alternativa que apresenta corretamente o nome desse ocultamento de informações, crucial na boa prática da engenharia de software.

a) Herança.
b) Recursão.
c) Instanciação.
d) Polimorfismo.
e) Encapsulamento.

Comentários:

O princípio que protege os dados de um objeto, ocultando informações e impedindo o acesso direto a eles é o encapsulamento.

Gabarito: Letra E

---

9. **(QUADRIX / Prefeitura de Jataí - GO – 2019)** Na análise orientada a objetos, o diagrama que descreve os tipos de objetos e seus relacionamentos, descreve a estrutura estática de um sistema, isto é, descreve como o sistema é estruturado, e não como ele se comporta, é o diagrama de:

a) classe.
b) pacotes.
c) colaboração.
d) estados.
e) atividades.

Comentários:

Aqui estamos falando de UML. Existem diversos diagramas de UML, por exemplo a versão 2.3 da UML fornece 13 diagramas diferentes. O diagrama de classe fornece uma visão estática ou estrutural sistema, ou seja, ele não mostra a natureza dinâmica das comunicações entre os objetos. Ademais, os diagramas de classe também exibem relações entre classes.

Gabarito: Letra A

---

10. (QUADRIX / Prefeitura de Jataí - GO – 2019) No que se refere aos conceitos de herança múltipla, julgue os itens subsequentes.

I. Com a implementação da herança múltipla, é possível simplificar os programas e proporcionar soluções para resolver problemas difíceis.
II. Ocorre herança múltipla quando uma classe herda de mais de uma classe, ou seja, existem múltiplas classes-base (pais) para a classe derivada (filha).
III. Na herança múltipla, uma classe herda apenas a estrutura, e não o comportamento de mais de uma classe-base.

a) Apenas o item I está certo.
b) Apenas o item II está certo.
c) Apenas os itens I e II estão certos.
d) Apenas os itens I e III estão certos.
e) Apenas os itens II e III estão certos.

Comentários:

(I) Correto, porém essa alternativa é discutível. A herança múltipla pode tanto simplificar como pode causar problemas de ambiguidade. Fato é que algumas linguagens não implementam a mesma, como por exemplo Java e C#; (II) Correto, é a definição de herança múltipla; (III) Errado, ela irá herdar a estrutura e os comportamentos de mais de uma classe-base.

Gabarito: Letra C

---

11. (QUADRIX / CRO-GO – 2019) Alguns conceitos como herança, polimorfismo e encapsulamento são abordados no desenvolvimento de sistemas orientados a objetos.

Comentários:

Perfeito! Encapsulamento, herança e polimorfismo são os pilares fundamentais do Paradigma Orientado a Objetos (POO).

Gabarito: Correto

---

12. (COVEST-COPSET / UFPE – 2019) No contexto dos principais elementos conceituais do modelo de objetos, assinale a alternativa que define corretamente o conceito de encapsulamento.

a) É a capacidade de uma abstração referenciar outro tipo de dados abstrato, reutilizando comportamento e estabelecendo uma hierarquia de tipos que permite o reuso.

b) É o processo de compartimentalizar os elementos de uma abstração que constituem sua estrutura e comportamento, servindo para separar a interface contratual de uma abstração e sua implementação.

c) Refere-se à capacidade de salvar atributos de um objeto em um meio permanente de persistência, tais como bancos de dados ou arquivos, possibilitando que objetos possam reutilizar o seu estado em diferentes execuções de um mesmo programa.

d) Consiste em dividir um programa em módulos que podem ser compilados separadamente e ter conexões com outros módulos.

e) Denota as maneiras pelas quais um objeto pode agir e reagir, constituindo toda a visão externa estática e dinâmica da abstração.

Comentários:

Encapsulamento é a técnica utilizada para esconder detalhes internos de funcionamento de uma classe. Em suma, há o encapsulamento de uma classe com sua estrutura, seus métodos e seus atributos.

Gabarito: Letra B

---

13. (CESGRANRIO / UNIRIO – 2019) Em orientação a objetos, uma classe abstrata é uma classe que:

a) possui apenas métodos estáticos.
b) não pode ser instanciada.
c) não possui métodos.
d) não possui variáveis de instância.
e) não pode ter subclasses.

Comentários:

Uma classe abstrata é definida para representar entidades e conceitos abstratos, que é sempre uma superclasse e não possui instâncias. Logo, uma classe abstrata não pode ser instanciada.

Gabarito: Letra B

---

14. (IF-PE / IF-PE – 2019) Sobre o uso de interfaces em orientação a objetos, podemos afirmar que:

I. evita que alterações de código em determinados componentes do sistema sejam refletidas por todo o sistema.
II. representa um contrato entre componentes do sistema.
III. permite a utilização das implementações das classes concretas ao invés da utilização das classes abstratas.

Está(ão) CORRETA(S), apenas, a(s) proposição(ões)

a) I e II.
b) I.
c) II e III.
d) I e III.
e) II.

Comentários:

(I) Correto, pois uma interface é utilizada para reduzir o acoplamento; (II) Correto, uma interface representa um contrato sem implementação entre dois ou mais objetos; (III) Errado, em uma interface os métodos são todos abstratos;

Gabarito: Letra A

---

15. (IF-PE / IF-PE – 2019) Estrutura que contém a representação de dados e rotinas que processam esses dados, assim como representa um conjunto de objetos similares. A definição apresentada é sobre:

a) Objeto.
b) Classe.
c) Atributo.
d) Método.
e) Construtor.

Comentários:

Classes são agrupamentos de coisas. Além disso, uma classe é a descrição dos atributos e serviços comuns a um grupo de objetos. Em suma, é um modelo a partir do qual os objetos são construídos.

Gabarito: Letra B

---

16. **(IF-PE / IF-PE – 2019)** Marque a alternativa que representa a definição de herança, em orientação a objetos.

a) Mecanismo que indica que o acesso aos dados (atributos) dos objetos só deve ocorrer pelos métodos do próprio objeto.
b) Mecanismo que permite o reaproveitamento de comportamentos e dados de outras classes do sistema.
c) Mecanismo que permite que métodos, com o mesmo nome, possam ser reimplementados dentro da própria classe.
d) Representação do quanto uma classe depende de outra classe do sistema.
e) Representação das características essenciais de um objeto e que o diferencia de outros objetos do sistema.

Comentários:

(a) Errado, trata-se do encapsulamento; (b) Correto, essa é a definição de herança; (c) Errado, trata-se do polimorfismo; (d) Errado, trata-se do acoplamento; (e) Errado, trata-se da abstração.

Gabarito: Letra B

---

17. **(VUNESP / Câmara de Piracicaba - SP – 2019)** Considerando a orientação a objetos, assinale a alternativa que define corretamente o que é polimorfismo.

a) O armazenamento de objetos de forma permanente, para posterior recuperação em novas execuções do programa.
b) O fato de uma subclasse herdar o comportamento de sua classe mãe.
c) Uma solução reusável para um problema comum.
d) Uma unidade coesa de funcionalidades que podem ser desenvolvidas e entregues independentemente para compor uma unidade maior.
e) Diferentes objetos podem responder à mesma mensagem de maneiras diferentes, possibilitando que objetos interajam uns com os outros sem conhecer seus tipos exatos.

Comentários:

(a) Errado, não está relacionado a polimorfismo; (b) Errado, retrata a herança; (c) Errado, retrata um Padrão de Design para software; (d) Errado, não se trata de polimorfismo; (e) Correto, retrata o polimorfismo.

Gabarito: Letra E

---

18. **(VUNESP / Câmara de Piracicaba - SP – 2019)** No contexto da orientação a objetos, existe uma medida de quanto dois itens, tais como classes ou métodos, estão inter-relacionados. Esta

medida costuma ser classificada como forte, quando um item depende da forma como o outro foi implementado, ou fraca, quando um item depende do outro, mas não de seus detalhes de implementação. Essa medida é conhecida como:

a) coesão.
b) exceção.
c) sobrecarga.
d) interface.
e) acoplamento.

Comentários:

À medida que trata do nível de dependência entre módulos de um software é o acoplamento.

Gabarito: Letra E

---

19. (VUNESP / UFABC – 2019) Considerando a orientação a objetos, é correto afirmar que:

a) uma mensagem é composta exclusivamente pelo objeto que deve receber tal mensagem.
b) os métodos de uma classe são acionados periodicamente, por decurso de tempo.
c) uma classe deve possuir número de atributos maior do que o seu número de métodos.
d) cada objeto possui um limite máximo de número de mensagens que pode enviar.
e) na herança múltipla, uma subclasse pode ter associada uma ou mais superclasses.

Comentários:

(a) Errado, uma mensagem é composta pelo objeto, a quem a mensagem é endereçada, pelo nome do método ou serviço que se deseja executar e pelos parâmetros necessários ao método (se existirem); (b) Errado, métodos são acionados quando são chamados; (c) Errado, não existe essa limitação; (d) Errado, essa limitação também não existe; (e) Correto, quando uma subclasse herda de duas ou mais superclasses, trata-se de herança múltipla.

Gabarito: Letra E

---

20. (UFMG / UFMG – 2019) Uma classe abstrata A contém o método abstrato acao(). A classe B herda da classe A e não implementa o método acao(). Neste contexto, assinale a alternativa CORRETA.

a) A chamada do método acao() de um objeto da classe B chamará a implementação existente na classe A.
b) A classe B não pode sobrecarregar o método acao().
c) A implementação do método abstrato acao() é obrigatória na classe B para que ela compile.
d) A classe B compila sem erros.

Comentários:

Uma classe abstrata é sempre uma superclasse que não possui instâncias. Além disso, classes derivadas completam a funcionalidade da classe abstrata adicionando comportamentos específicos. Por fim, uma classe que herda de uma superclasse abstrata também é abstrata ou então deve obrigatoriamente implementar os métodos abstratos da classe-pai.

Gabarito: Letra C

---

21. (UFMG / UFMG – 2019) Os quatro pilares do paradigma de Orientação a Objetos são:

a) Sequenciamento, Procedimentos, Bibliotecas e Herança.
b) Herança, Polimorfismo, Classes e Objetos.
c) Classes, Atributos, Métodos e Abstração.
d) Abstração, Encapsulamento, Herança e Polimorfismo.

Comentários:

Os pilares fundamentais do POO são Encapsulamento, Herança, Polimorfismo e Abstração, sendo que este último pode ser tratado também como um princípio para alguns autores (princípio da abstração).

Gabarito: Letra D

---

22. (IDECAN / IF-PB – 2019) Sobre os conceitos de Orientação a Objetos, identifique com "V" caso verdadeiro ou "F" caso falso as assertivas a seguir.

( ) A Sobrescrita permite que, em uma mesma classe, tenhamos vários métodos com o mesmo nome, mas com a assinatura diferente.
( ) Objetos são instâncias de uma classe que possui os atributos e as operações definidos na classe.
( ) Superclasse é uma especialização de um conjunto de classes através de herança.
( ) A Sobrecarga possibilita que o mesmo nome possa ser utilizado em diferentes métodos em uma mesma classe, desde que, por exemplo, as quantidades de parâmetros sejam diferentes.
( ) Classe encapsula dados para descrever o conteúdo de alguma entidade do mundo real.

a) V, F, V, F, F
b) V, V, V, F, V
c) F, V, F, V, V
d) F, V, V, V, V
e) V, V, F, F, F

Comentários:

(F) Errado, na verdade, na sobrescrita - que está associada ao conceito de herança - a subclasse deve manter a mesma assinatura da superclasse; (V) Correto, objetos são instâncias de classes, além disso, os componentes de um objeto são: identidade, estado (propriedades) e comportamento (operações); (F) Errado, superclasse é uma generalização e subclasse é uma especialização; (V) Correto, trata-se do polimorfismo por sobrecarga ou overloading, nesse caso, o nome de método deve ser igual e os parâmetros devem ser diferentes; (V) Correto, uma classe é um agrupamento de objetos, é a descrição dos atributos e serviços comuns a um grupo de objetos (reais ou abstratos) .

Gabarito: Letra C

---

23. (CCV-UFC / UFC – 2019) Um dos recursos existentes na orientação a objetos é denominado polimorfismo. Com relação a esse recurso, é correto afirmar:

a) O polimorfismo refere-se à característica de uma classe poder herdar os atributos e métodos de outra classe, tornando-se uma classe mais especializada.

b) Com o polimorfismo, é possível a criação de métodos onde os parâmetros são sempre passados por cópia, independentemente se são tipos primitivos ou objetos.

c) Com o polimorfismo é possível que uma variável de um tipo mais genérico (abstrato) referencie um objeto de um tipo mais específico na sua hierarquia de classes.

d) O polimorfismo restringe que os tipos das variáveis que armazenam um determinado objeto sejam explicitamente do mesmo tipo do objeto, não permitindo generalizações.

e) Refere-se à propriedade de somente tornar visível as informações importantes para o contexto da aplicação, enquanto as demais ficam disponíveis somente no escopo da classe.

Comentários:

(a) Errado, na verdade, trata-se da herança; (b) Errado, quando se fala que os parâmetros são passados por cópia, na verdade, isso se trata de um método de passagem de parâmetros; (c) Correto, trata-se do polimorfismo por sobrecarga. Além disso, não há nenhum problema em se atribuir um objeto da classe-filha (mais específica) à referência de uma classe pai (mais genérica); (d) Errado, é permitido o uso de generalização no polimorfismo; (e) Errado, trata-se do encapsulamento.

Gabarito: Letra C

---

24. (CS-UFG / IF Goiano – 2019) Programação orientada a objetos está baseada no que é conhecido por orientação a objetos. Nesse contexto,

a) a delegação e a herança são recursos para reutilizar comportamento.
b) a serialização é necessária no processo de especialização.
c) a generalização depende de composição.
d) o polimorfismo depende de sobrecarga (overloading).

Comentários:

(a) Correto, tanto herança como a delegação reutilizam comportamentos. No entanto, a delegação não é utilizada por muitas das linguagens de programação; (b) Errado, não existe essa exigência; (c) Errado, não há essa dependência; (d) Errado, o polimorfismo também pode ocorrer por overriding (sobrescrita).

Gabarito: Letra A

---

25. (COSEAC / UFF – 2019) Na orientação objeto existe basicamente três modos de visibilidade. São eles:

a) herança, polimorfismo e agregação.
b) especialização, agregação e associação.
c) composição, realização e agregação.
d) realização, composição e associação.
e) público, protegido e privado.

Comentários:

Os modos de visibilidade são também conhecidos como especificadores ou modificadores de acesso, eles são utilizados para privar o acesso direto a atributos e obrigar o usuário a fazer tal acesso por meio de métodos públicos. Ademais, os principais modificadores de acesso em orientação a objetos são públicos, protegidos e privados.

Gabarito: Letra E

---

26. (COSEAC / UFF – 2019) Em relação à orientação objetos, avalie se são verdadeiras (V) ou falsas (F) as afirmativas a seguir:

I Um método pode receber ou não parâmetros e pode retornar valores.
II Uma classe sempre deve possuir atributos e métodos.
III O polimorfismo trabalha com a redeclaração de métodos previamente herdados por uma classe.

a) V, F e V.
b) F, V e V.

c) V, F e F.
d) F, F e V.
e) V, V e V.

Comentários:

(I) Correto, realmente um método pode receber ou não parâmetros, da mesma forma pode retornar valores ; (II) Errado, uma classe não necessariamente deve possuir métodos e atributos; (III) Correto, o polimorfismo consiste em alterar alguma ação herdada de uma superclasse.

Gabarito: Letra A

---

27. (IF-PA / IF-PA – 2019) Quanto aos conceitos do paradigma da orientação a objetos, é CORRETO afirmar:

a) por meio do conceito de Polimorfismo, é possível a definição de vários métodos ou funções com o mesmo nome, porém com diferentes assinaturas. Essa característica do conceito de Polimorfismo é denominada de Delegação.

b) por meio do conceito de Herança, uma subclasse é capaz de reutilizar os métodos e atributos de uma superclasse, desde que esses métodos e atributos estejam encapsulados, ou seja, suas visibilidades estejam como "private".

c) uma classe definida como Abstrata, é uma classe que define os seus atributos e métodos para que sejam herdados por uma outra classe que irá implementar os seus métodos. Em uma classe Abstrata não é possível a implementação dos seus métodos, somente os seus cabeçalhos.

d) em uma Interface definimos comportamentos (métodos) sem os implementar. Por meio da Interface podemos definir o que um objeto obrigatoriamente deve fazer e não como ele faz.

e) uma classe que implementa uma classe Abstrata, deverá obrigatoriamente redefinir os métodos e atributos que herdou. A classe que implementa a classe Abstrata não pode definir seus próprios atributos.

Comentários:

(a) Errado, trata-se do polimorfismo por sobrescrita ou overriding; (b) Errado, se a visibilidade estiver como private, os métodos e atributos não poderão ser acessados; (c) Errado, uma classe abstrata pode conter métodos concretos, que devem ser implementados; (d) Correto, pois uma interface não lida com o estado interno de um objeto, ela apenas define o que um objeto pode fazer, sem dizer como ele faz; (e) Errado, ela pode definir seus próprios atributos.

Gabarito: Letra D

---

28. (QUADRIX / CRQ 4ª Região-SP – 2018) Com o princípio da "herança", um objeto faz reuso de código, possibilitando a redução de esforços no desenvolvimento de sistemas pelo reaproveitamento de códigos herdados de outros objetos ou classes.

Comentários:

Perfeito! Em suma, a herança facilita o compartilhamento de comportamento comum entre classes, permitindo que classes compartilhem atributos e métodos.

Gabarito: Correto

---

29. (QUADRIX / CRQ 4ª Região-SP – 2018) Graças ao encapsulamento, os atributos de um objeto podem ser protegidos, permitindo o acesso a eles somente a partir de métodos específicos e autorizados.

Comentários:

Perfeito! O encapsulamento é uma forma de restringir o acesso ao comportamento interno de um objeto.

Gabarito: Correto

---

30. (QUADRIX / CRQ 4ª Região-SP – 2018) Polimorfismo em POO é a combinação de atributos e métodos internos a uma classe, de forma a deixar visível apenas o que é necessário para a comunicação entre dois objetos.

Comentários:

Na verdade, este é o conceito de encapsulamento. O encapsulamento é uma forma de restringir o acesso ao comportamento interno de um objeto.

Gabarito: Errado

---

31. (QUADRIX / CRQ 4ª Região-SP – 2018) No conceito de orientação a objetos, a herança é a capacidade de um novo objeto tomar atributos e operações de um objeto ou classe já existente.

Comentários:

Perfeito! Na Herança, classes semelhantes são agrupadas em uma hierarquia. Ademais, A herança facilita o compartilhamento de comportamento comum entre classes.

Gabarito: Correto

---

32. **(QUADRIX / CRQ 4ª Região-SP – 2018)** Na POO, uma classe possui atributos, que são as características comuns a todos os objetos dela derivados, e métodos, que são as operações que devem estar escritas em cada objeto.

Comentários:

De fato, uma classe possui atributos e métodos, mas objetos não precisam obrigatoriamente possuir todos os métodos. A palavra "devem" acabou deixando o item errado.

Gabarito: Errado

---

33. **(FUNDEP / Prefeitura de Pará de Minas - MG – 2018)** Qual recurso da programação orientada a objetos permite que dois ou mais métodos possuam o mesmo nome desde que utilizem assinaturas diferentes?

a) Encapsulamento.
b) Sobrecarga.
c) Herança.
d) Interface.

Comentários:

Trata-se de uma espécie de polimorfismo! O polimorfismo estático, também conhecido como sobrecarga ou overloading, ocorre quando os nomes dos métodos são iguais e os parâmetros são diferentes.

