MODELO E **PROVA 2º BIMESTRE**

Universidade Tuiuti do Paraná

Aluno

Professor:	Chauã Queirolo
Disciplina:	Programação Orientada a Objetos
Curso:	Análise e Desenvolvimento de Sistemas
Data:	27/06/2025 - 21h00 - 22h40
Valor:	10 **Nota:**

Instruções

Desligue o celular # A prova é **individual** e sem consulta # O gabarito deve ser preenchido a **caneta** # Questões com **rasura** serão **desconsideradas** # Compreensão do **enunciado** faz parte da prova # Tentativas de **fraude** ou comunicação sob pena de atribuição de **nota zero** # Cada questão vale **0,5 pontos**

Gabarito

1	Ⓐ Ⓑ Ⓒ Ⓓ **Ⓔ**	**6**	Ⓐ **Ⓑ** Ⓒ Ⓓ Ⓔ	**11**	Ⓐ Ⓑ Ⓒ **Ⓓ** Ⓔ	**16**	Ⓐ Ⓑ **Ⓒ** Ⓓ Ⓔ
2	Ⓐ **Ⓑ** Ⓒ Ⓓ Ⓔ	**7**	**Ⓐ** Ⓑ Ⓒ Ⓓ Ⓔ	**12**	**Ⓐ** Ⓑ Ⓒ Ⓓ Ⓔ	**17**	Ⓐ Ⓑ Ⓒ **Ⓓ** Ⓔ
3	Ⓐ **Ⓑ** Ⓒ Ⓓ Ⓔ	**8**	Ⓐ Ⓑ **Ⓒ** Ⓓ Ⓔ	**13**	**Ⓐ** Ⓑ Ⓒ Ⓓ Ⓔ	**18**	Ⓐ Ⓑ Ⓒ Ⓓ **Ⓔ**
4	Ⓐ **Ⓑ** Ⓒ Ⓓ Ⓔ	**9**	Ⓐ Ⓑ Ⓒ Ⓓ **Ⓔ**	**14**	**Ⓐ** Ⓑ Ⓒ Ⓓ Ⓔ	**19**	Ⓐ Ⓑ Ⓒ Ⓓ **Ⓔ**
5	Ⓐ Ⓑ **Ⓒ** Ⓓ Ⓔ	**10**	Ⓐ Ⓑ Ⓒ **Ⓓ** Ⓔ	**15**	Ⓐ Ⓑ **Ⓒ** Ⓓ Ⓔ	**20**	Ⓐ Ⓑ Ⓒ **Ⓓ** Ⓔ

Questões

Questão 1 - Em uma aplicação Java, é definida uma classe com o modificador final. Considerando as características desse modificador, assinale a alternativa correta.

a. A classe pode ser estendida por outras subclasses livremente.

b. A classe deve conter apenas métodos estáticos.

c. A classe não pode ser instanciada.

d. A classe deve ser necessariamente abstrata.

e. A classe não pode ser herdada por nenhuma outra classe.

A. Incorreta: Uma classe marcada como final não pode ser estendida por nenhuma outra classe, portanto não pode ser herdada livremente.

B. Incorreta: O modificador final não impõe que a classe contenha apenas métodos estáticos; ela pode conter métodos de instância normalmente.

C. Incorreta: A classe final pode ser instanciada normalmente; o modificador final não impede a instanciação.

D. Incorreta: O modificador final não obriga a classe a ser abstrata; pelo contrário, uma classe abstrata e final simultaneamente não faz sentido, pois a final impede herança e a abstrata impede instanciação direta.

E. Correta: Uma classe declarada como final não pode ser herdada por nenhuma outra classe, impedindo a criação de subclasses.

Questão 2 - Considere a classe Produto que possui os atributos nome e preco. Um construtor é utilizado para inicializar esses atributos no momento da criação do objeto.

