PROVA 1º BIMESTRE

Universidade Tuiuti do Paraná

Aluno

Professor:	Chauã Queirolo
Disciplina:	Programação Orientada a Objetos
Curso:	Análise e Desenvolvimento de Sistemas
Data:	25/04/2025 - 21h00 - 22h40
Valor:	10 **Nota:**

Instruções

Desligue o celular # A prova é **individual** # Apenas uma **folha A4** escrita à **mão** é permitida e deverá ser entregue com a prova # 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: Considere o seguinte cenário:

Uma equipe de desenvolvimento está mantendo um sistema legado escrito inteiramente em linguagem procedural. Um novo módulo, com regras de negócio complexas e múltiplas entidades inter-relacionadas, será adicionado ao sistema. A equipe está avaliando se o novo módulo deve seguir o mesmo paradigma estruturado ou adotar a programação orientada a objetos (POO).

Com base nessa situação, responda: qual seria a principal vantagem da adoção do paradigma orientado a objetos para esse novo módulo?

A. Reduzir o tempo de execução do sistema por meio de algoritmos mais eficientes.
B. Permitir maior reutilização de código e modularização através de encapsulamento e herança.
C. Diminuir a quantidade de código necessário ao remover a necessidade de estruturas condicionais.
D. Aumentar a dependência entre módulos para facilitar a integração do sistema.
E. Evitar a criação de novos arquivos, mantendo o código centralizado em uma única unidade.

Gabarito: B

Justificativa: A programação orientada a objetos favorece a modularização e a reutilização de código ao introduzir conceitos como encapsulamento, herança e polimorfismo. Esses mecanismos permitem representar entidades do domínio com maior fidelidade e facilitam a manutenção do sistema.

Questão 2: Em qual situação é mais indicado o uso do paradigma orientado a objetos em vez do paradigma estruturado?

A. Em programas simples, com poucas regras de negócio e baixo acoplamento.
B. Em scripts de automação pontual sem necessidade de manutenção.
C. Em sistemas complexos, com múltiplas entidades e requisitos de reuso e manutenção.

D. Em programas que utilizam exclusivamente estruturas condicionais e laços.

E. Em algoritmos matemáticos de alto desempenho, com baixa interação entre componentes.

Gabarito: C

Justificativa: A orientação a objetos é mais adequada para sistemas de maior complexidade, onde há necessidade de abstrair entidades e promover reuso e manutenção eficiente.

Questão 3: O encapsulamento é alcançado em Java principalmente através do uso de:

A. variáveis globais

B. construtores sobrecarregados

C. interfaces públicas

D. modificadores de acesso

E. palavra-chave static

Gabarito: D

Justificativa: O encapsulamento é implementado com modificadores de acesso como private, protected e public, que controlam a visibilidade.

Questão 4: Uma empresa de logística está desenvolvendo um sistema orientado a objetos em Java. Para representar os elementos do domínio, a equipe decidiu criar classes como 'Caminhão', 'Motorista' e 'Rota'. Com base nesse contexto, analise as afirmações a seguir:

I. Um objeto representa uma instância de uma classe, como um 'Caminhão' específico.

II. A classe define o comportamento e os atributos comuns a todos os objetos do mesmo tipo.

III. A criação de uma instância é feita por meio do operador 'new' em Java.

IV. Uma classe pode existir sem que nenhum objeto seja criado a partir dela.

Assinale a alternativa correta:

A. Apenas as afirmações I e II estão corretas.

B. Apenas as afirmações II e III estão corretas.

C. Apenas as afirmações I, II e III estão corretas.

D. Apenas a afirmação IV está incorreta.

E. Todas as afirmações estão corretas.

Gabarito: E

Justificativa: Todas as afirmações representam corretamente os conceitos fundamentais da orientação a objetos: instanciação de objetos (I), definição de classe como modelo (II), uso do operador 'new' (III), e existência de classes sem objetos (IV).

Questão 5: Complete a frase: Em orientação a objetos, uma _____ é um modelo ou estrutura que define os atributos e comportamentos comuns de um conjunto de objetos, enquanto um _____ é uma instância concreta dessa estrutura.

A. função / objeto

B. classe / objeto

C. método / classe

D. atribuição / referência

E. interface / método

Gabarito: B

Justificativa: A definição clássica em orientação a objetos distingue 'classe' como modelo e 'objeto' como instância concreta dessa classe.