Gabarito: Letra B

---

34. **(IBADE / IPM-JP – 2018)** No que diz respeito à Orientação a Objetos - OO, analise as abordagens descritas a seguir.

I. Foca o desenvolvimento de um modelo orientado a objetos de um sistema de software para implementar os requisitos especificados. Esses objetos estão relacionados à solução do problema.
II. Foca o desenvolvimento de um modelo orientado a objetos do domínio da aplicação. Esses objetos refletem as entidades e operações associadas ao problema a ser resolvido.

Nesse contexto, I e II são denominadas, respectivamente:

a) Análise OO e Projeto OO.
b) Projeto OO e Análise OO.
c) Programação OO e Análise OO.

d) Projeto OO e Programação OO.
e) Programação OO e Projeto OO.

Comentários:

(I) Um Projeto consiste na modelagem da solução do problema; (II) A Análise foca na modelagem do problema.

Em suma, a Análise consiste em o que deve ser feito, enquanto o Projeto consiste em como deve ser feito.

Gabarito: Letra B

---

35. **(FAURGS / UFRGS – 2018)** Em relação ao paradigma de orientação a objetos, assinale as afirmações abaixo com V (verdadeiro) ou F (falso).

( ) Uma linguagem de programação orientada a objetos pode permitir que uma classe tenha zero, um ou vários métodos construtores distintos.

( ) Um método construtor é responsável por alocar espaço em memória para os atributos do objeto.

( ) Um método construtor padrão atribui valores default para todos os atributos do objeto. Esses valores são obrigatoriamente definidos pela linguagem de programação (ex.: o para atributos numéricos).

( ) Uma classe herdeira não precisa definir um método construtor, pois herda, automaticamente, o construtor da classe base.

( ) Um método construtor não padrão permite que cada objeto de uma classe seja instanciado com valores distintos e adequados ao contexto daquele objeto especificamente.

a) V – F – F – F – V.
b) V – V – F – V – V.
c) F – V – F – V – F.
d) V – F – V – F – V.
e) V – F – V – V – F.

Comentários:

(V) Correto, uma classe pode ter zero, um ou vários métodos construtores, neste último caso por meio da sobrecarga; (F) Errado, na verdade, o método construtor aloca recursos necessários para o funcionamento de um objeto; (F) Errado, ele não atribui valor default, mas sim valor NULL; (F)

Errado, os construtores não são herdados; (V) Correto, é possível inicializá-los com valores distintos.

Gabarito: Letra A

---

36. (COPESE - UFT / Câmara de Palmas - TO – 2018) Em Orientação a Objetos, a associação possibilita um relacionamento entre classes/objetos, no qual estes possam pedir ajuda a outros e assim representar de forma completa o conceito no qual se destinam. Neste tipo de relacionamento, as classes e os objetos interagem entre si para atingir seus objetivos. São os tipos de uma associação, EXCETO:

a) acoplamento.
b) agregação.
c) composição.
d) dependência.

Comentários:

A associação ocorre quando duas classes estão relacionadas. Ademais, são tipos específicos de associação a agregação, a composição e a dependência.

Gabarito: Letra A

---

37. (QUADRIX / CRM-PR – 2018) Com o polimorfismo, é possível que uma operação seja implementada, em uma classe-filha, de forma diferente da classe-pai.

Comentários:

Polimorfismo é o princípio pelo qual duas ou mais classes, derivadas de uma mesma superclasse, podem invocar métodos que têm a mesma identificação, mas comportamentos distintos.

Gabarito: Correto

---

38. (QUADRIX / CRM-PR – 2018) É obrigatório que uma classe possua, no mínimo, um atributo.

Comentários:

Não há obrigatoriedade que uma classe tenha atributos, ou seja, é possível criar uma classe que não tenha atributos.

Gabarito: Errado

---

39. (QUADRIX / CRM-PR – 2018) Com a herança, é admitido estabelecer relações entre classes, permitindo o compartilhamento de atributos e operações idênticas.

Comentários:

Perfeito! É justamente o que a herança faz: facilitar o compartilhamento comum entre classes com o intuito de reaproveitar código ou comportamento generalizado ou especializar operações ou atributos.

Gabarito: Correto

---

40. (QUADRIX / CRM-PR – 2018) Uma classe especifica a estrutura de um objeto, informando quais serão seus valores.

Comentários:

Uma classe é um agrupamento de objetos, é a descrição dos atributos e serviços comuns a um grupo de objetos (reais ou abstratos). Assim, ela especifica o nome da classe, os atributos e os métodos, mas não os valores.

Por exemplo: uma classe Gato terá: atributos (cor e o peso) e métodos (miar e caçar). Veja que a classe especifica a estrutura do objeto, mas não informa os valores, pois cada gato terá uma cor e um peso diferente.

Gabarito: Errado

---

41. (QUADRIX / CRM-PR – 2018) Embora a orientação a objetos seja um paradigma eficaz de análise e desenvolvimento de sistema, com ela, torna-se difícil abstrair, de uma maneira mais fidedigna, as situações do mundo real.

Comentários:

A orientação a objetos é realmente eficaz, no entanto, ela torna mais fácil o entendimento das situações do mundo real, ou seja, a sua principal finalidade é facilitar a vida dos programadores.

Gabarito: Errado

---

42. (QUADRIX / CRM-PR – 2018) Algumas linguagens de programação orientadas a objeto são baseadas em classes, mas há outras que não utilizam as classes de objetos.

Comentários:

Perfeito! Um exemplo de uma linguagem orientada a objetos que não usa classes é o Javascript. Ela é chamada de linguagem baseada em protótipos pois ela tem apenas objetos.

Gabarito: Correto

---

43. (QUADRIX / SEDF – 2018) A sobreposição é um tipo importante de polimorfismo, também conhecida como polimorfismo ad-hoc. Ela permite que o programador use o mesmo nome de método para muitos métodos diferentes.

Comentários:

O polimorfismo ad-hoc é aquele que pode trabalhar com um número finito de tipos. Ele é subdividido em duas categorias: polimorfismo de sobrecarga e de coerção. Com isso não há que se falar em sobreposição e, sim, sobrecarga.

Gabarito: Errado

---

44. (QUADRIX / SEDF – 2018) Na programação orientada a objetos, o estado do objeto é representado ao armazenar valores em campos.

Comentários:

O estado de um objeto é o conjunto de valores das propriedades de um objeto.

Gabarito: Correto

---

45. (COMPERVE / UFRN – 2018) A orientação a objetos é um paradigma importante para a programação de sistemas. Sobre esse tipo de paradigma, é correto afirmar:

a) apenas classes abstratas podem referenciar métodos abstratos.
b) uma classe pode ser abstrata sem possuir nem referenciar métodos abstratos.
c) um método é abstrato por possuir parâmetros com tipos abstratos ainda não definidos.
d) métodos abstratos são aqueles cujo código invoca métodos ainda não implementados.

Comentários:

(a) Errado, interfaces também podem referenciar métodos abstratos; (b) Correto, não necessariamente uma classe abstrata deve possuir métodos abstratos; (c) Errado, na verdade, ele é abstrato quando não é implementado; (d) Errado, métodos abstratos são por natureza não implementados

Gabarito: Letra B

46.(CS-UFG / Câmara de Goiânia - GO – 2018) Polimorfismo é um conceito usado em programação orientada a objetos, e envolve a seleção dinâmica baseada

a) no objeto referenciado em vez do tipo da referência ao objeto, quando há a sobrescrita (overriding) de métodos.

b) no objeto referenciado em vez do tipo da referência ao objeto, quando há a sobrecarga (overloading) de métodos.

c) no tipo da referência ao objeto em vez do objeto referenciado, quando há a sobrescrita (overriding) de métodos.

d) no tipo da referência ao objeto em vez do objeto referenciado, quando há a sobrecarga (overloading) de métodos.

Comentários:

Polimorfismo pode ser estático ou dinâmico. O estático é conhecido como polimorfismo por sobrecarga ou overloading; o dinâmico é conhecido como polimorfismo por sobrescrita ou overriding. Além disso, a seleção dinâmica é baseada no objeto e não no tipo.

Gabarito: Letra A

47.(CESGRANRIO / Transpetro – 2018) Qual a propriedade, típica da orientação a objeto, que habilita uma quantidade de operações diferentes a ter o mesmo nome, diminuindo o acoplamento entre objetos?

a) Encapsulamento
b) Especialização
c) Herança
d) Padrões de projeto
e) Polimorfismo

Comentários:

Várias operações (métodos) com o mesmo nome? Estamos falando do polimorfismo. Ademais, o acoplamento diminui, pois, a dependência dos módulos do software diminui.

Gabarito: Letra E

48.(FAURGS / TJ-RS – 2018) Considere as seguintes afirmações sobre herança.

I - Herança é um dos diferenciadores-chaves entre sistemas convencionais e sistemas orientados

a objetos. Uma subclasse Y herda todos os atributos e operações associadas a sua superclasse X, ou seja, todos as estruturas de dados e operações de X ficam imediatamente disponíveis para Y.

II - Em cada nível de uma hierarquia de classes com herança, novos atributos e operações não podem ser acrescentados àqueles que foram herdados de níveis mais altos da hierarquia.

III. A herança pode proporcionar benefício significativo ao projeto, mas, se for usada de forma não apropriada, pode complicar um projeto desnecessariamente e resultar em um software passível de erros e difícil de manter.

Quais estão corretas?

a) Apenas I.
b) Apenas I e II.
c) Apenas I e III.
d) Apenas II e III.
e) I, II e III.

Comentários:

(I) Correto, vimos que na herança a classe que herda é chamada de subclasse (classe-filha) e a classe que é herdada é chamada de superclasse (classe-pai/mãe); (II) Errado, as subclasses podem sim ter outros métodos e atributos, não há nenhum problema nisso; (III) Correto, a herança deve ser usada de forma apropriada para evitar erros no software.

Gabarito: Letra C

49.(FAURGS / TJ-RS – 2018) _____ é uma característica que reduz bastante o esforço necessário para ampliar o projeto de um sistema orientado a objetos, permitindo que várias operações diferentes tenham o mesmo nome.

a) Encapsulamento
b) Agregação
c) Acoplamento
d) Polimorfismo
e) Coesão

Comentários:

Várias operações (métodos) com o mesmo nome? O enunciado está tratando do conceito de polimorfismo. O Polimorfismo é caracterizado quando duas ou mais classes distintas têm métodos

de mesmo nome, de forma que uma função possa utilizar um objeto de qualquer uma das classes polimórficas, sem necessidade de tratar de forma diferenciada conforme a classe do objeto.

Gabarito: Letra D

---

50. (FAURGS / TJ-RS – 2018) No contexto da orientação a objetos, _____ é um conceito que encapsula dados e abstrações procedurais necessárias para descrever o conteúdo e comportamento de alguma entidade do mundo real.

a) herança
b) polimorfismo
c) classe
d) método
e) mensagem

Comentários:

Essa questão basicamente retirou a definição de classe do Pressman (8ª edição):

"Classe é um conceito de orientação a objeto que encapsula os dados e as abstrações procedurais necessárias para descrever o conteúdo e o comportamento de alguma entidade do mundo real".

Gabarito: Letra C

---

51. (FAURGS / BANRISUL – 2018) Considere as seguintes afirmações sobre alguns fundamentos de Análise de Sistemas orientada a objetos.

I - Classe é um conceito orientado a objeto que encapsula dados e abstrações procedurais necessárias para descrever o conteúdo e o comportamento de alguma entidade do mundo real. Pode-se dizer que classe é uma descrição generalizada que descreve uma coleção de objetos similares.
II - Superclasse é a generalização de um conjunto de classes a ela relacionadas.
III - Subclasse é uma especialização da superclasse. Uma subclasse herda todos os atributos e operações associadas à sua superclasse e não pode incorporar atributos ou operações adicionais específicos.

Quais estão corretas?

a) Apenas I.
b) Apenas I e II.
c) Apenas I e III.
d) Apenas II e III.
e) I, II e III.

Comentários:

(I) Correto, essa é uma definição de classe dada por Pressman; (II) Correto, uma superclasse é generalização de subclasses, e uma subclasse é uma especialização de superclasse; (III) Errado, na verdade, ela pode – sim – incorporar atributos e operações adicionais.

Gabarito: Letra B

---

52. **(IF-TO / IF-TO – 2018)** Considere o cenário onde uma Classe B lega suas estruturas e comportamentos de uma Classe A. Essa relação entre a Classe A e a Classe B é caracterizada por:

a) uma anomalia
b) um polimorfismo
c) uma herança
d) uma instância da classe
e) uma anomalia de instância

Comentários:

A palavra lega está relacionada a herdar, ou seja, a classe B herda estruturas e comportamentos da classe A. Ocorre então uma herança.

Gabarito: Letra C

---

53. **(CEPS-UFPA / UFPA – 2018)** Em relação ao paradigma orientado a objetos, é CORRETO afirmar que:

a) um objeto consiste em um conjunto de operações encapsuladas e um estado que grava e recupera os efeitos dessas operações.

b) mensagens são requisições enviadas de um atributo para outro, para que o objeto receptor forneça algum resultado por meio da execução de uma operação.

c) classe define as características de um conjunto de atributos que podem herdar dados de outras classes.

d) polimorfismo permite que mensagens sejam propagadas somente utilizando um único método para diferentes classes.

e) a herança permite que objetos herdem atributos de outras classes, sem estender esta característica para métodos.

Comentários:

(a) Correto, em um objeto o conjunto de operações encapsuladas são os métodos, além disso o estado é determinado pelo valor dos atributos; (b) Errado, mensagens são enviadas de um objeto para outro, e não de um atributo para outro; (c) Errado, as classes definem características de uma coleção de objetos; (d) Errado, polimorfismo pode utilizar diferentes métodos; (e) Errado, os objetos herdam todos os componentes da classe pai.

Gabarito: Letra A

---

54. **(FUNRIO / AL-RR – 2018)** Um programador, utilizando orientação a objeto, deseja fazer com que os atributos e serviços disponíveis em uma classe estejam escondidos, de modo que o acesso aos mesmos se dê através de interfaces. O mecanismo de orientação a objetos, adequado para essa característica, é o /a:

a) encapsulamento.
b) estruturação.
c) persistência.
d) polimorfismo.

Comentários:

O encapsulamento que visa esconder detalhes internos de um objeto.

Gabarito: Letra A

---

55. **(CS-UFG / UFG – 2018)** Na programação orientada a objetos, há um mecanismo que permite definir modificadores de acesso. Quando se define um atributo de uma classe com o modificador de acesso privado, significa que:

a) o acesso à classe é privado.
b) o atributo é acessível a um programa que tenha uma referência a um objeto da classe.
c) a classe é abstrata.
d) o atributo é acessível somente aos métodos da classe.

Comentários:

Com modificador de acesso privado, atributos e métodos podem apenas serem modificados no local em que foram definidos. Logo, um atributo somente é acessível dentro da classe em que foi criado.

Gabarito: Letra D

---

56. (CS-UFG / SANEAGO - GO – 2018) Em programação orientada a objetos, o uso de composição ao invés de herança é preferível porque

a) facilita o emprego de estruturas de decisão e controle.
b) reduz o acoplamento.
c) elimina bugs.
d) minimiza o consumo de memória e CPU.

Comentários:

Tanto a herança como a composição são utilizadas para a reutilização de funcionalidades. Em uma composição o todo depende da parte, ou seja, a parte não pode existir sem a existência do todo. Ademais, vimos que o acoplamento trata do nível de dependência entre os módulos. Logo, em uma composição o acoplamento é menor.

Gabarito: Letra B

---

57. (CS-UFG / SANEAGO - GO – 2018) A programação orientada a objetos:

a) impossibilita o polimorfismo sem herança (extends em JAVA).
b) impede a construção de software de difícil manutenção.
c) faz uso de conceitos como classe, interfaces e envio de mensagens.
d) requer que classes sejam estendidas para a reutilização de código.

Comentários:

(a) Errado, é possível ter polimorfismo sem que se use herança; (b) Errado, não há esse impedimento; (c) Correto, ela realmente usa os conceitos de classe, interface e envio de mensagens; (d) Errado, não necessariamente a reutilização de classes deve ser feita dessa forma, pode ser feita, por exemplo, com herança.

Gabarito: Letra C

---

58. (CS-UFG / SANEAGO - GO – 2018) Uma classe abstrata A contém o método abstrato foo(), que não foi reimplementado pela classe B que herda de A. Nesse contexto,

a) a criação de uma classe abstrata C, que herda de B, requer a implementação do método foo().
b) a implementação do método foo() em B é obrigatória para que ela compile.
c) a chamada do método foo() de um objeto de B chamará a implementação existente em A.
d) a classe B não pode sobrecarregar o método foo().

Comentários:

(a) Errado, uma classe abstrata que herda de outra classe abstrata não precisa ser implementar os métodos abstratos; (b) Correto, se o método abstrato foo() foi herdado da classe A, ele deve ser implementado na classe B; (c) Errado, foo() deve ser implementado em B; (d) Errado, ele pode sobrecarregar.

Gabarito: Letra B

---

59. (PR-4 UFRJ / UFRJ – 2018) Com relação aos conceitos de orientação objeto, existe uma característica que faz com que detalhes internos do funcionamento dos métodos de uma classe permaneçam ocultos para os objetos e que por conta dessa técnica, o conhecimento a respeito da implementação interna da classe é desnecessário do ponto de vista do objeto, uma vez que isso passa a ser responsabilidade dos métodos internos da classe. A característica apresentada se refere a:

a) encapsulamento.
b) polimorfismo.
c) abstração.
d) herança.
e) namespaces.

Comentários:

Muita atenção com as palavras-chave da questão. O mecanismo que oculta detalhes internos do funcionamento dos métodos de uma classe é o encapsulamento. Além disso, a implementação interna, ou seja, a forma pela qual a operação é realizada não precisa ser conhecida pelo requisitante, ele precisa apenas saber que o receptor sabe realizar tal operação.

Gabarito: Letra A

---

60. (IBFC / Prefeitura de Divinópolis-MG – 2018) Assinale a alternativa que complete correta e respectivamente as lacunas da frase a seguir:

O _____ permite que referências de tipos de classes mais _____ representem o comportamento das classes _____que referenciam:

a) polimorfismo - abstratas - concretas
b) encapsulamento - abstratas - concretas
c) encapsulamento - concretas - abstratas
d) polimorfismo - concretas - abstratas

Comentários:

O polimorfismo permite que referências de tipos de classes mais abstratas representem o comportamento das classes concretas que referenciam. Esse recurso permite tratar vários tipos diferentes de maneira homogênea, por meio da interface do tipo mais abstrato. Em suma, o polimorfismo busca obter um comportamento específico através de uma mesma interface da classe.

Gabarito: Letra A

---

61. (IBFC / TJ-PE – 2017) Em um programa orientado a objetos, verifica-se que a classe X estende a classe Y. Ou seja, pode-se afirmar, pelos preceitos da POO (Programação Orientada a Objetos), que:

a) a classe X é superclasse de Y
b) a classe X é uma interface de Y
c) a classe Y é derivada de X
d) a classe Y é subclasse de X
e) a classe X é derivada de Y

Comentários:

Se a Classe X estende a Classe Y, então a Y é superclasse de X. Dito de outra forma, X é subclasse de Y ou X é classe derivada de Y.

Gabarito: Letra E

---

62. (IBFC / EMBASA – 2017) Quanto aos fundamentos básicos de programação orientada a objetos, relacione os quatro conceitos abaixo com os respectivos significados mencionados logo em seguida:

CONCEITOS:

(1) herança.
(2) método.
(3) polimorfismo.
(4) encapsulamento.