Qual a função do construtor nessa classe?

a. Garantir a herança dos atributos para subclasses.
b. Inicializar os atributos da classe no momento da criação do objeto.
c. Realizar validações obrigatórias em tempo de compilação.
d. Criar automaticamente atributos privados para segurança.
e. Impedir que a classe seja instanciada diretamente.

A. Incorreta: O construtor não tem função direta de garantir herança; essa é uma característica da linguagem, não do construtor.

B. Correta: O principal papel do construtor é inicializar os atributos da classe ao criar o objeto, assegurando que ele comece em estado válido.

C. Incorreta: Validações em tempo de compilação são feitas por mecanismos estáticos da linguagem, não pelo construtor em si.

D. Incorreta: O construtor não cria automaticamente atributos privados; o encapsulamento é definido pela declaração explícita dos atributos.

E. Incorreta: O construtor não impede a instanciação; ao contrário, é usado justamente para permitir a criação e inicialização do objeto.

Questão 3 - No desenvolvimento de um sistema com várias formas geométricas, foi criada uma classe abstrata chamada Forma, que tem um método abstrato calcularArea(). As subclasses, como Circulo e Retangulo, implementam esse método de forma específica.

Qual a principal razão para a classe Forma ser abstrata?

a. Permitir que métodos tenham múltiplas implementações dentro da mesma classe.
b. Forçar que as subclasses implementem o método calcularArea().
c. Habilitar a herança múltipla entre as subclasses.
d. Evitar que objetos sejam instanciados diretamente a partir da classe Forma.
e. Garantir que a classe utilize somente atributos estáticos.

A. Incorreta: Métodos não podem ter múltiplas implementações dentro da mesma classe; isso caracteriza sobrecarga, e o método abstrato obriga implementação nas subclasses.

B. Correta: A classe abstrata força as subclasses a implementarem o método abstrato calcularArea(), garantindo que cada forma tenha sua forma específica de calcular área.

C. Incorreta: Java não suporta herança múltipla de classes; portanto, isso não é habilitado pela classe abstrata.

D. Parcialmente correta, porém incompleta: a classe abstrata não pode ser instanciada diretamente, mas essa não é a principal razão para a existência dela nesse contexto.

E. Incorreta: A classe abstrata pode ter atributos estáticos e não está limitada a isso; o modificador abstrato não impõe essa restrição.

Questão 4 - Analise o seguinte código em Java:

```
class Animal {
    void emitirSom() {
        System.out.println("Som genérico");
    }
}

class Cachorro extends Animal {
    void emitirSom() {
        System.out.println("Latido");
    }
}
```

Quando uma variável do tipo Animal referencia um objeto Cachorro e o método emitirSom é invocado, a saída é "Latido". Esse comportamento é um exemplo de qual conceito da programação orientada a objetos?

a. Definição de múltiplas classes com o mesmo nome.
b. Polimorfismo, permitindo que diferentes classes respondam ao mesmo método com comportamentos específicos.
c. Evitar a criação de construtores para simplificar o código.
d. Herança múltipla para reutilizar código.
e. Métodos idênticos em classes sem relação de herança.

A. Incorreta: Não há definição de múltiplas classes com o mesmo nome no código.

B. Correta: O comportamento demonstrado é um exemplo clássico de polimorfismo, onde o método da subclasse sobrescreve o da superclasse e é chamado mesmo pela referência do tipo pai.

C. Incorreta: A presença de métodos sobrescritos não impede a criação de construtores; não há relação direta.

D. Incorreta: Java não permite herança múltipla de classes; o exemplo não demonstra esse conceito.

E. Incorreta: Os métodos são idênticos, mas as classes têm relação de herança, o que é requisito para o polimorfismo.

Questão 5 - Em Java, interfaces podem conter métodos sem implementação, além de permitir que uma classe implemente múltiplas interfaces. Considerando essas informações, assinale a alternativa que melhor descreve as possibilidades das interfaces em Java.

a. Uma interface pode conter apenas métodos abstratos e não métodos com implementação.
b. Uma classe pode implementar apenas uma interface por vez.
c. Interfaces permitem herança múltipla de tipo, possibilitando que uma interface estenda várias outras interfaces.
d. Uma interface pode conter atributos com estado mutável.
e. Métodos em interfaces não podem ter corpo.

A. Incorreta: Desde Java 8, interfaces podem conter métodos com implementação padrão (default), portanto não são restritas a métodos abstratos.

B. Incorreta: Uma classe pode implementar múltiplas interfaces simultaneamente; não há limite de uma única interface.

C. Correta: Interfaces suportam herança múltipla de tipo, permitindo que uma interface estenda várias outras interfaces.

D. Incorreta: Interfaces não podem conter atributos com estado mutável; seus atributos são implicitamente public static final (constantes).

E. Incorreta: Métodos em interfaces podem ter corpo a partir do Java 8 com default e static, não são obrigatoriamente abstratos.

Questão 6 - Qual parte do MVC (Model-View-Controller), mostra a interface gráfica, como formulários e botões para o usuário?

a. Model
b. View
c. Controller
d. DAO
e. VO

A. Incorreta: O Model representa os dados e regras de negócio, não a interface gráfica.

B. Correta: A View é responsável por apresentar a interface gráfica ao usuário, incluindo elementos visuais como formulários e botões.

C. Incorreta: O Controller trata a lógica de controle e interação entre Model e View, não a interface visual direta.

D. Incorreta: DAO (Data Access Object) é um padrão para acesso a dados, não faz parte direta do MVC clássico.

E. Incorreta: VO (Value Object) refere-se a objetos de valor, não a componentes da interface gráfica.

Questão 7 - Durante o desenvolvimento de um sistema, um programador implementou a seguinte estrutura para tratar exceções em Java:

```
try {
    int resultado = 10 / 0;
} catch (ArithmeticException e) {
    System.out.println("Erro de divisão por zero.");
}
```

Qual é a principal função do bloco try-catch no código acima?

a. Tratar erros que ocorrem durante a execução e evitar falhas no programa.
b. Inicializar variáveis estáticas usadas nos cálculos.
c. Controlar acesso a atributos privados da classe.
d. Definir constantes imutáveis durante a execução.
e. Substituir o uso de herança múltipla.

A. Correta: O bloco try-catch captura exceções em tempo de execução, permitindo tratamento adequado e evitando a falha abrupta do programa.

B. Incorreta: Inicialização de variáveis estáticas não é função do bloco try-catch.

C. Incorreta: Controle de acesso a atributos privados é feito por modificadores de acesso, não por tratamento de exceção.

D. Incorreta: Definição de constantes imutáveis é feita por palavras-chave como final, não pelo try-catch.

E. Incorreta: try-catch não substitui herança múltipla, conceito estrutural da linguagem.

Questão 8 - Em um sistema baseado no padrão MVC, a classe DAO (Data Access Object) tem um papel específico para a organização do código. Qual alternativa descreve corretamente sua função?

a. Controlar regras de negócio complexas da aplicação.
b. Gerar a interface gráfica para o usuário.

c. Intermediar o acesso aos dados, encapsulando operações de persistência em banco.

d. Armazenar dados temporários em memória.

e. Validar os dados diretamente na interface.

A. Incorreta: Regras de negócio são tratadas geralmente na camada Model ou Controller, não no DAO.

B. Incorreta: Interface gráfica não é gerada pela classe DAO.

C. Correta: DAO é responsável por encapsular o acesso e manipulação dos dados persistidos, como operações em banco de dados.

D. Incorreta: Armazenamento temporário em memória é responsabilidade de outras estruturas, não DAO.

E. Incorreta: Validação de dados na interface é feita na camada View ou Controller, não diretamente no DAO.

Questão 9 - Considerando o uso de PreparedStatement em Java para operações de banco de dados, assinale qual é a principal vantagem desse recurso.

a. Permitir que somente comandos SELECT sejam executados.

b. Eliminar a necessidade de tratar exceções com blocos try-catch.

c. Permitir a execução simultânea de múltiplas conexões ao banco.

d. Gerar automaticamente relatórios a partir dos dados consultados.

e. Evitar injeção de SQL e melhorar desempenho com pré-compilação.