SIGNIFICADOS:

(A) definem as habilidades dos objetos.
(B) é o princípio pelo qual duas ou mais classes, derivadas de uma mesma superclasse, podem invocar métodos que têm a mesma identificação mas comportamentos distintos.
(C) é o mecanismo pelo qual uma classe pode estender outra classe ou, ainda, ser estendida de outra classe.

(D) consiste na separação de aspectos internos e externos de um objeto.

a) 1C - 2B - 3A - 4D
b) 1C - 2A - 3B - 4D
c) 1D - 2A - 3B - 4C
d) 1A - 2C - 3D - 4B.

Comentários:

(1C) Herança é o mecanismo pelo qual uma classe pode estender outra classe ou, ainda, ser estendida de outra classe; (2A) Métodos definem as habilidades dos objetos; (3B) Polimorfismo é o princípio pelo qual duas ou mais classes, derivadas de uma mesma superclasse, podem invocar métodos que têm a mesma identificação, mas comportamentos distintos; (4D) Encapsulamento consiste na separação de aspectos internos e externos de um objeto.

Gabarito: Letra B

---

63. (IBFC / EBSERH – 2017) Um dos conceitos em Orientação a Objetos é a classe abstrata. Assinale a alternativa que complete correta e respectivamente as lacunas da frase abaixo:

"A classe abstrata é sempre um(a) _____ que não possui _____"

a) método - instâncias
b) ator/atriz - atributos
c) superclasse - instâncias
d) superclasse - atores/atrizes
e) método - atributos.

Comentários:

A classe abstrata é sempre uma superclasse que não possui instâncias – lembrem-se que ela não pode ser instanciada.

Gabarito: Letra C

---

64. (IBFC / EBSERH – 2017) Consiste no princípio pelo qual duas ou mais classes derivadas de uma mesma superclasse podem invocar métodos que têm a mesma identificação (assinatura) mas comportamentos distintos, especializados para cada classe:

a) abstração
b) herança
c) interface
d) mensagem

e) polimorfismo

Comentários:

Princípio pelo qual duas ou mais classes derivadas de uma mesma superclasse podem invocar métodos que têm a mesma identificação (assinatura), mas comportamentos distintos é o princípio do... polimorfismo.

Gabarito: Letra E

---

65. (CESGRANRIO / UNIRIO – 2016) Em linguagens orientadas a objetos (OO), classes representam a descrição da implementação de tipos abstratos a partir dos quais instâncias podem ser criadas. Cada instância, depois de criada, guarda seu estado próprio independente das demais instâncias. Esse estado pode ser alterado de acordo com operações definidas pela classe, mas, ao serem executadas, as operações atuam individualmente sobre cada instância.

Na nomenclatura OO, instâncias e operações são conhecidas, respectivamente, como

a) Métodos e Funções
b) Objetos e Heranças
c) Objetos e Métodos
d) Tipos e Objetos
e) Tipos e Heranças

Comentários:

Os objetos são a materialização em runtime (tempo de execução) das classes, enquanto o meio de ter acesso aos seus atributos modificando o estado são realizados através dos métodos.

Gabarito: Letra C

---

66. (IBFC / EMDEC – 2016) Quanto a Programação Orientada a Objeto identifique a alternativa que representa o mecanismo pelo qual uma classe (sub-classe) pode estender outra classe (super-classe), aproveitando seus comportamentos (métodos) e variáveis possíveis (atributos):

a) herança
b) encapsulamento
c) mensagem
d) polimorfismo.

Comentários:

Mecanismo pelo qual uma subclasse pode estender uma superclasse (opa, isso já parece herança...) aproveitando seus comportamentos e variáveis (é, realmente é herança!).

Gabarito: Letra A

---

67. (CESGRANRIO / EPE – 2014) Considere que um programa orientado a objeto possui 5 classes: Máquina, Motor, MotorExplosão, MotorVapor e Gerador. MotorExplosão e MotorVapor são especializações de Motor. Motor e Gerador são especializações de Máquina. Todas as classes respondem a uma mensagem chamada "calcularPotencia", sem argumentos, que calcula e retorna um número real que indica potência do objeto, em watts, de acordo com os valores de alguns atributos, com um algoritmo diferente em cada classe. O exemplo acima caracteriza a capacidade de enviar a mesma mensagem para vários objetos e que cada objeto responda a essa mensagem de acordo com sua classe. Tal característica é conhecida como:

a) Polimorfismo
b) Refatoração
c) Herança Múltipla
d) Independência de Dados
e) Tratamento de Exceção

Comentários:

(a) Correto. Polimorfismo permite que referências de tipos de classes mais abstratas (Pai) representem o comportamento das classes concretas (Filhas) que referenciam. Assim, é possível tratar vários tipos de maneira homogênea (através da interface do tipo mais abstrato). Ele é caracterizado quando duas ou mais classes distintas têm métodos de mesmo nome, de forma que uma função possa utilizar um objeto de qualquer uma das classes polimórficas, sem necessidade de tratar de forma diferenciada conforme a classe do objeto. Uma das formas de implementar o polimorfismo é através de uma classe abstrata, cujos métodos são declarados, mas não são definidos, e através de classes que herdam os métodos desta classe abstrata. Outra forma é através da sobrecarga de métodos de mesmo nome, mas com assinaturas diferentes; (b) Errado. Consiste no processo de modificação um sistema de software para melhorar a estrutura interna do código sem alterar seu comportamento externo; (c) Errado. Consiste em implementar uma classe através da herança de duas ou mais classes; (d) Errado. Conceito de Banco de Dados que diz que é permitido efetuar alterações no esquema ou no nível de um banco de dados, sem alterar um nível superior. Existem dois tipos: independência de dados lógica e independência de dados física; (e) Errado. Trata-se de um mecanismo capaz de realizar o tratamento da ocorrência de condições que alteram o fluxo normal da execução de programas de computadores. O fluxo então é alterado para uma nova condição que trata a condição não contida no fluxo normal.

Gabarito: Letra A

---

68. (CESGRANRIO / IBGE – 2014) Em linguagens orientadas a objetos, existem dois conceitos fundamentais:

I – a definição de uma estrutura, a partir da qual é possível especificar todas as características da implementação, operações e armazenamento de informações para instâncias que serão criadas posteriormente.

II – instâncias específicas criadas a partir da definição das estruturas referentes ao conceito I.

Esses conceitos correspondem, respectivamente, ao que se conhece pelos nomes de:

a) I - Tipo; II - Classe
b) I - Tipo; II - Construtor
c) I - Classe; II - Tipo
d) I - Classe; II - Objeto
e) I - Classe ; II - Metaclasse

Comentários:

Classe é uma definição esquemática abstrata que se manifesta em tempo de execução através de suas instâncias chamadas Objetos.

Gabarito: Letra D

---

69. (IBFC / TRE-AM – 2014) Em programação orientada a objetos significa separar o programa em partes, o mais isoladas possível, tornando o software mais flexível e fácil de modificar:

a) Herança.
b) Encapsulamento.
c) Polimorfismo.
d) Atributo.

Comentários:

Separar em partes? Isoladas? Tornando o software mais flexível e fácil de modificar? Trata-se de características típicas do encapsulamento!

Gabarito: Letra B

---

70. (IBFC / PC-RJ – 2013) Quanto à programação orientada a objeto, simplificadamente classe é o conjunto de objetos com características similares. O conjunto de atributos e métodos agregados a um só objeto, que podem ser visíveis ou invisíveis, é denominado de:

a) Evento
b) Subclasse.
c) Estado.
d) Encapsulamento.
e) Herança.

Comentários:

O conjunto de atributos e métodos agregados a um só objeto (opa, parece encapsulamento), que podem ser visíveis ou invisíveis... agora só pode ser encapsulamento mesmo.

Gabarito: Letra D

---

71. **(IBFC / HEMOMINAS – 2013)** Complete a frase a seguir com uma das alternativas abaixo: "_____ permite que os atributos de classes possam ser declarados como públicos, privados ou protegidos".

a) Polimorfismo.
b) Herança.
c) Abstração.
d) Encapsulamento.

Comentários:

Encapsulamento permite que os atributos de classes possam ser declarados como públicos, privados ou protegidos.

Gabarito: Letra D

---

72. **(ESAF / DNIT – 2013)** A herança de D a partir de C é a habilidade que uma classe D tem implicitamente definida:

a) em atributos e análises da classe C.
b) em cada um dos modelos e concepções da classe C.
c) em cada um dos atributos e operações da classe C.
d) em parte das funcionalidades e operações de classes equivalentes.
e) nos programas das classes.

Comentários:

A herança (de D a partir de C) é a habilidade que uma classe D tem implicitamente definida em cada um dos atributos e operações da classe C, como se esses atributos e operações tivessem sido

definidos com base na própria classe D. C é caracterizada como uma superclasse de D. Em contrapartida, D é caracterizada com uma subclasse de C.

Gabarito: Letra C

---

73. (ESAF / CGU – 2012) Assinale a opção correta.

a) As classes podem formar heranças segmentadas em classes adjacentes.

b) Overflow é a redefinição do fluxo de uma classe, em uma de suas subclasses.

c) Overriding é a redefinição de um método, definido em uma classe, em uma de suas subclasses.

d) Overriding é a redefinição de uma classe através de métodos de objetos diferentes.

e) As classes não podem formar hierarquias de herança de superclasses e subclasses.

Comentários:

(a) Errado, classes não podem formar heranças segmentadas – ou herda tudo ou não herda nada; (b) Errado, esse não é um conceito de orientação a objetos; (c) Correto, overriding é a redefinição de um método, definido em uma classe, em uma de suas subclasses; (d) Errado, não é redefinição de classe; (e) Errado, claro que podem – é exatamente isso que elas fazem.

Gabarito: Letra C

---

74. (IADES / PGDF – 2011) Dentro do paradigma de programação orientada a objetos (POO), há um mecanismo utilizado para impedir o acesso direto ao estado de um objeto, restando apenas os métodos externos que podem alterar esses estados. Assinale a alternativa que apresenta o nome deste mecanismo.

a) Mensagem
b) Herança
c) Polimorfismo
d) Encapsulamento
e) Subclasse

Comentários:

Observem que a questão fala em "*mecanismos utilizado para impedir o acesso direto ao estado de um objeto*". Trata-se, portanto, do mecanismo de encapsulamento.

Gabarito: Letra D

---

75. (CESGRANRIO / PETROBRÁS – 2010) Análise as afirmativas a seguir relativas ao paradigma da orientação a objetos.

I - O princípio do encapsulamento preconiza que um objeto deve esconder a sua complexidade interna.

II - Uma mensagem de um objeto A para um objeto B indica que A realizou uma tarefa requisitada por B.

III - A existência da mesma operação polimórfica definida em duas classes, ClasseA e ClasseB, implica necessariamente que ou ClasseA seja subclasse de ClasseB, ou que ClasseB seja subclasse de ClasseA.

É correto APENAS o que se afirma em:

a) I.
b) II.
c) I e II.
d) I e III.
e) II e III.

Comentários:

(I) Correto. De fato, o princípio do encapsulamento preconiza que um objeto deve esconder sua complexidade interna. Recomenda-se esconder os atributos e expor os métodos; (II) Errado. Uma mensagem de um Objeto A para um Objeto B indica que B realizou uma tarefa requisitada por A; (III) Errado. Imagine que ClasseA e ClasseB são ambas filhas de ClasseC. A existência de uma mesma operação polimórfica definida nas duas classes (uma sobrescrita, por exemplo) não implica necessariamente que ou ou ClasseA seja subclasse de ClasseB, ou que ClasseB seja subclasse de ClasseA. Ambas podem ser filhas de ClasseC.

Gabarito: Letra A

---

76. (ESAF / SUSEP – 2010) Em relação à programação orientada a objetos, é correto afirmar que:

a) o objeto é definido por atributos.
b) objetos são instâncias de um atributo.
c) apenas atributos numéricos são válidos.
d) atributos podem ser agrupados em pointvalues.
e) atributos adequados dispensam referências a objetos.

Comentários:

(a) Correto, um objeto é definido por meio de seus atributos; (b) Errado, objetos são instâncias de classes; (c) Errado, há diversos outros tipos de atributos; (d) Errado, não faz sequer sentido essa sentença; (e) Errado, atributos sempre se referem a objetos.

Gabarito: Letra A

---

77. (ESAF / SUSEP – 2010) É correto afirmar que em herança simples uma superclasse pode ter apenas uma subclasse.

Comentários:

Opa... herança simples significa que uma subclasse tem apenas uma superclasse direta.

Gabarito: Errado

---

78. (ESAF / SUSEP – 2010) Polimorfismo é a:

a) utilização múltipla de programas em análise orientada a objetos.

b) habilidade de uma única operação ou nome de atributo ser definido em mais de uma classe e assumir diferentes implementações em cada uma dessas classes.

c) habilidade de um programador em desenvolver aplicações e caracterizar objetos com múltiplos atributos.

d) utilização de uma classe com diferentes formatos em programas com definição de objetos e atributos.

e) habilidade de uma única variável ser utilizada em diferentes programas orientados a objetos.

Comentários:

Polimorfismo é a habilidade de uma única operação ou nome de atributo ser definido em mais de uma classe e assumir diferentes implementações em cada uma dessas classes.

Gabarito: Letra B

---

79. (IADES / IADES – 2010) A análise de sistemas no mundo orientado a objeto é feita analisando-se os objetos e os eventos que interagem com esses objetos. O projeto de software é feito reusando-se classes de objetos existentes e, quando necessário, construindo-se novas classes. Análise e projeto orientados a objeto modelam o mundo em termos de objetos que têm

propriedades e comportamentos e eventos que disparam operações que mudam o estado dos objetos que interagem entre si. Sobre os conceitos ou ideias fundamentais da metodologia da análise de sistemas orientada a objeto, assinale a alternativa incorreta.

a) Uma classe é a implementação de software de um tipo de objeto, podendo ser abstrata (quando possui objetos instanciados a partir dela) ou concreta (quando não possui objetos criados a partir dela).

b) Um objeto é qualquer coisa, real ou abstrata, a respeito do qual armazenamos dados e os métodos que os manipulam.

c) Um método de um tipo de objeto referência somente as estruturas de dados desse tipo de objeto. Comparativamente, é similar às funções e procedures do universo da programação.

d) O encapsulamento é importante porque separa a maneira como um objeto se comporta da maneira como ele é implementado, uma vez que a definição sobre como implementar os conhecimentos ou ações de uma classe não são informadas.

Comentários:

(a) Errado, quando concreta, ela pode instanciar objetos a partir dela; quando abstrata, ela não pode instanciar objetos a partir dela. (b) Correto, é isso mesmo; (c) Errado, um método pode – sim – referenciar estruturas de dados de outro objeto. (d) Correto, é exatamente assim que funciona o encapsulamento.

Bem, observem que temos duas respostas erradas. *O que fazer?* Achem a mais errada – vocês verão que, infelizmente, isso é muito comum em provas de múltipla escolha. A mais errada é a primeira, porque a terceira ainda pode ser interpretada como um método estático, apesar de a questão não dizer!

Gabarito: Letra A

Lista de Questões – CESPE

1. **(CESPE / CAU-BR – 2024)** Em sistemas orientados a objetos, os objetos podem ser de natureza física, por exemplo, cadeira, ou de natureza conceitual, por exemplo, inscrição em um curso.

2. **(CESPE / TC-DF – 2023)** O "problema do diamante", que surge quando uma classe herda de duas classes que compartilham uma mesma classe pai, não resulta em conflitos de métodos ou ambiguidades na resolução de herança múltipla.

3. **(CESPE / TC-DF – 2023)** Considere-se que, em um encapsulamento de uma classe de nome Carro para um sistema de automóveis, exista um atributo privado de nome quilometragem. Nesse caso, ao se fornecerem métodos públicos do tipo obter_quilometragem() e atualizar_quilometragem(), protegem-se detalhes internos da classe Carro.

4. **(CESPE / MPE-RO – 2023)** Em orientação a objetos, o conceito utilizado para descrever os vários comportamentos que um método possui, visando a um melhor aproveitamento de partes de um código, é denominado:

   a) herança.
   b) encapsulamento.
   c) atributos.
   d) polimorfismo.
   e) interface.

5. **(CESPE / DATAPREV – 2023)** No paradigma da orientação a objetos, o polimorfismo permite que várias operações distintas possuam o mesmo nome, desacoplando, assim, os objetos uns dos outros, tornando-os mais independentes.

6. **(CESPE / DATAPREV – 2023)** Herança é uma característica do paradigma orientado a objetos, a qual possibilita que haja hierarquia de classes, de forma que as alterações em uma classe-pai possam ser imediatamente propagadas para a classe-filha.

7. **(CESPE / BANRISUL – 2022)** Nas classes abstratas, que servem de modelo para outras classes, é obrigatória a existência de pelo menos um método abstrato, sem corpo.

8. **(CESPE / BANRISUL – 2022)** Em um projeto orientado a objetos, a decomposição do sistema em objetos é influenciada por fatores como encapsulamento, granularidade e desempenho.

9. **(CESPE / TELEBRÁS - 2021)** Na orientação a objetos, o polimorfismo permite que os programadores definam comportamentos diferentes para um mesmo método. Além disso, dados os tipos de polimorfismo, é possível que, dentro de uma herança, um comportamento seja reescrito à medida que a classe se torna mais específica, possibilitando que seja criada uma abstração mais próxima do mundo real, facilitando a compreensão do código como um todo.

No polimorfismo, a decisão sobre qual método deve ser selecionado é tomada em tempo de concepção.

10. **(CESPE / Petrobrás - 2022)** A POO é embasada nos conceitos de classe, objeto, encapsulamento, herança, interfaces e polimorfismo; uma característica das interfaces, por exemplo, é o fato de que elas não podem ser implementadas por uma classe, mas sim herdadas.

11. **(CESPE / FUB – 2018)** Na orientação a objetos, estes possuem diversos atributos e métodos, os quais são utilizados para se definir as características e ações das classes.

12. **(CESPE / BNB – 2018)** O encapsulamento em uma classe garante que seus métodos e suas variáveis tenham alta coesão e baixo acoplamento, seguindo os objetivos básicos da programação orientada a objetos.

13. **(CESPE / BNB – 2018)** As interfaces são definições a respeito de como um objeto pode ser utilizado por outros objetos, sem envolver necessariamente uma interação com o usuário.

14. **(CESPE / BNB – 2018)** De acordo com o conceito de herança, uma classe derivada é uma implementação mais genérica da classe da qual ela deriva, o que permite a reutilização de métodos e de variáveis.

15. **(CESPE / BNB – 2018)** Em programação orientada a objetos, a técnica utilizada para esconder detalhes internos de funcionamento de uma classe é denominada generalização.

16. **(CESPE / TCE-MG – 2018)** Em uma programação orientada a objetos, a técnica de programação que mantém ocultos detalhes internos do funcionamento dos métodos de uma classe é denominada

a) encapsulamento.
b) polimorfismo.
c) generalização.
d) abstração.
e) herança.

17. **(CESPE / ABIN – 2018)** Se, em tempo de execução de um sistema, ocorrer associação entre uma entidade e um atributo, então essa associação será considerada um acoplamento dinâmico.

18. **(CESPE / ABIN – 2018)** Considere que, em um sistema de informações, um objeto possua dados de uma pessoa, tais como: nome, endereço, data de aniversário e número do cartão de crédito. Considere, ainda, que esse sistema exponha, de forma pública, informações sobre o nome e a data de aniversário e deixe os dados do cartão de crédito protegidos em formato privado. Nesse caso, o sistema estará usando o recurso de interface.

19. **(CESPE / STM – 2018)** Um recurso de grande utilidade nesse tipo de programação consiste na possibilidade de um objeto exercer o comportamento de outro objeto.