A. Incorreta: PreparedStatement não se limita a comandos SELECT; suporta INSERT, UPDATE, DELETE.

B. Incorreta: Exceções ainda devem ser tratadas com blocos try-catch ao usar PreparedStatement.

C. Incorreta: Execução simultânea de múltiplas conexões depende da implementação do driver e do banco, não do PreparedStatement especificamente.

D. Incorreta: Geração automática de relatórios não é função do PreparedStatement.

E. Correta: PreparedStatement previne injeção de SQL e otimiza desempenho via pré-compilação da consulta.

Questão 10 - Durante o desenvolvimento de uma aplicação, o programador criou um método que pode lançar uma exceção de entrada e saída (IOException). Para que o código seja compilado corretamente, o método deve indicar essa possibilidade.

Qual é o papel da cláusula throws na assinatura de um método em Java?

a. Garantir que a exceção será tratada automaticamente pelo compilador.

b. Substituir o uso do bloco try-catch.

c. Impedir que o método lance exceções durante sua execução.

d. Indicar que o método pode lançar a exceção, transferindo a responsabilidade do tratamento para quem o invoca.

e. Lançar a exceção automaticamente sem necessidade de tratamento.

A. Incorreta: O compilador não trata exceções automaticamente; cabe ao programador indicar e tratar.

B. Incorreta: A cláusula throws não substitui o uso do try-catch, apenas declara que o método pode propagar a exceção.

C. Incorreta: throws não impede o método de lançar exceções; é justamente o contrário.

D. Correta: A cláusula throws informa que o método pode lançar uma exceção, transferindo a responsabilidade do tratamento para o chamador.

E. Incorreta: throws não lança exceção automaticamente; é uma declaração, não uma ação.

Questão 11 - Em Java, o uso da palavra-chave final aplicada a uma classe impede que ela seja estendida. Considere o seguinte código:

```
final class Produto {
    private double preco;
}
class Eletronico extends Produto {
}
```

Qual problema conceitual ocorre neste trecho?

a. Classes final podem ser herdadas normalmente.

b. O atributo preco deveria ser público para permitir herança.

c. O modificador final só pode ser utilizado em métodos.

d. A classe Eletronico tenta estender uma classe final, o que não é permitido.

e. A classe Produto deveria ser abstrata.

A. Incorreta: Classes final não podem ser herdadas, portanto não podem ser estendidas normalmente.

B. Incorreta: O modificador de acesso do atributo não influencia na possibilidade de herança da classe final.

C. Incorreta: O modificador final pode ser utilizado em classes, métodos e variáveis; não é restrito a métodos.

D. Correta: A classe Eletronico tenta estender uma classe Produto declarada como final, o que é proibido em Java, causando erro de compilação.

E. Incorreta: Uma classe final não precisa ser abstrata; essas são características diferentes e incompatíveis.

Questão 12 - Considere o código abaixo:

```
class Pessoa {
    private String nome;
    public void Pessoa(String nome) {
        this.nome = nome;
    }
}
```

Qual é o erro conceitual presente nesta classe?

a. O método Pessoa está declarado com tipo de retorno void, tornando-se um método comum e não um construtor.

b. A variável nome não foi inicializada.

c. O atributo nome não é acessível para outras classes.

d. O método Pessoa não pode ter o mesmo nome da classe.

e. O uso do this é inválido neste contexto.

A. Correta: O método que tem o mesmo nome da classe mas declara tipo void não é um construtor, é um método comum, o que impede a correta inicialização do atributo.

B. Incorreta: A variável nome foi declarada, mas não inicializada por causa do erro no construtor.

C. Incorreta: O atributo nome é privado, o que impede acesso direto, mas isso não é um erro conceitual, apenas uma prática de encapsulamento.

D. Incorreta: O método pode ter o mesmo nome da classe, mas não pode declarar tipo de retorno para ser construtor.

E. Incorreta: O uso do this é válido para referenciar atributos da classe.

Questão 13 - Em relação às classes abstratas em Java, analise o seguinte exemplo:

```
abstract class Figura {
    abstract void desenhar();
    void mover() {
        System.out.println("Movendo...");
    }
}
Figura f = new Figura();
```

Qual é o erro conceitual ao tentar executar a última linha do código?

a. Não é possível instanciar diretamente uma classe abstrata.
b. O método mover() não é válido em classes abstratas.
c. O método desenhar() deveria ser declarado como final.
d. A classe Figura deveria ser concreta para ser instanciada.
e. O método desenhar() deveria ser estático.

A. Correta: Não é permitido instanciar diretamente uma classe abstrata, pois ela pode conter métodos abstratos sem implementação.

B. Incorreta: Métodos concretos são válidos em classes abstratas; o método mover() está correto.

C. Incorreta: Métodos abstratos não devem ser final, pois devem ser sobrescritos.

D. Incorreta: A classe pode ser abstrata, não necessariamente concreta.

E. Incorreta: Métodos abstratos não devem ser estáticos.

Questão 14 - Considere a classe abaixo com dois construtores:

```
class Produto {
    String nome;
    double preco = 10;
    Produto(String nome) {
        this.nome = nome;
    }
    Produto() {
        preco = 20;
    }
}
```

Qual é o problema conceitual encontrado nesta implementação?

a. Os construtores diferem corretamente pela quantidade ou tipo de parâmetros, o que é permitido.
b. Uma classe não pode possuir mais de um construtor.
c. O construtor sem parâmetros sobrescreve o construtor com parâmetros.
d. A ordem dos construtores influencia a compilação.
e. O compilador não permite sobrecarga de construtores.

A. Correta: A sobrecarga de construtores é permitida e ocorre quando há diferença no número ou tipo dos parâmetros, como no exemplo.
B. Incorreta: Classes podem ter múltiplos construtores, desde que estejam sobrecarregados corretamente.
C. Incorreta: O construtor sem parâmetros não sobrescreve o outro; são métodos distintos pela assinatura.
D. Incorreta: A ordem dos construtores no código não influencia na compilação.
E. Incorreta: O compilador permite sobrecarga de construtores.

Questão 15 - No tratamento de exceções em Java, para lançar manualmente uma exceção quando uma condição é inválida, qual comando deve ser utilizado?

```
if (valor < 0) {
    ________ new IllegalArgumentException("Valor não pode ser negativo");
}
```

a. break
b. return
c. throw
d. throws
e. raise

A. Incorreta: break é usado para sair de loops ou switch, não para lançar exceções.
B. Incorreta: return encerra o método, não lança exceções.
C. Correta: O comando throw é utilizado para lançar explicitamente uma exceção.
D. Incorreta: throws declara que um método pode lançar exceções, não lança diretamente.
E. Incorreta: raise não é uma palavra-chave válida em Java.

Questão 16 - Em Java, qual é a principal característica do bloco finally no tratamento de exceções?

a. Executa apenas quando uma exceção ocorre.
b. Nunca é executado se não houver catch.
c. Sempre é executado após o bloco try/catch, independentemente de ocorrer exceção.
d. Substitui a necessidade de blocos catch.
e. Executa somente quando o catch captura a exceção.

A. Incorreta: O bloco finally é executado independentemente da ocorrência de exceção, não apenas quando uma exceção ocorre.
B. Incorreta: O bloco finally será executado mesmo se não houver bloco catch.
C. Correta: O bloco finally é sempre executado após o bloco try e catch, seja qual for o resultado, garantindo a execução de instruções essenciais, como garantir o encerramento de uma conexão com BD.
D. Incorreta: O finally não substitui blocos catch, que são necessários para capturar exceções.
E. Incorreta: O bloco finally é executado mesmo se o catch não capturar nenhuma exceção.