20. **(CESPE / STM – 2018)** Os atributos de um objeto podem ser expostos tanto por meio de um enlace direto a uma variável interna quanto por meio do retorno de um valor por meio de um método.

21. **(CESPE / STM – 2018)** O tipo de herança mais eficiente e indicado é a herança de implementação, pois possibilita que uma nova classe reutilize a implementação de outra classe sem a necessidade de se recortar e colar o código de forma manual, tornando o código automaticamente disponível, como parte da nova classe.

22. **(CESPE / STM – 2018)** O encapsulamento permite que um programa seja dividido em várias partes menores; contudo, as partes tornam-se dependentes umas das outras em relação à implementação e em relação ao trabalho realizado.

23. **(CESPE / STM – 2018)** Em orientação a objetos, os membros de dados de uma classe devem ser acessados por um método específico do objeto, e não diretamente.

24. **(CESPE / SEDF – 2017)** Um objeto define atributos, comportamentos e abstrações comuns compartilhados por um tipo de classe.

25. **(CESPE / TRE-BA – 2017)** Na orientação a objetos, o conceito de polimorfismo é implementado, em algumas linguagens, por meio da técnica de sobrecarga de métodos. Sobre a aplicação desse conceito, é possível afirmar que:

a) um mesmo método pode estar em classes diferentes, com a mesma assinatura, sem que isso prejudique a sua identificação pelo compilador.

b) a identificação da assinatura do método corresponde ao seu nome.

c) uma classe derivada da classe-mãe tem vários métodos com a mesma assinatura, e o compilador realiza o primeiro deles.

d) a classe derivada da classe-mãe herda os métodos da sua classe-mãe na forma como são implementados.

e) cada método tem um nome único na classe derivada da classe-mãe a ser identificado pelo compilador.

26. **(CESPE / TRE-BA – 2017)** A partir de uma classe derivada de uma superclasse, podem-se invocar métodos que tenham a mesma assinatura, mas comportamentos distintos, ou seja, em que haja alteração do funcionamento interno de um método herdado de um objeto pai. Na orientação a objetos, isso é possível por meio de:

a) polimorfismo.
b) abstração.
c) encapsulamento.
d) namespaces.
e) atributos.

27. **(CESPE / TCE-PR – 2016)** Em se tratando de orientação a objetos, o polimorfismo refere-se:

a) ao reconhecimento do comportamento variado de um método, o que melhora o aproveitamento do código.

b) à transmissão dos métodos e atributos de uma classe para suas subclasses, quando ad hoc.

c) à variação das saídas de um método com relação às entradas recebidas, caso em que é considerado universal.

d) ao uso que um objeto faz dos recursos de outro objeto.

e) à utilização de métodos específicos para manipular dados com proteção por meio de encapsulamento.

28. **(CESPE / FUB – 2016)** Uma das vantagens de se utilizar encapsulamento em orientação a objetos é impedir o acesso direto aos atributos de um objeto.

29. **(CESPE / FUB – 2016)** O conjunto de valores das características de determinado objeto é denominado estado.

30. **(CESPE / FUB – 2016)** Quando uma classe é subclasse de várias superclasses, mas somente herda características de uma classe, a herança é do tipo simples.

31. **(CESPE / TRE-GO – 2015)** Uma classe abstrata possui instâncias diretas, bem como suas classes descendentes.

32. **(CESPE / TJDFT – 2015)** Recurso de classes é a propriedade dos objetos que viabiliza a implementação de hierarquia entre objetos.

33. **(CESPE / TJDFT – 2015)** A combinação de dados com o código que os manipula em um único objeto é denominada encapsulamento.

34. **(CESPE / TCE-RN – 2015)** O processo de herança permite a reutilização de código, como também o reaproveitamento de atributos e métodos. Assim, em aplicações que utilizam herança, a obtenção de polimorfismo é uma possibilidade.

35. **(CESPE / MEC – 2015)** Objetos são definidos como entidades da modelagem de sistemas que armazenam estados com a utilização de atributos dos próprios objetos, sem interação com outros objetos por meio de mensagens.

36. **(CESPE / INPI – 2013)** Ao se utilizar o encapsulamento, não é necessário saber como ele funciona internamente, apenas como transmite os seus atributos.

37. **(CESPE / TRE-MS – 2013)** Em programação orientada a objetos, a possibilidade de haver funções de mesmo nome, com funcionalidades similares em classes sem nenhuma relação entre elas, denomina-se:

a) encapsulamento.
b) objeto.
c) classe.
d) polimorfismo.
e) relacionamento hierárquico.

38. **(CESPE / MPU – 2013)** Se uma subclasse herdar características de duas ou mais superclasses, ocorrerá uma herança múltipla.

39. **(CESPE / INPI – 2013)** Em uma operação de sobrecarga, uma classe derivada pode redefinir operações de sua classe base.

40. **(CESPE / TRE-RJ – 2012)** Cada classe pode ter implementações de operação — ou métodos — com denominações únicas. Classes diferentes podem ter métodos com denominações iguais, porém, uma classe não pode ter métodos com denominações iguais e parâmetros diferentes.

41. **(CESPE / MPE-PI – 2012)** É possível que um mesmo objeto tenha mais de um método com o mesmo nome.

42. **(CESPE / TRE-RJ – 2012)** As heranças, que são princípios de orientação a objetos, permitem o compartilhamento de atributos e métodos pelas classes e são usadas com o intuito de se reaproveitar código ou comportamento generalizado ou especializar operações ou atributos.

43. **(CESPE / TCE-ES – 2012)** A implementação de polimorfismo envolve o mecanismo de redefinição de métodos, assim como o conceito de ligação tardia.

44. **(CESPE / TRE-RJ – 2012)** Polimorfismo consiste em focalizar nos aspectos essenciais inerentes a uma entidade e ignorar propriedades acidentais ou menos importantes. No desenvolvimento de sistemas, isso significa concentrar-se no que um objeto é e no que ele faz antes de se decidir como esse objeto será implementado.

45. **(CESPE / TRE-RJ – 2012)** O polimorfismo de sobrecarga é realizado por meio da aplicação de parâmetros diferentes em operadores distintos com o mesmo nome e semânticas distintas.

46. **(CESPE / IPEA – 2012)** A análise orientada a objetos, o projeto orientado a objetos e a programação orientada a objetos compreendem atividades de engenharia de software voltadas à construção de sistemas orientados a objetos. Nesses sistemas, objetos interagem para prover serviços. No nível de programação, as interações ocorrem via interfaces das classes das quais os objetos são instâncias. Essas interfaces contêm membros públicos das classes.

47. **(CESPE / FUB – 2011)** Um objeto possui dados internos e métodos que definem, respectivamente, seu estado atual e seu comportamento. Um objeto pode se comunicar com outros objetos passando mensagens.

48. **(CESPE / TJ-ES – 2011)** Na programação orientada a objetos, o encapsulamento representa a ação em que diversas implementações de uma operação utilizam vários tipos de parâmetros diferentes.

49. **(CESPE / EBC – 2011)** A herança representa uma generalização, dessa forma, por meio de herança é feito o compartilhamento de atributos e operações entre classes, com base em um relacionamento hierárquico.

50. **(CESPE / TRE-ES – 2011)** Em programação orientada a objetos, a herança serve para criar classes que incorporem propriedades e métodos de outras classes. Assim, é possível construir uma classe a partir de outra sem ter de reescrevê-la.

51. **(CESPE / BRB – 2011)** Para que a interface pública de uma classe seja considerada coesa, é necessário que todos os recursos dessa interface estejam relacionados ao conceito que a classe representa.

52. **(CESPE / TRE-ES – 2011)** Objetos de software interagem e comunicam-se com os outros por meio de mensagens. Por exemplo, quando o objeto A deseja que o objeto B execute um de seus métodos, envia a este uma mensagem. Algumas vezes, o objeto receptor precisa de mais informação para que saiba exatamente o que deve fazer, de modo que essa informação seja transmitida juntamente com a mensagem por meio de parâmetros.

53. **(CESPE / MPE-TO – 2011)** Entre os diversos diagramas utilizados em análise e projeto orientados a objetos, o diagrama de casos de uso, por procurar representar todas as possíveis situações de utilização do sistema, é considerado o diagrama responsável por mostrar a estrutura estática do sistema.

54. **(CESPE / TRT-RN – 2010)** Programa que utilize uma linguagem orientada a objetos (OO), ao incorporar corretamente ao seu funcionamento conceitos como os de encapsulamento, herança e polimorfismo, beneficia-se das características da OO.

55. **(CESPE / TRT-RN – 2010)** Além dos conceitos de objeto e classe, o paradigma da orientação a objetos envolve os princípios de:

a) abstração, encapsulamento, herança e polimorfismo.
b) abstração, métodos, instâncias e herança.
c) abstração, encapsulamento, generalização e especialização.
d) generalização, especialização, herança e polimorfismo.
e) atributos, métodos, instâncias e mensagens.

56. **(CESPE / Banco da Amazônia – 2010)** Objetos têm identidade própria. Isso garante que, mesmo tendo os mesmos valores de variáveis e pertencendo à mesma classe, dois objetos sejam considerados diferentes.

57. **(CESPE / ABIN – 2010)** Um objeto apresenta três características básicas, o estado, a identidade e o comportamento. A parte de dados de um objeto é definida por um conjunto de mensagens, e a porção funcional, por um conjunto de atributos.

58. **(CESPE / ABIN – 2010)** Objeto é o agrupamento de classes similares que apresentam os mesmos atributos e operações. Na definição de uma classe, é necessário estabelecer a que objeto ela ocorre como instância.

59. **(CESPE / TCU – 2010)** Uma classe pode ser vista como uma descrição generalizada de uma coleção de objetos semelhantes.

60. **(CESPE / MPU – 2010)** Uma mensagem enviada a um objeto pode levar à execução de um método que não esteja implementado na classe à qual o objeto pertence.

61. **(CESPE / DETRAN-ES – 2010)** Um dos conceitos em programação orientada a objetos é o de abstração, por meio da qual as características do mundo real podem ser modeladas, por exemplo, mediante o agrupamento de objetos e classes.

62. **(CESPE / MPU – 2010)** Considerando as características do relacionamento entre uma classe e suas subclasses, é correto afirmar que toda implementação de subclasse é polimórfica.

63. **(CESPE / Banco da Amazônia – 2010)** A herança é um conceito implementado por todas as linguagens de programação orientadas a objeto. No entanto, algumas delas somente permitem o uso de herança simples, não sendo possível a criação de classes por meio de herança múltipla.

64. **(CESPE / TRT-RN – 2010)** Uma subclasse, por ser derivada de uma superclasse e possuir todos os atributos da superclasse, além de atributos específicos, é mais especializada que a superclasse da qual foi derivada.

65. **(CESPE / MPU – 2010)** Em uma hierarquia de classes na qual exista herança múltipla, não é possível fazer uso do polimorfismo na implementação das classes.

66. **(CESPE / BASA – 2010)** Na modelagem de classes, a hierarquia entre elas é representada por meio de um relacionamento chamado generalização.

67. **(CESPE / TRE-BA – 2010)** O estado de um objeto é definido pelo conjunto de valores de suas propriedades.

68. **(CESPE / Banco da Amazônia – 2010)** O comportamento de um objeto é definido em sua respectiva classe, por meio da implementação de métodos que são executados quando tal objeto recebe uma mensagem.

69. **(CESPE / Banco da Amazônia – 2010)** A abstração permite, entre outras funcionalidades, identificar e compor objetos complexos e construir estruturas, na forma de classes de objetos, para organizar objetos de diferentes tipos. Porém, conceitos implementados por classes que são construídas com base na abstração não podem ser generalizados nem especializados.

70. **(CESPE / TRE-BA – 2010)** Em programação orientada a objetos, as propriedades que definem a estrutura e o comportamento de um objeto são especificadas para a classe da qual o objeto é instância e são válidas para todos os objetos dessa classe.

71. **(CESPE / ANAC – 2009)** O uso de mais de uma super classe imediata é usualmente denominado herança múltipla; ter somente uma super classe direta é denominado herança simples.

72. **(CESPE / DETRAN-DF – 2009)** Uma classe é uma descrição de um conjunto de objetos que compartilham os mesmos atributos, operações, relacionamentos e semântica.

73. **(CESPE / TRT-BA – 2008)** Os objetos permitem encapsular dados e funções, que modelam comportamentos e atributos, respectivamente.

74. **(CESPE / MPE-RR – 2008)** Na orientação a objetos, um objeto tipicamente possui estado e operações definidas. O estado é representado por atributos, e as operações associadas ao objeto podem fornecer serviços a outros objetos. Os objetos são criados de acordo com classes.

75. **(CESPE / PETROBRÁS – 2007)** Em um modelo de análise, as classes de fronteira modelam interações entre o sistema e os atores. Cada classe de fronteira deve estar relacionada a um ou mais atores. Pode-se também ter classes de entidade, as quais tipicamente modelam dados persistentes.

76. **(CESPE / PETROBRÁS – 2007)** Em um modelo de análise, as classes de controle podem encapsular controles relacionados a casos de uso e representar lógicas de negócio que não se relacionem a uma classe de entidade específica.

77. **(CESPE / PETROBRÁS – 2007)** Em um modelo de projeto, para que um subsistema seja coeso, seus conteúdos devem ser fortemente relacionados e, para que ele seja fracamente acoplado, é necessário que se minimizem as dependências entre subsistemas.

78. **(CESPE / ANATEL – 2006)** Uma classe na análise orientada a objeto representa uma abstração que pode ser mapeada para mais de uma classe no projeto. As classes na análise podem ser fronteiras, controladoras ou entidades. Uma fronteira modela interações entre o sistema e atores, uma entidade modela apenas objetos persistentes e uma controladora só pode controlar interações entre instâncias de uma mesma classe.

79. **(CESPE / TSE – 2006)** Um modelo de análise é menos abstrato que um de projeto e as classes em um modelo de análise não podem ser conceituais. As classes na análise podem modelar objetos persistentes, mas não transientes.

80. **(CESPE / TSE – 2006)** Uma importante responsabilidade da análise é definir a arquitetura do sistema, dividindo-o em subsistemas. Um subsistema expõe serviços via interfaces, que devem ser especificadas na análise.

81. **(CESPE / TSE – 2006)** Uma classe descreve objetos com as mesmas responsabilidades, relacionamentos, operações, atributos e semântica. As instâncias de uma classe têm, portanto, os mesmos valores para os seus atributos.

82. **(CESPE / SERPRO – 2006)** Uma das vantagens dos métodos de análise e projeto orientado a objetos é o aumento do gap conceitual entre os artefatos produzidos nas fases de análise, projeto e implementação.

83. **(CESPE / TRE-AL – 2004)** Uma hierarquia de classes é um mecanismo por meio do qual as modificações nos níveis inferiores da hierarquia se propagam de imediato para os níveis superiores.

84. **(CESPE / TRE-AL – 2004)** O polimorfismo ocorre quando uma subclasse herda atributos e operações de classes diferentes.

85. **(CESPE / STJ – 2004)** Com a análise orientada a objetos, busca-se identificar entidades do domínio do problema e caracterizá-las de acordo com sua importância para o problema. Essa atividade tem consequências nas etapas de projeto de software, uma vez que as entidades identificadas darão sustentação para a definição das classes de objetos a serem implementadas.

86. **(CESPE / STJ – 2004)** A definição da linguagem de programação a ser usada na implementação tem igual importância e impacto no projeto e na análise orientados a objetos.

GABARITO

1. CORRETO
2. ERRADO
3. CORRETO
4. LETRA D
5. CORRETO
6. CORRETO
7. CORRETO
8. CORRETO
9. ERRADO
10. ERRADO
11. ERRADO
12. ERRADO
13. CORRETO
14. ERRADO
15. ERRADO
16. LETRA A
17. CORRETO
18. ERRADO
19. CORRETO
20. CORRETO
21. ERRADO
22. ERRADO
23. CORRETO
24. ERRADO
25. LETRA A
26. LETRA A
27. LETRA A
28. CORRETO
29. CORRETO
30. ERRADO
31. ERRADO
32. ERRADO
33. CORRETO
34. CORRETO
35. ERRADO
36. CORRETO
37. LETRA D
38. ERRADO
39. ERRADO
40. ERRADO
41. CORRETO
42. CORRETO
43. CORRETO
44. ERRADO
45. CORRETO
46. CORRETO
47. CORRETO
48. ERRADO
49. CORRETO
50. CORRETO
51. CORRETO
52. CORRETO
53. ERRADO
54. CORRETO
55. LETRA A
56. CORRETO
57. ERRADO
58. ERRADO
59. CORRETO
60. CORRETO
61. CORRETO
62. ERRADO
63. CORRETO
64. CORRETO
65. ERRADO
66. CORRETO
67. CORRETO
68. CORRETO
69. ERRADO
70. CORRETO
71. CORRETO
72. CORRETO
73. ERRADO
74. CORRETO
75. CORRETO
76. CORRETO
77. CORRETO
78. ERRADO
79. ERRADO
80. ERRADO
81. ERRADO
82. ERRADO
83. ERRADO
84. ERRADO
85. CORRETO
86. ERRADO

LISTA DE QUESTÕES – FCC

1. **(FCC / METRÔ-SP – 2019)** Considere as seguintes situações:

   1. Um grupo foi formado por um conjunto de pessoas que têm vida própria, independente desse grupo.
   2. Uma equipe de TI é formada por um conjunto de programadores com dependência de vida dessa equipe.

   Na modelagem Orientação a Objetos com UML, essas situações são reconhecidas, respectivamente, como

   a) composição e associação todo-parte.
   b) composição e dependência funcional.
   c) composição todo-parte e dependência.
   d) associação todo-parte e composição.
   e) associação independente e composição.

2. **(FCC / TRF3 – 2019)** O Polimorfismo, um dos Pilares da Programação Orientada a Objetos - POO,

   a) ocorre quando uma classe tem um relacionamento do tipo "1 para" com outra classe e isso implica no modo como a definição das classes devem ocorrer nas aplicações.

   b) consiste em esconder os atributos da classe de quem for utilizá-la. Isso se deve a: 1 - para quem for usar a classe não a use de forma errada; e 2 - para que implementação seja feita por meio dos métodos get e set.

   c) permite que um mesmo método possa ter vários comportamentos e a definição de qual comportamento será executado se dá pelo valor diferente de um de seus atributos.

   d) é um conceito que permite que as características bem como as operações, de um modo global, possam ser repassadas para várias funcionalidades da aplicação.

   e) permite utilizar atributos e operações diferentes de uma subclasse, acrescentando ou substituindo características herdadas da classe pai.

3. **(FCC / SANASA Campinas – 2019)** Considere que um Analista de TI sabe que uma classe Pessoa Física e uma classe Pessoa Jurídica possuem o atributo nome como uma informação em comum e que o CPF é um atributo específico para a Pessoa Física e o CNPJ é um atributo específico para Pessoa Jurídica. Então o Analista criou uma outra classe com o atributo nome e seu objetivo é

que haja herança deste e, eventualmente, outros métodos e atributos, para as classes filhas, Pessoa Física e Pessoa Jurídica, que já existiam.

Essa classe criada não é instanciada, apenas fornece um modelo para geração de outras classes, e é denominada:

a) Subclasse.
b) Classe construtora.
c) Classe abstrata.
d) Classe sobrescrita.
e) Pacote.

4. **(FCC / SANASA Campinas – 2019)** Considere:

Os hidrômetros, relógios registradores de consumo de água, têm determinadas características. Em um sistema de computação, para processar os dados que deles provêm deve-se atentar para o fato que eles têm atributos e operações comuns e outros específicos. Usando pilares da orientação a objeto e a capacidade de reuso viabilizada por linguagens desse paradigma, um Analista usou dois conceitos fundamentais sendo um empregado no âmbito da descrição e estruturação das classes de hidrômetros e outro no âmbito da invocação dos métodos com mesma assinatura, todavia levando em consideração o comportamento distinto de operação dos hidrômetros. Tais conceitos são:

a) herança e visibilidade.
b) herança e polimorfismo.
c) composição e agregação.
d) agregação e polimorfismo.
e) visibilidade e composição.

5. **(FCC / TRT-SC – 2013)** Na programação orientada a objetos, as classes podem conter, dentre outros elementos, métodos e atributos. Os métodos:

a) devem receber apenas parâmetros do mesmo tipo.
b) não podem ser sobrecarregados em uma mesma classe.
c) precisam possuir corpo em interfaces e classes abstratas.
d) podem ser sobrescritos em aplicações que possuem relação de herança.
e) definidos como private só podem ser acessados de classes do mesmo pacote.

6. **(FCC / TRT-SC – 2013)** Na programação orientada a objetos, as classes podem conter, dentre outros elementos, métodos e atributos. Os métodos:

a) devem receber apenas parâmetros do mesmo tipo.
b) não podem ser sobrecarregados em uma mesma classe.
c) precisam possuir corpo em interfaces e classes abstratas.

d) podem ser sobrescritos em aplicações que possuem relação de herança.
e) definidos como private só podem ser acessados de classes do mesmo pacote.

7. **(FCC / AL-RN – 2013)** Um dos conceitos básicos de orientação a objetos é o fato de um objeto, ao tentar acessar as propriedades de outro objeto, deve sempre fazê-lo por uso de métodos do objeto ao qual se deseja atribuir ou requisitar uma informação, mantendo ambos os objetos isolados. A essa propriedade da orientação a objetos se dá o nome de:

a) herança.
b) abstração.
c) polimorfismo.
d) mensagem.
e) encapsulamento.

8. **(FCC / TRE-SP – 2012)** Nos conceitos de orientação a objetos, ..I... é uma estrutura composta por ...II... que descrevem suas propriedades e também por ...III.... que moldam seu comportamento. ....IV.... são ....V.... dessa estrutura e só existem em tempo de execução.

Para completar corretamente o texto as lacunas devem ser preenchidas, respectivamente, por

a) objeto, métodos, assinaturas, Classes, cópias.
b) polimorfismo, funções, métodos, Herança, cópias.
c) classe, atributos, operações, Objetos, instâncias.
d) multiplicidade, símbolos, números, Classes, herdeiros.
e) domínio, diagramas, casos de caso, Diagramas de classe, exemplos.

9. **(FCC / TJ-RJ – 2012)** No contexto de programação orientada a objetos, considere as afirmativas abaixo.

I. Objetos são instâncias de classes.
II. Herança é uma relação entre objetos.
III. Mensagens são formas de executar métodos.
IV. Classes são apenas agrupamentos de métodos.
V. Ocorre herança múltipla quando mais de um método é herdado.
VI. Herança é uma relação entre classes.

Está correto o que se afirma APENAS em:

a) I, III e IV.
b) I, III e VI.
c) III, IV e VI.
d) II, III e V.
e) II, IV e V.

10. (FCC / TRF2 – 2012) Sobre orientação a objetos é correto afirmar:

a) Na hierarquia de classes, se superclasse é uma generalização de subclasses, pode-se inferir que a subclasse é uma especialização de superclasse.

b) Numa árvore genealógica de classes, a classe mais baixa herda os atributos e métodos somente da superclasse no nível imediatamente acima.

c) As variáveis de uma classe só podem ser alteradas por métodos definidos nos seus objetos.

d) O polimorfismo se caracteriza quando, para mensagens distintas, objetos diferentes responderem ou agirem de forma idêntica.

e) Os objetos de uma classe são idênticos no que se refere à sua interface e ao seu estado.

11. (FCC / TRE-CE – 2012) Orientação a Objetos é um paradigma de análise, projeto e programação de sistemas de software. A respeito desse paradigma, assinale a afirmativa incorreta.

a) Um objeto pode ser considerado um conjunto de dados.
b) Os objetos possuem identidade, estado e comportamento.
c) Um evento pode existir se não houver um objeto a ele associado.
d) Um objeto pode existir mesmo que não exista nenhum evento associado a ele.
e) A orientação a objetos implementa o conceito de abstração, classe, objeto, encapsulamento, herança e polimorfismo.

12. (FCC / TJ-PE – 2012) Sobre orientação a objetos, considere:

I. A relação de herança permite modelar as similaridades inerentes a uma classe e também as diferenças especializadas que distinguem uma classe de outra.

II. Objetos com os mesmos atributos e operações possuem a mesma identidade, podendo ser referenciados por outros objetos.

III. A possibilidade de uma operação ter o mesmo nome, diferentes assinaturas e possivelmente diferentes semânticas dentro de uma mesma classe ou de diferentes classes é chamada de polimorfismo.

Está correto o que se afirma em:

a) I, II e III.
b) I e III, apenas.
c) II e III, apenas.
d) III, apenas.
e) II, apenas.

13. **(FCC / TRE-CE – 2012)** Sobre orientação a objetos, é INCORRETO afirmar:

a) os conceitos de generalização e especialização da orientação a objetos estão diretamente associados ao conceito de herança.

b) um objeto pode existir mesmo que não exista nenhum evento a ele associado.

c) um construtor visa inicializar os atributos e pode ser executado automaticamente sempre que um novo objeto é criado.

d) polimorfismo é o princípio pelo qual duas ou mais classes derivadas de uma mesma superclasse podem invocar métodos que têm a mesma assinatura e mesmo comportamento.

e) uma classe define o comportamento dos objetos através de seus métodos, e quais estados ele é capaz de manter através de seus atributos.

14. **(FCC / TCE-AM – 2012)** Sobre a orientação a objeto é correto afirmar:

a) Herança permite o reaproveitamento de atributos e métodos, porém, isso não altera o tempo de desenvolvimento, não diminui o número de linhas de código e não facilita futuras manutenções.

b) Em uma aplicação que utiliza herança múltipla, uma superclasse deve herdar atributos e métodos de diversas subclasses. Todas as linguagens de programação orientadas a objeto permitem herança múltipla.

c) O polimorfismo associado à herança trabalha com a redeclaração de métodos previamente herdados por uma classe. Esses métodos, embora semelhantes, diferem de alguma forma da implementação utilizada na superclasse, sendo necessário, portanto, reimplementá-los na subclasse.

d) A visibilidade protegida é representada pelo símbolo til (~) e significa que somente os objetos da classe detentora do atributo ou método poderão enxergá-lo ou utilizá-lo.

e) Em uma relação de herança é possível criar classes gerais, com características compartilhadas por muitas classes. Essas classes não podem possuir diferenças.

15. **(FCC / TST – 2012)** Considere que a classe Pessoa possui 3 métodos que podem ser aplicados aos seus objetos: cadastrar, alterar e excluir. Considere que Aluno e Professor são classes derivadas da classe Pessoa e, por isso, herdam os métodos cadastrar, alterar e excluir, mas estes métodos são sobrescritos na classe Aluno e Professor com implementações bastante distintas, em função dos dados associados a cada um deles. O exemplo ilustra o conceito de:

a) hereditariedade.
b) polimorfismo.
c) encapsulamento.
d) abstração.
e) reusabilidade.

16. (FCC / TRT-AM – 2012) Sobre Programação Orientada a Objetos, analise:

I. A encapsulação garante que apenas as interfaces necessárias para interação com o objeto estejam visíveis, e atributos internos não sejam acessíveis.

II. O polimorfismo garante que objetos possam herdar métodos e atributos de uma superclasse para a geração de uma nova classe.

III. A herança possibilita que distintas operações na mesma classe tenham o mesmo nome, desde que alterada a assinatura.

Está correto o que se afirma em:

a) III, apenas.
b) II, apenas.
c) I, apenas.
d) II e III, apenas.

17. (FCC / TST – 2012) Na orientação a objetos:

a) a herança permite que os membros de uma classe, chamada de classe-pai, possam ser reaproveitados na definição de outra classe, chamada de classe-filha. Esta classe-filha tem acesso aos membros públicos e protegidos da classe-pai. O polimorfismo, associado à herança, permite que métodos abstratos definidos em uma classe abstrata sejam implementados nas classes-filhas, podendo estes métodos, nas classes-filhas, apresentar comportamentos distintos.

b) atributos e métodos podem ser reaproveitados através da herança, quando uma subclasse herda as características de uma superclasse. Uma subclasse pode ter acesso aos membros de uma superclasse, independente do modificador atribuído. O polimorfismo é um recurso que permite a uma subclasse reimplementar os métodos herdados de uma superclasse, sendo este método abstrato ou não.

c) a herança e o polimorfismo são complementares, ou seja, devem ser aplicados em conjunto. A herança existe a partir de classes abstratas que contêm atributos e métodos abstratos. O polimorfismo obriga que as classes-filhas implementem os métodos e atributos desta classe-pai. O acesso aos atributos da classe-pai independe do modificador utilizado.

d) o conceito de herança estabelece que uma classe possa aproveitar a implementação, definições dos atributos e métodos de uma classe-base. A classe-filha pode ter acesso aos métodos e atributos públicos e protegidos da classe-base. O polimorfismo é aplicado ao caso em que existe a necessidade de implementar métodos sobrecarregados, nos quais a classe-filha necessita implementar dois métodos com o mesmo nome e parâmetros diferentes.

e) o polimorfismo é uma técnica que permite um objeto nascer a partir do uso de sobrecarga de construtores de uma classe, ou seja, o polimorfismo permite que um objeto possa ser instanciado de diferentes maneiras. A herança permite que uma classe sirva de base para que outras classes sejam implementadas. Entretanto, os membros com modificadores públicos da classe-base podem ser acessados pela classe-filha.

18. **(FCC / TRT11 – 2012)** No contexto de Programação Orientada a Objetos (OOP), sobre a relação de agregação e composição, ou relação todo-parte, considere:

I. A relação de agregação expressa o ato ou resultado de formar um objeto usando outros objetos como seus componentes.
II. Na relação de agregação, as partes só existem enquanto o todo existir.
III. Na relação de composição, as partes são independentes da existência do todo.

Está correto o que se afirma em:

a) I, apenas.
b) II, apenas.
c) II e III, apenas.
d) III, apenas.
e) I, II e III.

19. **(FCC / INFRAERO – 2011)** Sobre orientação a objetos, é correto afirmar:

a) Uma classe é o projeto do objeto. Ela informa à máquina virtual como criar um objeto de um tipo específico. Cada objeto criado a partir da classe terá os mesmos valores para as variáveis de instância da classe.

b) Um relacionamento de herança significa que a superclasse herdará as variáveis de instância e métodos da subclasse.

c) Uma interface é uma classe 100% abstrata, ou seja, uma classe que não pode ser instanciada.

d) Os objetos têm seu estado definido pelos métodos e seu comportamento definido nas variáveis de instância.

e) A principal regra prática do encapsulamento é marcar as variáveis de instância como públicas e fornecer métodos de captura e configuração privados.

20. (FCC / INFRAERO – 2011) Sobre a programação orientada a objetos, analise:

I. Neste tipo de programação, objetos executam ações, mas não suportam propriedades ou atributos.
II. Uma classe especifica o formato geral de seus objetos.
III. As propriedades e ações disponíveis para um objeto não dependem de sua classe.
IV. A tecnologia orientada a objetos permite que classes projetadas adequadamente sejam reutilizáveis em vários projetos.

Está correto o que consta em:

a) II, III e IV, apenas.
b) I e II, apenas.
c) II e IV, apenas.
d) I, II e III, apenas.
e) I, II, III e IV.

21. (FCC / TRT-RS – 2011) O aumento da produtividade de desenvolvimento e a capacidade de compartilhar o conhecimento adquirido, representa uma vantagem no uso de projetos orientados a objeto, porque:

a) um objeto pode ser chamado por objetos de classe diferente da sua.
b) os objetos podem ser potencialmente reutilizáveis.
c) as classes podem ser concretas ou abstratas.
d) todo método pode ser derivado naturalmente das operações de sua classe.
e) o encapsulamento impossibilita equívocos de código.

22. (FCC / TRT14 – 2011) Considere:

I. Uma classe define o comportamento dos objetos através de seus métodos, e quais estados ele é capaz de manter através de seus atributos.

II. Na herança cada classe derivada (subclasse) apresenta as características (estrutura e métodos) da classe base (superclasse) e acrescenta a elas o que for definido de particularidade para ela.

III. Polimorfismo é o princípio pelo qual duas ou mais classes derivadas de uma mesma superclasse podem invocar métodos que têm a mesma identificação e mesmo comportamento.

IV. Um objeto é capaz de armazenar estados através de seus atributos e reagir a mensagens enviadas a ele, assim como se relacionar e enviar mensagens a outros objetos.

Na orientação a objetos é correto o que se afirma em:

a) I, II e III, apenas.
b) I, II e IV, apenas.
c) I, III e IV, apenas.
d) II, III e IV, apenas.
e) I, II, III e IV.

23. **(FCC / TRT-MT – 2011)** Sobre os conceitos de orientação a objetos, considere:

I. Classe encapsula dados para descrever o conteúdo de alguma entidade do mundo real.
II. Objetos são instâncias de uma classe que herdam os atributos e as operações da classe.
III. Superclasse é uma especialização de um conjunto de classes relacionadas a ela.
IV. Operações, métodos ou serviços fornecem representações dos comportamentos de uma classe.

Está completo e correto o que consta em:

a) I, II, III e IV.
b) I, II e IV, apenas.
c) II, III e IV, apenas.
d) I e II, apenas.
e) II e IV, apenas.

24. **(FCC / CAIXA – 2011)** Na orientação a objetos, é um recurso que serve para inicializar os atributos e é executado automaticamente sempre que um novo objeto é criado:

a) método.
b) polimorfismo.
c) interface.
d) classe.
e) construtor.

25. **(FCC / TRE-RN – 2011)** Método especial destinado ao preparo de novos objetos durante sua instanciação. Pode ser acionado por meio do operador new, recebendo parâmetros como métodos comuns, o que permite caracterizar os objetos já na instanciação. Trata-se de:

a) operação polimórfica.
b) construtor.
c) atributo.
d) herança polimórfica.
e) herança múltipla.

26. **(FCC / CAIXA – 2011)** Na programação orientada a objetos, subprogramas (ou subrotinas) são encapsuladas nos próprios objetos e passam a designar-se:

a) atributo.
b) herança.
c) instância.
d) método.
e) encapsulamento.

27. **(FCC / CAIXA – 2011)** Objetos se comunicam por passagem de mensagem, eliminando áreas de dados compartilhados.

28. **(FCC / TCE-PR – 2011)** Em relação à Programação Orientada a Objetos, é INCORRETO afirmar:

a) Polimorfismo pode ser entendido como um conceito complementar ao de herança. Assim, no polimorfismo é possível enviar a mesma mensagem a diferentes objetos e cada objeto responder da maneira mais apropriada para sua classe.

b) Uma agregação representa um todo que é composto de várias partes e constitui um relacionamento de contenção; se qualquer uma das partes for destruída, as demais partes também o serão.

c) Interfaces são como as classes abstratas, mas nelas não é possível implementar nenhum método, apenas declarar suas assinaturas; uma classe ao implementar uma interface deverá escrever todos os seus métodos.

d) No contexto da herança, uma instância da subclasse é, também, uma instância da superclasse.

e) A aplicação do polimorfismo utilizando interfaces requer que o método polimórfico seja definido na classe ancestral como abstract para possibilitar sua redefinição nas classes descendentes.

29. **(FCC / CAIXA – 2011)** Um detalhe importante que deve ser especificado para os atributos e operações das classes é a visibilidade. Desta forma, os símbolos: + (sinal de mais), # (sinal de número), - (sinal de menos) e ~ (til) correspondem respectivamente a:

a) público, pacote, privado e protegido.
b) público, protegido, privado e pacote.
c) privado, protegido, público e pacote.
d) privado, pacote, público e protegido.
e) pacote, protegido, privado e público.

30. **(FCC / TRT-MS – 2011)** Propriedade pela qual duas ou mais classes derivadas de uma mesma superclasse podem invocar métodos que têm a mesma assinatura mas comportamentos distintos. Trata-se de:

a) polimorfismo.
b) herança múltipla.
c) operação agregada.
d) multiplicidade.
e) visibilidade.

31. **(FCC / TRT14 – 2011)** A classe Veiculo contém alguns atributos de interesse da classe Aeronave. Todavia, as aeronaves também demonstram interesse em captar atributos e também operações da classe Elemento Turbinado. O enunciado enfatiza o conceito OO de:

a) polimorfismo.
b) herança múltipla.
c) dependência funcional.
d) realização.
e) encapsulamento.

32. **(FCC / TRT-RS – 2011)** Na taxonomia utilizada para as formas de polimorfismo são, respectivamente, dois tipos categorizados como universal e dois como Ad Hoc:

a) Paramétrico e Inclusão; Sobrecarga e Coerção.
b) Paramétrico e Coerção; Sobrecarga e Inclusão.
c) Paramétrico e Sobrecarga; Inclusão e Coerção.
d) Sobrecarga e Inclusão; Paramétrico e Coerção.
e) Sobrecarga e Coerção; Paramétrico e Inclusão.

33. **(FCC / TRE-CE – 2011)** Sobre conceitos em programação orientada a objetos (OOP), analise:

I. No polimorfismo ad-hoc, métodos com o mesmo nome e pertencentes à mesma classe, podem receber argumentos distintos, consequentemente alterando a assinatura do método.

II. No polimorfismo paramétrico é possível determinar o método como atributos de objetos são acessados por outros objetos, protegendo o acesso direto aos mesmos através de operações.

III. Na restrição de multiplicidade é possível determinar o número de atributos e operações que uma classe pode herdar de uma superclasse.

Está correto o que consta em:

a) I, II e III.
b) I, apenas.
c) III, apenas.
d) II e III, apenas.
e) I e II, apenas.

34. **(FCC / DPE-SP – 2010)** Classes e objetos são dois conceitos-chave da programação orientada a objetos. Com relação a estes conceitos, é correto afirmar que:

a) uma classe é uma descrição de um ou mais objetos por meio de um conjunto uniforme de atributos e serviços. Além disso, pode conter uma descrição de como criar novos objetos na classe.

b) uma classe é capaz de armazenar estados através de seus atributos e reagir a mensagens enviadas a ela, assim como se relacionar e enviar mensagens a outras classes.

c) uma classe é uma abstração de alguma coisa no domínio de um problema ou na sua implementação, refletindo a capacidade de um sistema para manter informações sobre ela, interagir com ela ou ambos.

d) um objeto em uma classe é apenas uma definição, pois a ação só ocorre quando o objeto é invocado através de um método.

e) herança é o mecanismo pelo qual um objeto pode estender outro objeto, aproveitando seus comportamentos e variáveis possíveis.

35. **(FCC / TCE-SP – 2010)** A descrição de um conjunto de entidades (reais ou abstratas) de um mesmo tipo e com as mesmas características e comportamentos. Trata-se da definição de:

a) String.
b) Método.
c) Conjunto.
d) Classe.
e) Objeto.

36. **(FCC / DPE-SP – 2010)** A cidade de São Paulo, que possuía uma população de 10.000.000 de habitantes, teve um aumento de mais 2.000.000 de novos habitantes. Na associação da frase acima aos conceitos da modelagem orientada a objeto, é correto afirmar que São Paulo, população e aumento, referem-se, respectivamente, a:

a) classe, objeto, instância de classe.
b) objeto, atributo, implementação por um método do objeto.
c) classe, objeto, atributo.
d) objeto, instância, operação.
e) classe, objeto, associação pelo método de agregação.