Questão 17 - Para implementar corretamente um construtor em Java que inicialize um atributo nome na classe Pessoa, qual das alternativas abaixo é adequada?

a. nome = nome;
b. Pessoa() { nome = "padrão"; }
c. String Pessoa(String nome) { nome = nome; }
d. Pessoa(String nome) { this.nome = nome; }
e. public void Pessoa(String nome) { this.nome = nome; }

A. Incorreta: A atribuição nome = nome; sem this referencia o parâmetro, não o atributo, não inicializando o atributo da classe.

B. Incorreta: Embora válido, este construtor padrão não inicializa o atributo via parâmetro.

C. Incorreta: Métodos não podem ter tipo de retorno se forem construtores; esta é uma declaração incorreta de construtor.

D. Correta: A sintaxe está correta: construtor com mesmo nome da classe, sem tipo de retorno, e atribuição correta usando this.nome = nome;.

Questão 18

Na programação orientada a objetos, a herança múltipla pode causar problemas, como o "problema do diamante da morte". Isso acontece quando uma classe herda de duas superclasses que têm versões diferentes do mesmo método, causando dúvida sobre qual método usar.

Com base nisso, qual alternativa explica corretamente o problema do diamante da morte?

a. Quando atributos privados causam conflitos entre classes filhas que impedem a compilação do código.
b. Quando métodos são sobrecarregados em uma interface que é implementada por múltiplas classes.
c. Quando uma interface possui métodos estáticos que conflitam com métodos abstratos em subclasses.
d. Quando a sobrecarga de construtores impede a criação de múltiplas instâncias da classe.
e. Quando uma subclasse herda métodos idênticos de duas superclasses e não consegue determinar qual implementação utilizar, gerando ambiguidade.

A. Incorreta: A questão não é conflito de atributos privados que impeçam compilação, mas ambiguidade de métodos herdados.

B. Incorreta: Sobrecarga de métodos não está relacionada ao problema do diamante.

C. Incorreta: Métodos estáticos em interfaces não causam esse problema.

D. Incorreta: Sobre carga de construtores não impede instâncias, nem está relacionada ao diamante.

E. Correta: O problema ocorre quando uma subclasse herda o mesmo método de duas superclasses e não sabe qual implementar, causando ambiguidade.

Questão 19

Em Java, qual modificador permite que um método pertença à classe, e não a uma instância específica?

a. final
b. private

c. abstract
d. synchronized
e. static

A. Incorreta: final impede sobrescrita, mas não determina se método pertence à classe.
B. Incorreta: private é modificador de acesso, não relacionado a pertencer à classe ou instância.
C. Incorreta: abstract indica método sem implementação, não pertence à classe ou instância necessariamente.
D. Incorreta: synchronized controla acesso concorrente, não escopo de método.
E. Correta: static indica que método pertence à classe, podendo ser chamado sem instanciar objetos.

Questão 20

Em Java, o termo interface refere-se a um tipo especial de contrato que define um conjunto de métodos que uma classe deve implementar.

Qual das alternativas abaixo melhor descreve o conceito de interface em Java?

a. Uma janela ou componente visual utilizado para interação com o usuário.
b. Uma classe abstrata que contém apenas atributos e nenhum método.
c. Um pacote que organiza componentes gráficos em uma aplicação.
d. Um tipo que define um contrato de métodos que uma classe deve implementar, sem conter implementação própria.
e. Um tipo de dado primitivo para manipulação de interfaces visuais.

A. Incorreta: Interface não é componente visual, mas um tipo abstrato.
B. Incorreta: Interface não é classe abstrata e não possui atributos, apenas métodos a implementar.
C. Incorreta: Interface não é um pacote.
D. Correta: Interface define um contrato de métodos que classes devem implementar, sem implementação própria.
E. Incorreta: Interface não é tipo primitivo.