37. **(FCC / TRE-RS – 2010)** Um objeto é, na orientação a objetos,

a) uma rotina de programação contida em uma classe que pode ser chamada diversas vezes possibilitando assim reuso de código de programação.

b) um conjunto de atributos primitivos tipados contido em uma classe.

c) uma entidade que possui um estado e um conjunto definido de operações definidas para funcionar nesse estado.

d) um elemento de uma classe que representa uma operação (a implementação de uma operação).

e) uma porção de código que resolve um problema muito específico, parte de um problema maior.

38. **(FCC / TRT-PI – 2010)** Em relação à orientação a objetos, considere as assertivas abaixo.

I. Um objeto pode ser real ou abstrato. Sendo uma instância de uma classe, possui informações e desempenha ações.

II. Uma classe especifica uma estrutura de dados e os métodos operacionais permissíveis que se aplicam a cada um de seus objetos. Pode ter sua própria estrutura de dados e métodos, bem como pode herdá-la de sua superclasse.

III. Todas as características de uma superclasse são reusáveis por aquelas classes que são seus subtipos. Assim, uma superclasse é um supertipo de uma ou mais classes.

IV. No polimorfismo duas ou mais classes derivadas de uma mesma superclasse podem invocar métodos que têm a mesma identificação, mas comportamentos distintos, especializados para cada classe derivada.

É correto o que se afirma em

a) I, II, III, apenas.
b) I, II e IV, apenas.
c) I, III e IV, apenas.
d) II, III e IV, apenas.
e) I, II, III e IV.

39. **(FCC / SEFAZ-SP – 2010)** Os valores das propriedades de um objeto em um determinado instante, que podem mudar ao longo do tempo, representam:

a) a instância de uma classe.
b) a identidade de um objeto.
c) o estado de um objeto.

d) o comportamento de um objeto.
e) as operações de uma classe.

40. (FCC / TCM-PA – 2010) Não possui instâncias diretas, mas apenas classes descendentes:

a) a classe concreta.
b) o objeto.
c) a classe abstrata.
d) o caso de uso de inclusão.
e) o pacote.

41. (FCC / TRT-PR – 2010) Uma técnica que consiste em separar aspectos externos dos internos da implementação de um objeto, isto é, determinados detalhes ficam ocultos aos demais objetos e dizem respeito apenas ao próprio objeto. Trata-se de:

a) polimorfismo.
b) generalização.
c) encapsulamento.
d) herança.
e) visibilidade.

42. (FCC / MPE-RN – 2010) Uma operação pode ter implementações diferentes em diversos pontos da hierarquia de classes, desde que mantenham a mesma assinatura. Na orientação a objetos, este é o conceito que embasa:

a) a multiplicidade.
b) o encapsulamento.
c) o protótipo.
d) o polimorfismo.
e) o estereótipo.

43. (FCC / Sergipe Gás SA – 2010) "É o mecanismo pelo qual uma classe pode estender outra classe, aproveitando seus comportamentos e variáveis possíveis." Na programação orientada a objetos esta afirmação refere-se aos conceitos essenciais de:

a) herança, métodos e atributos.
b) subclasse, instância e associação.
c) subclasse, encapsulamento e abstração.
d) herança, abstração e associação.
e) encapsulamento, polimorfismo e interface.

44. (FCC / TJ-SE – 2009) Na programação orientada a objetos, são características dos objetos:

a) As classes, os métodos e as mensagens.

b) A identidade, os atributos e as operações.
c) O encapsulamento, a herança e o polimorfismo.
d) A instanciação, a generalização e a especialização.
e) A classificação, a composição e a decomposição.

45. **(FCC / SEFAZ-SP – 2009)** Uma classe é uma abstração que ajuda a lidar com a complexidade e um bom exemplo de abstração é:

a) um aluno e as disciplinas que está cursando.
b) um professor e os cursos nos quais ministra aulas.
c) um funcionário e o departamento em que trabalha.
d) uma pessoa e o número do seu CPF na Receita Federal.
e) uma casa e a empresa que a projetou e construiu.

46. **(FCC / SEFAZ-SP – 2009)** Na orientação a objetos, ao nível de classe, são definidos os:

a) atributos e os valores dos atributos.
b) atributos e a invocação das operações.
c) atributos e os métodos.
d) métodos e os valores dos atributos.
e) métodos e a invocação das operações.

47. **(FCC / SEFAZ-SP – 2009)** O método utilizado para inicializar objetos de uma classe quando estes são criados é denominado:

a) void.
b) interface.
c) agregação.
d) composição.
e) construtor.

48. **(FCC / TJ-PI – 2009)** Na programação orientada a objetos, é o princípio que oferece a capacidade de um método poder ser implementado de diferentes formas, ou mesmo de realizar coisas diferentes, ou seja, um único serviço pode oferecer variações, conforme se aplique a diferentes subclasses de uma superclasse. No contexto, o termo método é:

a) o mecanismo pelo qual um objeto utiliza os recursos de outro.
b) uma instância de uma classe.
c) o elemento que define as habilidades do objeto.
d) uma chamada a um objeto para invocar uma classe.
e) um objeto capaz de armazenar estados através de seus atributos.

49. **(FCC / PGE-RJ – 2009)** Sobre orientação a objetos, considere:

I. Os valores dos atributos são definidos no nível de classe.
II. Os métodos são definidos no nível de objeto.
III. A invocação de uma operação é definida no nível de objeto.

Está correto o que se afirma em:

a) II e III, apenas.
b) I, II e III.
c) III, apenas.
d) I e II, apenas.
e) I e III, apenas.

50. (FCC / TJ-PA – 2009) A especificação de uma comunicação entre objetos, que contém informações relacionadas ao que se espera resultar dessa atividade, é:

a) uma restrição.
b) uma mensagem.
c) uma operação.
d) um processo oculto.
e) um diálogo.

51. (FCC / TRE-PI – 2009) A afirmação de que o estado de um objeto não deve ser acessado diretamente, mas sim por meio de métodos de acesso, está associada ao conceito de encapsulamento.

52. (FCC / SEFAZ-SP – 2009) Sobre a visibilidade dos métodos na orientação a objetos considere:

I. Os métodos públicos de uma classe definem a interface da classe.
II. Os métodos privativos de uma classe não fazem parte da interface da classe.
III. O nome dos métodos é a informação reconhecida como a assinatura dos métodos.

Está correto o que consta APENAS em:

a) I e II.
b) I e III.
c) II e III.
d) II.
e) I.

53. (FCC / MPE-SE – 2009) "A utilização de um sistema orientado a objetos não deve depender de sua implementação interna, mas de sua interface." Esta afirmação remete ao conceito de:

a) herança múltipla.
b) herança polimórfica.

c) prototipação.
d) encapsulamento.
e) especialização.

54. **(FCC / TRT-CE – 2009)** Considere: A classe Pedido contém um método chamado obter Produtos() que retorna uma lista de produtos pertencentes a um determinado pedido. O código que usa esta classe desconhece completamente como esta lista de produtos é montada. Tudo que interessa é a lista de produtos que o método retorna. Na essência, o texto explica um dos fundamentos das linguagens OO que é:

a) polimorfismo.
b) encapsulamento.
c) dependência.
d) herança múltipla.
e) estereotipagem.

55. **(FCC / MPE-SE – 2009)** *"...distintas implementações de uma operação de classe e que, no entanto, o nome e os parâmetros dessa operação sejam os mesmos"*. Trata-se de:

a) objeto persistente.
b) enumeração.
c) polimorfismo.
d) subclasse.
e) pseudo-estado.

56. **(FCC / TJ-PI – 2009)** Na programação orientada a objetos, é o princípio que oferece a capacidade de um método poder ser implementado de diferentes formas, ou mesmo de realizar coisas diferentes, ou seja, um único serviço pode oferecer variações, conforme se aplique a diferentes subclasses de uma superclasse. O texto acima trata do Princípio de:

a) Polimorfismo.
b) Reutilização.
c) Abstração.
d) Herança.
e) Encapsulamento.

57. **(FCC / TRT-MA – 2009)** Um analista desenvolveu métodos de impressão de dados com a mesma assinatura para três classes de impressoras (jato de tinta, laser e matricial) derivadas de uma mesma superclasse impressora. Tal prática:

a) aplica o conceito de herança múltipla.
b) aplica o conceito de polimorfismo.
c) constitui-se em ferimento à regra de herança.
d) visa ao aumento da coesão entre os atributos da superclasse.

e) não é recomendada na orientação a objetos.

58. **(FCC / PGE-RJ – 2009)** Um comando "abrir" ao provocar diferentes ações em objetos distintos, por exemplo: em uma caixa, porta ou janela, representa figurativamente na orientação a objetos o princípio denominado:

a) persistência.
b) polimorfismo.
c) abstração.
d) agregação.
e) herança.

59. **(FCC / SEFAZ-SP – 2009)** Compartilhamento de atributos e operações genéricas entre diversas classes descendentes de uma classe ancestral remete ao conceito de:

a) cardinalidade.
b) encapsulamento.
c) herança.
d) agregação.
e) multiplicidade.

60. **(FCC / PGE-RJ – 2009)** O conceito de Herança, na orientação a objetos, está especificamente associado ao significado de:

a) cardinalidade.
b) generalização.
c) multiplicidade.
d) encapsulamento.
e) composição.

61. **(FCC / TCE-AL – 2008)** Considere: *Casas ABC Ltda., Empresa e Nome da Empresa.* Na orientação a objetos, os itens acima representam, respectivamente,

a) atributo, classe e objeto.
b) classe, atributo e objeto.
c) classe, objeto e atributo.
d) objeto, atributo e classe.
e) objeto, classe e atributo.

62. **(FCC / TRT18 – 2008)** São dois tipos de relacionamento todo-parte:

a) agregação e composição.
b) generalização e composição.
c) generalização e especialização.

d) composição e dependência.
e) especialização e agregação.

63. **(FCC / TCE-AL – 2008)** Os conceitos de generalização e especialização da orientação a objetos estão diretamente relacionados ao conceito de:

a) Agregação.
b) Associação.
c) Encapsulamento.
d) Polimorfismo.
e) Herança.

64. **(FCC / TRF4 – 2007)** A proteção de atributos e operações das classes, fazendo com que estas se comuniquem com o meio externo por meio de suas interfaces, define o conceito de:

a) polimorfismo.
b) encapsulamento.
c) herança.
d) agregação.
e) especialização.

GABARITO

1. LETRA D
2. LETRA C
3. LETRA C
4. LETRA B
5. LETRA D
6. LETRA D
7. LETRA E
8. LETRA C
9. LETRA B
10. LETRA A
11. LETRA C
12. ERRADO
13. LETRA D
14. LETRA C
15. LETRA B
16. LETRA C
17. LETRA A
18. LETRA A
19. LETRA C
20. LETRA C
21. LETRA B
22. CORRETO
23. LETRA E
24. LETRA E
25. LETRA B
26. LETRA D
27. CORRETO
28. ANULADA
29. LETRA B
30. LETRA A
31. LETRA B
32. LETRA A
33. LETRA B
34. LETRA A
35. LETRA D
36. LETRA B
37. LETRA C
38. LETRA E
39. LETRA C
40. LETRA C
41. LETRA C
42. LETRA D
43. LETRA A
44. LETRA B
45. LETRA D
46. LETRA C
47. LETRA E
48. LETRA C
49. LETRA C
50. LETRA B
51. CORRETO
52. LETRA A
53. LETRA D
54. LETRA B
55. LETRA C
56. LETRA A
57. LETRA B
58. LETRA B
59. LETRA C
60. LETRA B
61. LETRA E
62. LETRA A
63. LETRA E
64. LETRA B

LISTA DE QUESTÕES – FGV

1. **(FGV / AL-SC – 2024)** Considerando o paradigma da orientação a objetos, identifique os quatro pilares mestres que norteiam o fundamento da tecnologia.

   a) Classe, Abstração, Coesão e Encapsulamento.
   b) Variáveis, Classe, Objeto e Método.
   c) Agregação, Acoplamento, Colaboração e Especificação.
   d) Classe, Objeto, Atributo e Método
   e) Acoplamento, Coesão, Colaboração e Encapsulamento.

2. **(FGV / SEFAZ-MG – 2023)** Os padrões de projeto de software (design patterns) tiram proveito máximo dos pilares da orientação a objetos. Usemos como exemplo o padrão Abstract Factory, que é um padrão de projeto criacional que permite produzir famílias de objetos relacionados sem especificar suas classes concretas. O principal pilar da orientação a objetos usado nesse design pattern é:

   a) o polimorfismo.
   b) a herança.
   c) o encapsulamento.
   d) a abstração.
   e) a sublevação.

3. **(FGV / SEAD-AP – 2022)** No contexto da orientação por objetos, o princípio pelo qual duas ou mais classes derivadas de uma mesma classe podem incorporar métodos que têm a mesma assinatura original, porém com comportamentos especializados, é conhecido como:

   a) abstração.
   b) encapsulamento.
   c) herança.
   d) interfaceamento.
   e) polimorfismo.

4. **(FGV / TCE-TO – 2022)** O sistema de controle automotivo SisAut foi modelado orientado a objetos. O SisAut possui a classe Veículo, que compartilha seu código-fonte com suas subclasses: Carro e Moto. As subclasses Carro e Moto aproveitam os métodos e atributos da classe Veículo.

   Em orientação a objeto, o mecanismo pelo qual uma classe pode estender outra classe ou ser estendida por outra classe é:

   a) interface;
   b) herança;

c) pacotes;
d) agregação;
e) encapsulamento.

5. **(FGV / IBGE – 2016)** Em Orientação a Objetos, para que uma subclasse de uma classe possa ter seu próprio comportamento, e mesmo assim compartilhar algumas das funcionalidades da classe pai, deve-se implementar:

a) generalização;
b) agregação;
c) abstração;
d) composição;
e) polimorfismo.

6. **(FGV / CODEBA – 2016)** Durante a fase de análise de um sistema que está sendo desenvolvido sob o paradigma de orientação a objetos, o analista Pedro quer representar, em um diagrama de classes, que uma turma é formada por alunos. Os alunos, porém, também podem ser considerados individualmente no sistema, independente da turma.

Para representar a relação entre alunos e turma, Pedro deve utilizar:

a) composição.
b) agregação.
c) herança.
d) encapsulamento.
e) atributo classe.

7. **(FGV / TCE-SE – 2015)** Em POO (Programação Orientada a Objetos), dizer que a classe A estende a classe B é o mesmo que dizer que:

a) a classe B é subclasse de A;
b) a classe A é superclasse de B;
c) a classe A é derivada de B;
d) a classe B é derivada de A;
e) as classes A e B são irmãs.

8. **(FGV / TCE-SE – 2015)** Em POO (programação orientada a objetos), dizer que a classe A é superclasse de B é o mesmo que dizer que:

a) A é derivada de B;
b) A estende B;
c) B é derivada de A;
d) B implementa A;
e) A implementa B.

9. **(FGV / PROCEMPA – 2014)** Definir a responsabilidade de cada classe é um aspecto muito importante que deve ser observado durante a modelagem de um projeto de sistema de software. Em relação aos princípios essenciais de boas práticas de modelagem orientada a objeto assinale a afirmativa correta.

a) O encapsulamento e a adoção de interfaces permitem diminuir o acoplamento e a coesão entre classes.

b) Cada classe deve assumir uma única responsabilidade bem definida como meio de aumentar a coesão e assegurar o encapsulamento.

c) As responsabilidades de uma classe devem estar relacionadas entre si para facilitar o entendimento e aumentar as chances de sua reutilização.

d) Para promover a flexibilidade e a facilidade de manutenção a longo prazo, as classes devem ser fortemente acopladas e encapsuladas.

e) Minimizar o acoplamento implica em classes com alta coesão.

10. **(FGV / TJ-GO – 2014)** Analise as afirmativas a seguir, no contexto das linguagens de programação orientadas a objetos:

I. A herança múltipla é a possibilidade de uma classe estender uma ou mais classes simultaneamente.

II. A herança múltipla é a possibilidade de uma classe implementar uma ou mais inferfaces simultaneamente.

III. A herança múltipla é a possibilidade de, numa dada classe, coexistirem métodos homônimos com múltiplas assinaturas, desde que distintas.

É verdadeiro somente o que se afirma em:

a) I;
b) II;
c) III;
d) I e II;
e) II e III.

11. **(FGV / TJ-AM – 2013)** No que diz respeito à programação orientada a objetos, um recurso refere-se ao poder que os objetos de classes distintas têm de invocar um mesmo método e obter comportamento diferente. Esse recurso é conhecido por:

a) polimorfsimo.
b) encapsulamento.
c) abstração.
d) herança.
e) coesão.

12. **(FGV / FIOCRUZ – 2010)** Num banco de dados orientado a objetos, a informação é armazenada na forma de objetos. Este tipo de banco de dados possui três bases principais. Assinale a alternativa que atende ao paradigma de Orientação a Objetos.

a) Herança; isomorfismo; multilateralidade.
b) Herança; polimorfismo; encapsulamento.
c) Identidade; isomorfismo; armazenamento.
d) Geomorfismo; identidade; herança.
e) Identidade; classificação; multilateralidade.

13. **(FGV / SEAD-AP - 2010)** Em conformidade com a metodologia orientada a objetos, com a finalidade de evitar que partes de um programa se tornem tão independentes que uma pequena alteração tenha grandes efeitos em cascata, é aplicado um recurso que separa os aspectos externos e acessíveis de um objeto dos detalhes internos de implementação.

Esse recurso utiliza um princípio da Orientação a Objetos que propõe ocultar determinados elementos de uma classe das demais classes. O objetivo ao colocar uma proteção ao redor é prevenir contra os efeitos colaterais indesejados ao ter essas propriedades modificadas de forma inesperada.

Este recurso é conhecido por:

a) coesão.
b) acoplamento.
c) polimorfismo.
d) modularidade.
e) encapsulamento.

14. **(FGV / MEC - 2009)** Na Análise Orientada a Objetos, o princípio pelo qual duas ou mais classes derivadas de uma mesma superclasse podem invocar métodos que têm a mesma identificação (assinatura) mas comportamentos distintos, especializados para cada classe derivada, usando para tanto uma referência a um objeto do tipo da superclasse é denominado:

a) encapsulamento.
b) independência.
c) modularidade.
d) polimorfismo
e) herança.

GABARITO

1. LETRA D
2. LETRA D
3. LETRA E
4. LETRA B
5. LETRA E
6. LETRA B
7. LETRA C
8. LETRA C
9. LETRA C
10. LETRA A
11. LETRA A
12. LETRA B
13. LETRA E
14. LETRA D

LISTA DE QUESTÕES – DIVERSAS BANCAS

1. **(CESGRANRIO / IPEA – 2024)** Em um diagrama de classes criado por uma equipe, há uma classe P que possui um relacionamento de associação com a classe Q. Qual situação, em código, representa, de maneira adequada, tal ideia de relacionamento entre essas classes?

   a) A classe P possui um método que instancia um objeto da classe Q.
   b) A classe P possui um método que recebe como parâmetro um objeto da classe Q.
   c) A classe P possui uma propriedade de objeto da classe Q.
   d) A classe P herda da classe Q.
   e) A classe Q herda da classe P.

2. **(CESGRANRIO / TRANSPETRO – 2023)** Um desenvolvedor pretende criar três métodos com o mesmo nome em uma mesma classe, mas com parâmetros diferentes. Para essa tarefa, utilizará uma técnica que faz parte do paradigma de orientação a objetos. Qual o nome da técnica a ser usada pelo desenvolvedor?

   a) Herança
   b) Interface
   c) Instanciação
   d) Sobrecarga
   e) Sobrescrita

3. **(VUNESP / TCM-SP - 2023)** Considere que, na modelagem de dados utilizando a orientação a objetos, há três classes: a classe Filme e as classes Suspense e Comédia, ambas herdando as características da classe Filme. Portanto, está-se fazendo uma aplicação do conceito de herança, segundo o qual:

   a) a classe Filme assume o papel de uma subclasse, e as classes Suspense e Comédia assumem o papel de superclasses.

   b) a classe Filme só pode ter atributos do tipo literal, enquanto que as classes Suspense e Comédia só podem conter atributos do tipo numérico.

   c) as classes Suspense e Comédia herdam atributos e métodos da classe Filme, podendo, por exemplo, acrescentar atributos peculiares à classe Filme.

   d) as classes Suspense e Comédia assumem exatamente os mesmos métodos e atributos da classe Filme, sem poder alterar qualquer um deles.

   e) as quantidades de objetos oriundos das classes Suspense e Comédia devem ser exatamente as mesmas.

4. **(ACCESS / Câmara de Mangaratiba-RJ – 2020)** No que diz respeito à Orientação a Objetos, dois princípios são caracterizados a seguir:

I. faz com que detalhes internos do funcionamento dos métodos de uma classe permaneçam ocultos para os objetos. O conhecimento a respeito da implementação interna da classe é desnecessário do ponto de vista do objeto, uma vez que isso passa a ser responsabilidade dos métodos internos da classe.

II. indica a capacidade de abstrair várias implementações diferentes em uma única interface. As classes derivadas de uma única classe base são capazes de invocar os métodos que, embora apresentem a mesma assinatura, comportam-se de maneira diferente para cada uma das classes derivadas.

Os princípios caracterizados em I e II são respectivamente denominados

a) encapsulamento e coesão.
b) acoplamento e coesão.
c) encapsulamento e acoplamento.
d) acoplamento e polimorfismo.
e) encapsulamento e polimorfismo.

5. **(IBFC / TRE-PA – 2020)** Assinale a alternativa que identifica incorretamente um conceito básico das linguagens orientadas a objetos.

a) interface
b) herança
c) encapsulamento
d) subprogramação

6. **(IBFC / Prefeitura de Cruzeiro do Sul - AC – 2019)** Um dos conceitos do paradigma orientado a objetos consiste na alteração do funcionamento interno de um método herdado de um objeto pai. Assinale a alternativa correta que apresenta este conceito.

a) Polimorfismo
b) Pai e filho
c) Encapsulamento
d) Abstração

7. **(CONSULPLAN/ Prefeitura de Suzano - SP – 2019)** Na orientação a objetos, todo objeto está relacionado a uma classe que o representa e que serve como forma de modelo. O objeto terá atributos e métodos definidos na classe. Polimorfismo se refere a:

a) Classes com vários métodos com o mesmo nome, mas com parâmetros diferentes.

b) Mecanismo que permite definir uma nova classe (subclasse) a partir de uma já existente (superclasse).
c) Classes com vários tipos diferentes de objetos instanciados; porém, cada um com sua própria variável.
d) Superclasses e classes relacionadas, com interfaces idênticas; porém, com implementações diferentes.

8. **(AOCP / IBGE – 2019)** As classes, bem como os seus objetos, contêm atributos e métodos que estão intimamente relacionados. Os objetos podem se comunicar entre si, mas eles, em geral, não sabem como outros objetos são implementados, uma vez que os detalhes de implementação permanecem ocultos dentro dos próprios objetos. Assinale a alternativa que apresenta corretamente o nome desse ocultamento de informações, crucial na boa prática da engenharia de software.

a) Herança.
b) Recursão.
c) Instanciação.
d) Polimorfismo.
e) Encapsulamento.

9. **(QUADRIX / Prefeitura de Jataí - GO – 2019)** Na análise orientada a objetos, o diagrama que descreve os tipos de objetos e seus relacionamentos, descreve a estrutura estática de um sistema, isto é, descreve como o sistema é estruturado, e não como ele se comporta, é o diagrama de:

a) classe.
b) pacotes.
c) colaboração.
d) estados.
e) atividades.

10. **(QUADRIX / Prefeitura de Jataí - GO – 2019)** No que se refere aos conceitos de herança múltipla, julgue os itens subsequentes.

I. Com a implementação da herança múltipla, é possível simplificar os programas e proporcionar soluções para resolver problemas difíceis.
II. Ocorre herança múltipla quando uma classe herda de mais de uma classe, ou seja, existem múltiplas classes-base (pais) para a classe derivada (filha).
III. Na herança múltipla, uma classe herda apenas a estrutura, e não o comportamento de mais de uma classe-base.

a) Apenas o item I está certo.
b) Apenas o item II está certo.
c) Apenas os itens I e II estão certos.

d) Apenas os itens I e III estão certos.
e) Apenas os itens II e III estão certos.

11. **(QUADRIX / CRO-GO – 2019)** Alguns conceitos como herança, polimorfismo e encapsulamento são abordados no desenvolvimento de sistemas orientados a objetos.

12. **(COVEST-COPSET / UFPE – 2019)** No contexto dos principais elementos conceituais do modelo de objetos, assinale a alternativa que define corretamente o conceito de encapsulamento.

a) É a capacidade de uma abstração referenciar outro tipo de dados abstrato, reutilizando comportamento e estabelecendo uma hierarquia de tipos que permite o reuso.

b) É o processo de compartimentalizar os elementos de uma abstração que constituem sua estrutura e comportamento, servindo para separar a interface contratual de uma abstração e sua implementação.

c) Refere-se à capacidade de salvar atributos de um objeto em um meio permanente de persistência, tais como bancos de dados ou arquivos, possibilitando que objetos possam reutilizar o seu estado em diferentes execuções de um mesmo programa.

d) Consiste em dividir um programa em módulos que podem ser compilados separadamente e ter conexões com outros módulos.

e) Denota as maneiras pelas quais um objeto pode agir e reagir, constituindo toda a visão externa estática e dinâmica da abstração.

13. **(CESGRANRIO / UNIRIO – 2019)** Em orientação a objetos, uma classe abstrata é uma classe que:

a) possui apenas métodos estáticos.
b) não pode ser instanciada.
c) não possui métodos.
d) não possui variáveis de instância.
e) não pode ter subclasses.

14. **(IF-PE / IF-PE – 2019)** Sobre o uso de interfaces em orientação a objetos, podemos afirmar que:

I. evita que alterações de código em determinados componentes do sistema sejam refletidas por todo o sistema.
II. representa um contrato entre componentes do sistema.
III. permite a utilização das implementações das classes concretas ao invés da utilização das classes abstratas.

Está(ão) CORRETA(S), apenas, a(s) proposição(ões)

a) I e II.
b) I.
c) II e III.
d) I e III.
e) II.

15. **(IF-PE / IF-PE – 2019)** Estrutura que contém a representação de dados e rotinas que processam esses dados, assim como representa um conjunto de objetos similares. A definição apresentada é sobre:

a) Objeto.
b) Classe.
c) Atributo.
d) Método.
e) Construtor.

16. **(IF-PE / IF-PE – 2019)** Marque a alternativa que representa a definição de herança, em orientação a objetos.

a) Mecanismo que indica que o acesso aos dados (atributos) dos objetos só deve ocorrer pelos métodos do próprio objeto.
b) Mecanismo que permite o reaproveitamento de comportamentos e dados de outras classes do sistema.
c) Mecanismo que permite que métodos, com o mesmo nome, possam ser reimplementados dentro da própria classe.
d) Representação do quanto uma classe depende de outra classe do sistema.
e) Representação das características essenciais de um objeto e que o diferencia de outros objetos do sistema.

17. **(VUNESP / Câmara de Piracicaba - SP – 2019)** Considerando a orientação a objetos, assinale a alternativa que define corretamente o que é polimorfismo.

a) O armazenamento de objetos de forma permanente, para posterior recuperação em novas execuções do programa.
b) O fato de uma subclasse herdar o comportamento de sua classe mãe.
c) Uma solução reusável para um problema comum.
d) Uma unidade coesa de funcionalidades que podem ser desenvolvidas e entregues independentemente para compor uma unidade maior.
e) Diferentes objetos podem responder à mesma mensagem de maneiras diferentes, possibilitando que objetos interajam uns com os outros sem conhecer seus tipos exatos.

18. **(VUNESP / Câmara de Piracicaba - SP – 2019)** No contexto da orientação a objetos, existe uma medida de quanto dois itens, tais como classes ou métodos, estão inter-relacionados. Esta medida costuma ser classificada como forte, quando um item depende da forma como o outro foi implementado, ou fraca, quando um item depende do outro, mas não de seus detalhes de implementação. Essa medida é conhecida como:

a) coesão.
b) exceção.
c) sobrecarga.
d) interface.
e) acoplamento.

19. **(VUNESP / UFABC – 2019)** Considerando a orientação a objetos, é correto afirmar que:

a) uma mensagem é composta exclusivamente pelo objeto que deve receber tal mensagem.
b) os métodos de uma classe são acionados periodicamente, por decurso de tempo.
c) uma classe deve possuir número de atributos maior do que o seu número de métodos.
d) cada objeto possui um limite máximo de número de mensagens que pode enviar.
e) na herança múltipla, uma subclasse pode ter associada uma ou mais superclasses.

20. **(UFMG / UFMG – 2019)** Uma classe abstrata A contém o método abstrato acao(). A classe B herda da classe A e não implementa o método acao(). Neste contexto, assinale a alternativa CORRETA.

a) A chamada do método acao() de um objeto da classe B chamará a implementação existente na classe A.
b) A classe B não pode sobrecarregar o método acao().
c) A implementação do método abstrato acao() é obrigatória na classe B para que ela compile.
d) A classe B compila sem erros.

21. **(UFMG / UFMG – 2019)** Os quatro pilares do paradigma de Orientação a Objetos são:

a) Sequenciamento, Procedimentos, Bibliotecas e Herança.
b) Herança, Polimorfismo, Classes e Objetos.
c) Classes, Atributos, Métodos e Abstração.
d) Abstração, Encapsulamento, Herança e Polimorfismo.

22. **(IDECAN / IF-PB – 2019)** Sobre os conceitos de Orientação a Objetos, identifique com "V" caso verdadeiro ou "F" caso falso as assertivas a seguir.

( ) A Sobrescrita permite que, em uma mesma classe, tenhamos vários métodos com o mesmo nome, mas com a assinatura diferente.
( ) Objetos são instâncias de uma classe que possui os atributos e as operações definidos na classe.

( ) Superclasse é uma especialização de um conjunto de classes através de herança.
( ) A Sobrecarga possibilita que o mesmo nome possa ser utilizado em diferentes métodos em uma mesma classe, desde que, por exemplo, as quantidades de parâmetros sejam diferentes.
( ) Classe encapsula dados para descrever o conteúdo de alguma entidade do mundo real.

a) V, F, V, F, F
b) V, V, V, F, V
c) F, V, F, V, V
d) F, V, V, V, V
e) V, V, F, F, F

23. **(CCV-UFC / UFC – 2019)** Um dos recursos existentes na orientação a objetos é denominado polimorfismo. Com relação a esse recurso, é correto afirmar:

a) O polimorfismo refere-se à característica de uma classe poder herdar os atributos e métodos de outra classe, tornando-se uma classe mais especializada.

b) Com o polimorfismo, é possível a criação de métodos onde os parâmetros são sempre passados por cópia, independentemente se são tipos primitivos ou objetos.

c) Com o polimorfismo é possível que uma variável de um tipo mais genérico (abstrato) referencie um objeto de um tipo mais específico na sua hierarquia de classes.

d) O polimorfismo restringe que os tipos das variáveis que armazenam um determinado objeto sejam explicitamente do mesmo tipo do objeto, não permitindo generalizações.

e) Refere-se à propriedade de somente tornar visível as informações importantes para o contexto da aplicação, enquanto as demais ficam disponíveis somente no escopo da classe.

24. **(CS-UFG / IF Goiano – 2019)** Programação orientada a objetos está baseada no que é conhecido por orientação a objetos. Nesse contexto,

a) a delegação e a herança são recursos para reutilizar comportamento.
b) a serialização é necessária no processo de especialização.
c) a generalização depende de composição.
d) o polimorfismo depende de sobrecarga (overloading).

25. **(COSEAC / UFF – 2019)** Na orientação objeto existe basicamente três modos de visibilidade. São eles:

a) herança, polimorfismo e agregação.
b) especialização, agregação e associação.
c) composição, realização e agregação.
d) realização, composição e associação.

e) público, protegido e privado.

26. (COSEAC / UFF – 2019) Em relação à orientação objetos, avalie se são verdadeiras (V) ou falsas (F) as afirmativas a seguir:

I Um método pode receber ou não parâmetros e pode retornar valores.
II Uma classe sempre deve possuir atributos e métodos.
III O polimorfismo trabalha com a redeclaração de métodos previamente herdados por uma classe.

a) V, F e V.
b) F, V e V.
c) V, F e F.
d) F, F e V.
e) V, V e V.

27. (IF-PA / IF-PA – 2019) Quanto aos conceitos do paradigma da orientação a objetos, é CORRETO afirmar:

a) por meio do conceito de Polimorfismo, é possível a definição de vários métodos ou funções com o mesmo nome, porém com diferentes assinaturas. Essa característica do conceito de Polimorfismo é denominada de Delegação.

b) por meio do conceito de Herança, uma subclasse é capaz de reutilizar os métodos e atributos de uma superclasse, desde que esses métodos e atributos estejam encapsulados, ou seja, suas visibilidades estejam como "private".

c) uma classe definida como Abstrata, é uma classe que define os seus atributos e métodos para que sejam herdados por uma outra classe que irá implementar os seus métodos. Em uma classe Abstrata não é possível a implementação dos seus métodos, somente os seus cabeçalhos.

d) em uma Interface definimos comportamentos (métodos) sem os implementar. Por meio da Interface podemos definir o que um objeto obrigatoriamente deve fazer e não como ele faz.

e) uma classe que implementa uma classe Abstrata, deverá obrigatoriamente redefinir os métodos e atributos que herdou. A classe que implementa a classe Abstrata não pode definir seus próprios atributos.

28. (QUADRIX / CRQ 4ª Região-SP – 2018) Com o princípio da "herança", um objeto faz reuso de código, possibilitando a redução de esforços no desenvolvimento de sistemas pelo reaproveitamento de códigos herdados de outros objetos ou classes.

29. **(QUADRIX / CRQ 4ª Região-SP – 2018)** Graças ao encapsulamento, os atributos de um objeto podem ser protegidos, permitindo o acesso a eles somente a partir de métodos específicos e autorizados.

30. **(QUADRIX / CRQ 4ª Região-SP – 2018)** Polimorfismo em POO é a combinação de atributos e métodos internos a uma classe, de forma a deixar visível apenas o que é necessário para a comunicação entre dois objetos.

31. **(QUADRIX / CRQ 4ª Região-SP – 2018)** No conceito de orientação a objetos, a herança é a capacidade de um novo objeto tomar atributos e operações de um objeto ou classe já existente.

32. **(QUADRIX / CRQ 4ª Região-SP – 2018)** Na POO, uma classe possui atributos, que são as características comuns a todos os objetos dela derivados, e métodos, que são as operações que devem estar escritas em cada objeto.

33. **(FUNDEP / Prefeitura de Pará de Minas - MG – 2018)** Qual recurso da programação orientada a objetos permite que dois ou mais métodos possuam o mesmo nome desde que utilizem assinaturas diferentes?

a) Encapsulamento.
b) Sobrecarga.
c) Herança.
d) Interface.

34. **(IBADE / IPM-JP – 2018)** No que diz respeito à Orientação a Objetos - OO, analise as abordagens descritas a seguir.

I. Foca o desenvolvimento de um modelo orientado a objetos de um sistema de software para implementar os requisitos especificados. Esses objetos estão relacionados à solução do problema.
II. Foca o desenvolvimento de um modelo orientado a objetos do domínio da aplicação. Esses objetos refletem as entidades e operações associadas ao problema a ser resolvido.

Nesse contexto, I e II são denominadas, respectivamente:

a) Análise OO e Projeto OO.
b) Projeto OO e Análise OO.
c) Programação OO e Análise OO.
d) Projeto OO e Programação OO.
e) Programação OO e Projeto OO.

35. **(FAURGS / UFRGS – 2018)** Em relação ao paradigma de orientação a objetos, assinale as afirmações abaixo com V (verdadeiro) ou F (falso).

( ) Uma linguagem de programação orientada a objetos pode permitir que uma classe tenha zero, um ou vários métodos construtores distintos.

( ) Um método construtor é responsável por alocar espaço em memória para os atributos do objeto.

( ) Um método construtor padrão atribui valores default para todos os atributos do objeto. Esses valores são obrigatoriamente definidos pela linguagem de programação (ex.: o para atributos numéricos).

( ) Uma classe herdeira não precisa definir um método construtor, pois herda, automaticamente, o construtor da classe base.

( ) Um método construtor não padrão permite que cada objeto de uma classe seja instanciado com valores distintos e adequados ao contexto daquele objeto especificamente.

a) V – F – F – F – V.
b) V – V – F – V – V.
c) F – V – F – V – F.
d) V – F – V – F – V.
e) V – F – V – V – F.

36. **(COPESE - UFT / Câmara de Palmas - TO – 2018)** Em Orientação a Objetos, a associação possibilita um relacionamento entre classes/objetos, no qual estes possam pedir ajuda a outros e assim representar de forma completa o conceito no qual se destinam. Neste tipo de relacionamento, as classes e os objetos interagem entre si para atingir seus objetivos. São os tipos de uma associação, EXCETO:

a) acoplamento.
b) agregação.
c) composição.
d) dependência.

37. **(QUADRIX / CRM-PR – 2018)** Com o polimorfismo, é possível que uma operação seja implementada, em uma classe-filha, de forma diferente da classe-pai.

38. **(QUADRIX / CRM-PR – 2018)** É obrigatório que uma classe possua, no mínimo, um atributo.

39. **(QUADRIX / CRM-PR – 2018)** Com a herança, é admitido estabelecer relações entre classes, permitindo o compartilhamento de atributos e operações idênticas.

40. **(QUADRIX / CRM-PR – 2018)** Uma classe especifica a estrutura de um objeto, informando quais serão seus valores.

41. **(QUADRIX / CRM-PR – 2018)** Embora a orientação a objetos seja um paradigma eficaz de análise e desenvolvimento de sistema, com ela, torna-se difícil abstrair, de uma maneira mais fidedigna, as situações do mundo real.

42. **(QUADRIX / CRM-PR – 2018)** Algumas linguagens de programação orientadas a objeto são baseadas em classes, mas há outras que não utilizam as classes de objetos.

43. **(QUADRIX / SEDF – 2018)** A sobreposição é um tipo importante de polimorfismo, também conhecida como polimorfismo ad-hoc. Ela permite que o programador use o mesmo nome de método para muitos métodos diferentes.

44. **(QUADRIX / SEDF – 2018)** Na programação orientada a objetos, o estado do objeto é representado ao armazenar valores em campos.

45. **(COMPERVE / UFRN – 2018)** A orientação a objetos é um paradigma importante para a programação de sistemas. Sobre esse tipo de paradigma, é correto afirmar:

a) apenas classes abstratas podem referenciar métodos abstratos.
b) uma classe pode ser abstrata sem possuir nem referenciar métodos abstratos.
c) um método é abstrato por possuir parâmetros com tipos abstratos ainda não definidos.
d) métodos abstratos são aqueles cujo código invoca métodos ainda não implementados.

46. **(CS-UFG / Câmara de Goiânia - GO – 2018)** Polimorfismo é um conceito usado em programação orientada a objetos, e envolve a seleção dinâmica baseada

a) no objeto referenciado em vez do tipo da referência ao objeto, quando há a sobrescrita (overriding) de métodos.

b) no objeto referenciado em vez do tipo da referência ao objeto, quando há a sobrecarga (overloading) de métodos.

c) no tipo da referência ao objeto em vez do objeto referenciado, quando há a sobrescrita (overriding) de métodos.

d) no tipo da referência ao objeto em vez do objeto referenciado, quando há a sobrecarga (overloading) de métodos.

47. **(CESGRANRIO / Transpetro – 2018)** Qual a propriedade, típica da orientação a objeto, que habilita uma quantidade de operações diferentes a ter o mesmo nome, diminuindo o acoplamento entre objetos?

a) Encapsulamento
b) Especialização
c) Herança

d) Padrões de projeto
e) Polimorfismo

48.(FAURGS / TJ-RS – 2018) Considere as seguintes afirmações sobre herança.

I - Herança é um dos diferenciadores-chaves entre sistemas convencionais e sistemas orientados a objetos. Uma subclasse Y herda todos os atributos e operações associadas a sua superclasse X, ou seja, todos as estruturas de dados e operações de X ficam imediatamente disponíveis para Y.

II - Em cada nível de uma hierarquia de classes com herança, novos atributos e operações não podem ser acrescentados àqueles que foram herdados de níveis mais altos da hierarquia.

III. A herança pode proporcionar benefício significativo ao projeto, mas, se for usada de forma não apropriada, pode complicar um projeto desnecessariamente e resultar em um software passível de erros e difícil de manter.

Quais estão corretas?

a) Apenas I.
b) Apenas I e II.
c) Apenas I e III.
d) Apenas II e III.
e) I, II e III.

49.(FAURGS / TJ-RS – 2018) _____ é uma característica que reduz bastante o esforço necessário para ampliar o projeto de um sistema orientado a objetos, permitindo que várias operações diferentes tenham o mesmo nome.

a) Encapsulamento
b) Agregação
c) Acoplamento
d) Polimorfismo
e) Coesão

50. (FAURGS / TJ-RS – 2018) No contexto da orientação a objetos, _____ é um conceito que encapsula dados e abstrações procedurais necessárias para descrever o conteúdo e comportamento de alguma entidade do mundo real.

a) herança
b) polimorfismo
c) classe
d) método
e) mensagem

51. **(FAURGS / BANRISUL – 2018)** Considere as seguintes afirmações sobre alguns fundamentos de Análise de Sistemas orientada a objetos.

I - Classe é um conceito orientado a objeto que encapsula dados e abstrações procedurais necessárias para descrever o conteúdo e o comportamento de alguma entidade do mundo real. Pode-se dizer que classe é uma descrição generalizada que descreve uma coleção de objetos similares.
II - Superclasse é a generalização de um conjunto de classes a ela relacionadas.
III - Subclasse é uma especialização da superclasse. Uma subclasse herda todos os atributos e operações associadas à sua superclasse e não pode incorporar atributos ou operações adicionais específicos.

Quais estão corretas?

a) Apenas I.
b) Apenas I e II.
c) Apenas I e III.
d) Apenas II e III.
e) I, II e III.

52. **(IF-TO / IF-TO – 2018)** Considere o cenário onde uma Classe B lega suas estruturas e comportamentos de uma Classe A. Essa relação entre a Classe A e a Classe B é caracterizada por:

a) uma anomalia
b) um polimorfismo
c) uma herança
d) uma instância da classe
e) uma anomalia de instância

53. **(CEPS-UFPA / UFPA – 2018)** Em relação ao paradigma orientado a objetos, é CORRETO afirmar que:

a) um objeto consiste em um conjunto de operações encapsuladas e um estado que grava e recupera os efeitos dessas operações.

b) mensagens são requisições enviadas de um atributo para outro, para que o objeto receptor forneça algum resultado por meio da execução de uma operação.

c) classe define as características de um conjunto de atributos que podem herdar dados de outras classes.

d) polimorfismo permite que mensagens sejam propagadas somente utilizando um único método para diferentes classes.

e) a herança permite que objetos herdem atributos de outras classes, sem estender esta característica para métodos.

54. **(FUNRIO / AL-RR – 2018)** Um programador, utilizando orientação a objeto, deseja fazer com que os atributos e serviços disponíveis em uma classe estejam escondidos, de modo que o acesso aos mesmos se dê através de interfaces. O mecanismo de orientação a objetos, adequado para essa característica, é o /a:

a) encapsulamento.
b) estruturação.
c) persistência.
d) polimorfismo.

55. **(CS-UFG / UFG – 2018)** Na programação orientada a objetos, há um mecanismo que permite definir modificadores de acesso. Quando se define um atributo de uma classe com o modificador de acesso privado, significa que:

a) o acesso à classe é privado.
b) o atributo é acessível a um programa que tenha uma referência a um objeto da classe.
c) a classe é abstrata.
d) o atributo é acessível somente aos métodos da classe.

56. **(CS-UFG / SANEAGO - GO – 2018)** Em programação orientada a objetos, o uso de composição ao invés de herança é preferível porque

a) facilita o emprego de estruturas de decisão e controle.
b) reduz o acoplamento.
c) elimina bugs.
d) minimiza o consumo de memória e CPU.

57. **(CS-UFG / SANEAGO - GO – 2018)** A programação orientada a objetos:

a) impossibilita o polimorfismo sem herança (extends em JAVA).
b) impede a construção de software de difícil manutenção.
c) faz uso de conceitos como classe, interfaces e envio de mensagens.
d) requer que classes sejam estendidas para a reutilização de código.

58. **(CS-UFG / SANEAGO - GO – 2018)** Uma classe abstrata A contém o método abstrato foo(), que não foi reimplementado pela classe B que herda de A. Nesse contexto,

a) a criação de uma classe abstrata C, que herda de B, requer a implementação do método foo().

b) a implementação do método foo() em B é obrigatória para que ela compile.
c) a chamada do método foo() de um objeto de B chamará a implementação existente em A.
d) a classe B não pode sobrecarregar o método foo().

59. **(PR-4 UFRJ / UFRJ – 2018)** Com relação aos conceitos de orientação objeto, existe uma característica que faz com que detalhes internos do funcionamento dos métodos de uma classe permaneçam ocultos para os objetos e que por conta dessa técnica, o conhecimento a respeito da implementação interna da classe é desnecessário do ponto de vista do objeto, uma vez que isso passa a ser responsabilidade dos métodos internos da classe. A característica apresentada se refere a:

a) encapsulamento.
b) polimorfismo.
c) abstração.
d) herança.
e) namespaces.

60. **(IBFC / Prefeitura de Divinópolis-MG – 2018)** Assinale a alternativa que complete correta e respectivamente as lacunas da frase a seguir:

O _____ permite que referências de tipos de classes mais _____ representem o comportamento das classes _____que referenciam:

a) polimorfismo - abstratas - concretas
b) encapsulamento - abstratas - concretas
c) encapsulamento - concretas - abstratas
d) polimorfismo - concretas - abstratas

61. **(IBFC / TJ-PE – 2017)** Em um programa orientado a objetos, verifica-se que a classe X estende a classe Y. Ou seja, pode-se afirmar, pelos preceitos da POO (Programação Orientada a Objetos), que:

a) a classe X é superclasse de Y
b) a classe X é uma interface de Y
c) a classe Y é derivada de X
d) a classe Y é subclasse de X
e) a classe X é derivada de Y

62. **(IBFC / EMBASA – 2017)** Quanto aos fundamentos básicos de programação orientada a objetos, relacione os quatro conceitos abaixo com os respectivos significados mencionados logo em seguida:

CONCEITOS:

(1) herança.
(2) método.
(3) polimorfismo.
(4) encapsulamento.

SIGNIFICADOS:

(A) definem as habilidades dos objetos.
(B) é o princípio pelo qual duas ou mais classes, derivadas de uma mesma superclasse, podem invocar métodos que têm a mesma identificação mas comportamentos distintos.
(C) é o mecanismo pelo qual uma classe pode estender outra classe ou, ainda, ser estendida de outra classe.
(D) consiste na separação de aspectos internos e externos de um objeto.

a) 1C - 2B - 3A - 4D
b) 1C - 2A - 3B - 4D
c) 1D - 2A - 3B - 4C
d) 1A - 2C - 3D - 4B.

63. **(IBFC / EBSERH – 2017)** Um dos conceitos em Orientação a Objetos é a classe abstrata. Assinale a alternativa que complete correta e respectivamente as lacunas da frase abaixo:

"A classe abstrata é sempre um(a) _____ que não possui _____"

a) método - instâncias
b) ator/atriz - atributos
c) superclasse - instâncias
d) superclasse - atores/atrizes
e) método - atributos.

64. **(IBFC / EBSERH – 2017)** Consiste no princípio pelo qual duas ou mais classes derivadas de uma mesma superclasse podem invocar métodos que têm a mesma identificação (assinatura) mas comportamentos distintos, especializados para cada classe:

a) abstração
b) herança
c) interface
d) mensagem
e) polimorfismo

65. **(CESGRANRIO / UNIRIO – 2016)** Em linguagens orientadas a objetos (OO), classes representam a descrição da implementação de tipos abstratos a partir dos quais instâncias podem ser criadas. Cada instância, depois de criada, guarda seu estado próprio independente

das demais instâncias. Esse estado pode ser alterado de acordo com operações definidas pela classe, mas, ao serem executadas, as operações atuam individualmente sobre cada instância.

Na nomenclatura OO, instâncias e operações são conhecidas, respectivamente, como

a) Métodos e Funções
b) Objetos e Heranças
c) Objetos e Métodos
d) Tipos e Objetos
e) Tipos e Heranças

66. **(IBFC / EMDEC – 2016)** Quanto a Programação Orientada a Objeto identifique a alternativa que representa o mecanismo pelo qual uma classe (sub-classe) pode estender outra classe (super-classe), aproveitando seus comportamentos (métodos) e variáveis possíveis (atributos):

a) herança
b) encapsulamento
c) mensagem
d) polimorfismo.

67. **(CESGRANRIO / EPE – 2014)** Considere que um programa orientado a objeto possui 5 classes: Máquina, Motor, MotorExplosão, MotorVapor e Gerador. MotorExplosão e MotorVapor são especializações de Motor. Motor e Gerador são especializações de Máquina. Todas as classes respondem a uma mensagem chamada "calcularPotencia", sem argumentos, que calcula e retorna um número real que indica potência do objeto, em watts, de acordo com os valores de alguns atributos, com um algoritmo diferente em cada classe. O exemplo acima caracteriza a capacidade de enviar a mesma mensagem para vários objetos e que cada objeto responda a essa mensagem de acordo com sua classe. Tal característica é conhecida como:

a) Polimorfismo
b) Refatoração
c) Herança Múltipla
d) Independência de Dados
e) Tratamento de Exceção

68. **(CESGRANRIO / IBGE – 2014)** Em linguagens orientadas a objetos, existem dois conceitos fundamentais:

I – a definição de uma estrutura, a partir da qual é possível especificar todas as características da implementação, operações e armazenamento de informações para instâncias que serão criadas posteriormente.

II – instâncias específicas criadas a partir da definição das estruturas referentes ao conceito I.

Esses conceitos correspondem, respectivamente, ao que se conhece pelos nomes de:

a) I - Tipo; II - Classe
b) I - Tipo; II - Construtor
c) I - Classe; II - Tipo
d) I - Classe; II - Objeto
e) I - Classe ; II - Metaclasse

69. **(IBFC / TRE-AM – 2014)** Em programação orientada a objetos significa separar o programa em partes, o mais isoladas possível, tornando o software mais flexível e fácil de modificar:

a) Herança.
b) Encapsulamento.
c) Polimorfismo.
d) Atributo.

70. **(IBFC / PC-RJ – 2013)** Quanto à programação orientada a objeto, simplificadamente classe é o conjunto de objetos com características similares. O conjunto de atributos e métodos agregados a um só objeto, que podem ser visíveis ou invisíveis, é denominado de:

a) Evento
b) Subclasse.
c) Estado.
d) Encapsulamento.
e) Herança.

71. **(IBFC / HEMOMINAS – 2013)** Complete a frase a seguir com uma das alternativas abaixo: "_____ permite que os atributos de classes possam ser declarados como públicos, privados ou protegidos".

a) Polimorfismo.
b) Herança.
c) Abstração.
d) Encapsulamento.

72. **(ESAF / DNIT – 2013)** A herança de D a partir de C é a habilidade que uma classe D tem implicitamente definida:

a) em atributos e análises da classe C.
b) em cada um dos modelos e concepções da classe C.
c) em cada um dos atributos e operações da classe C.
d) em parte das funcionalidades e operações de classes equivalentes.
e) nos programas das classes.

73. **(ESAF / CGU – 2012)** Assinale a opção correta.

a) As classes podem formar heranças segmentadas em classes adjacentes.

b) Overflow é a redefinição do fluxo de uma classe, em uma de suas subclasses.

c) Overriding é a redefinição de um método, definido em uma classe, em uma de suas subclasses.

d) Overriding é a redefinição de uma classe através de métodos de objetos diferentes.

e) As classes não podem formar hierarquias de herança de superclasses e subclasses.

74. **(IADES / PGDF – 2011)** Dentro do paradigma de programação orientada a objetos (POO), há um mecanismo utilizado para impedir o acesso direto ao estado de um objeto, restando apenas os métodos externos que podem alterar esses estados. Assinale a alternativa que apresenta o nome deste mecanismo.

a) Mensagem
b) Herança
c) Polimorfismo
d) Encapsulamento
e) Subclasse

75. **(CESGRANRIO / PETROBRÁS – 2010)** Análise as afirmativas a seguir relativas ao paradigma da orientação a objetos.

I - O princípio do encapsulamento preconiza que um objeto deve esconder a sua complexidade interna.

II - Uma mensagem de um objeto A para um objeto B indica que A realizou uma tarefa requisitada por B.

III - A existência da mesma operação polimórfica definida em duas classes, ClasseA e ClasseB, implica necessariamente que ou ClasseA seja subclasse de ClasseB, ou que ClasseB seja subclasse de ClasseA.

É correto APENAS o que se afirma em:

a) I.
b) II.
c) I e II.
d) I e III.
e) II e III.

76. **(ESAF / SUSEP – 2010)** Em relação à programação orientada a objetos, é correto afirmar que:

a) o objeto é definido por atributos.
b) objetos são instâncias de um atributo.
c) apenas atributos numéricos são válidos.
d) atributos podem ser agrupados em pointvalues.
e) atributos adequados dispensam referências a objetos.

77. **(ESAF / SUSEP – 2010)** É correto afirmar que em herança simples uma superclasse pode ter apenas uma subclasse.

78. **(ESAF / SUSEP – 2010)** Polimorfismo é a:

a) utilização múltipla de programas em análise orientada a objetos.

b) habilidade de uma única operação ou nome de atributo ser definido em mais de uma classe e assumir diferentes implementações em cada uma dessas classes.

c) habilidade de um programador em desenvolver aplicações e caracterizar objetos com múltiplos atributos.

d) utilização de uma classe com diferentes formatos em programas com definição de objetos e atributos.

e) habilidade de uma única variável ser utilizada em diferentes programas orientados a objetos.

79. **(IADES / IADES – 2010)** A análise de sistemas no mundo orientado a objeto é feita analisando-se os objetos e os eventos que interagem com esses objetos. O projeto de software é feito reusando-se classes de objetos existentes e, quando necessário, construindo-se novas classes. Análise e projeto orientados a objeto modelam o mundo em termos de objetos que têm propriedades e comportamentos e eventos que disparam operações que mudam o estado dos objetos que interagem entre si. Sobre os conceitos ou ideias fundamentais da metodologia da análise de sistemas orientada a objeto, assinale a alternativa incorreta.

a) Uma classe é a implementação de software de um tipo de objeto, podendo ser abstrata (quando possui objetos instanciados a partir dela) ou concreta (quando não possui objetos criados a partir dela).

b) Um objeto é qualquer coisa, real ou abstrata, a respeito do qual armazenamos dados e os métodos que os manipulam.

c) Um método de um tipo de objeto referência somente as estruturas de dados desse tipo de objeto. Comparativamente, é similar às funções e procedures do universo da programação.

d) O encapsulamento é importante porque separa a maneira como um objeto se comporta da maneira como ele é implementado, uma vez que a definição sobre como implementar os conhecimentos ou ações de uma classe não são informadas.

GABARITO

1. LETRA C
2. LETRA D
3. LETRA C
4. LETRA E
5. LETRA D
6. LETRA A
7. LETRA D
8. LETRA E
9. LETRA A
10. LETRA C
11. CORRETO
12. LETRA B
13. LETRA B
14. LETRA A
15. LETRA B
16. LETRA B
17. LETRA E
18. LETRA E
19. LETRA E
20. LETRA C
21. LETRA D
22. LETRA C
23. LETRA C
24. LETRA A
25. LETRA E
26. LETRA A
27. LETRA D
28. CORRETO
29. CORRETO
30. ERRADO
31. CORRETO
32. ERRADO
33. LETRA B
34. LETRA B
35. LETRA A
36. LETRA A
37. CORRETO
38. ERRADO
39. CORRETO
40. ERRADO
41. ERRADO
42. CORRETO
43. ERRADO
44. CORRETO
45. LETRA B
46. LETRA A
47. LETRA E
48. LETRA C
49. LETRA D
50. LETRA C
51. LETRA B
52. LETRA C
53. LETRA A
54. LETRA A
55. LETRA D
56. LETRA B
57. LETRA C
58. LETRA B
59. LETRA A
60. LETRA A
61. LETRA E
62. LETRA B
63. LETRA C
64. LETRA E
65. LETRA C
66. LETRA A
67. LETRA A
68. LETRA D
69. LETRA B
70. LETRA D
71. LETRA D
72. LETRA C
73. LETRA C
74. LETRA D
75. LETRA A
76. LETRA A
77. ERRADO
78. LETRA B
79. LETRA A

ESSA LEI TODO MUNDO CONHECE:
PIRATARIA É CRIME.

Mas é sempre bom revisar o porquê e como você pode ser prejudicado com essa prática.

1. 

Professor investe seu tempo para elaborar os cursos e o site os coloca à venda.

2. 

Pirata divulga ilicitamente (grupos de rateio), utilizando-se do anonimato, nomes falsos ou laranjas (geralmente o pirata se anuncia como formador de "grupos solidários" de rateio que não visam lucro).

3. 

Pirata cria alunos fake praticando falsidade ideológica, comprando cursos do site em nome de pessoas aleatórias (usando nome, CPF, endereço e telefone de terceiros sem autorização).

4. 

Pirata compra, muitas vezes, clonando cartões de crédito (por vezes o sistema anti-fraude não consegue identificar o golpe a tempo).

5. 

Pirata fere os Termos de Uso, adultera as aulas e retira a identificação dos arquivos PDF (justamente porque a atividade é ilegal e ele não quer que seus fakes sejam identificados).

6. 

Pirata revende as aulas protegidas por direitos autorais, praticando concorrência desleal e em flagrante desrespeito à Lei de Direitos Autorais (Lei 9.610/98).

7. 

Concurseiro(a) desinformado participa de rateio, achando que nada disso está acontecendo e esperando se tornar servidor público para exigir o cumprimento das leis.

8. 

O professor que elaborou o curso não ganha nada, o site não recebe nada, e a pessoa que praticou todos os ilícitos anteriores (pirata) fica com o lucro.

Deixando de lado esse mar de sujeira, aproveitamos para agradecer a todos que adquirem os cursos honestamente e permitem que o site continue existindo.