Questão 6: Em Java, o que é necessário para criar um novo objeto a partir de uma classe?

A. Usar o operador instanceof.
B. Declarar a classe como static.
C. Utilizar o operador new com a classe desejada.
D. Atribuir diretamente os atributos da classe.
E. Importar obrigatoriamente o pacote java.util.

Gabarito: C

Justificativa: Em Java, a criação de um objeto é realizada por meio do operador 'new', que instancia a classe definida.

Questão 7: Em Java, qual das alternativas representa corretamente a definição de um construtor padrão (default)?

A. public void Construtor() {}
B. void Cliente() {}
C. public Cliente() {}
D. public static Cliente() {}
E. Cliente Cliente() {}

Gabarito: C

Justificativa: O construtor padrão é definido com o mesmo nome da classe e não possui tipo de retorno, nem mesmo void.

Questão 8: Associe os conceitos da coluna A aos seus respectivos significados na coluna B:

Coluna A	Coluna B
4. Método	( ) Comportamento de um objeto.
5. Instanciação	( ) Processo de criação de um objeto a partir de uma classe.
1. Objeto	( ) Representação de uma entidade concreta.
2. Classe	( ) Estrutura que define dados e comportamentos.
3. Atributo	( ) Característica ou propriedade de um objeto.

Assinale a alternativa com a sequência correta:

A. 5 – 4 – 2 – 1 – 3
B. 3 – 2 – 4 – 1 – 5
C. 2 – 1 – 5 – 4 – 3
D. 1 – 3 – 2 – 4 – 5
E. 4 – 5 – 1 – 2 – 3

Gabarito: E

Justificativa: Método define comportamento (4), instanciação é a criação do objeto (5), o objeto representa entidade concreta (1), classe define a estrutura (2) e atributo representa propriedade (3).

Questão 9: Considere o seguinte código:

```
class Animal {
    String nome;
    void emitirSom() {
        System.out.println(nome + " faz som");
    }
}

Animal a = new Animal();
a.nome = "Cachorro";
a.emitirSom();
```

Esse código é um exemplo de:

A. Herança entre classes.
B. Encapsulamento avançado.
C. Instanciação e chamada de método em um objeto.
D. Sobrescrita de método herdado.
E. Polimorfismo de tempo de compilação.

Gabarito: C

Justificativa: O código demonstra a criação de um objeto e a chamada de um método da classe, caracterizando o uso direto da instância.

Questão 10: Quando uma classe contém outra e controla o seu ciclo de vida, essa relação é chamada de:

A. Composição
B. Herança
C. Associação
D. Agregação
E. Modularização

Gabarito: A

Justificativa: Na composição, o objeto contido é criado e destruído junto com o objeto que o contém, caracterizando controle de ciclo de vida.

Questão 11: Ao declarar a classe abaixo, qual das afirmações está correta quanto à criação e uso de objetos?

```
class Pessoa {
    String nome;
    int idade;

    void apresentar() {
        System.out.println("Nome: " + nome);
    }
}
Pessoa p1 = new Pessoa();
p1.nome = "Carlos";
p1.apresentar();
```

A. O método apresentar() deve ser declarado como static.

B. O objeto p1 não pode acessar atributos diretamente.
C. A classe Pessoa precisa herdar de uma superclasse explícita.
D. O objeto p1 pode acessar o método apresentar() pois ele pertence à instância.
E. É necessário importar java.lang para declarar a classe Pessoa.

Gabarito: D

Justificativa: Métodos não estáticos pertencem à instância do objeto. Como p1 é uma instância, ele pode invocar normalmente o método apresentar().

Questão 12: Considere a seguinte classe Java:

```
class Cliente {
    String nome;

    Cliente(String n) {
        nome = n;
        System.out.println("Cliente criado: "
                           + nome);
    }
}
```

Sobre a execução do código abaixo, assinale a alternativa correta.

```
Cliente c = new Cliente("Ana");
```

A. O método Cliente(String n) é um destrutor, pois imprime uma mensagem ao ser chamado.
B. A linguagem Java não permite sobrecarga de métodos como o construtor Cliente.
C. A chamada new Cliente("Ana") resulta em erro, pois a classe não possui método main.
D. A execução cria um objeto da classe Cliente e chama seu construtor, inicializando o atributo nome.
E. A classe Cliente precisa herdar de uma superclasse para ter construtor válido.

Gabarito: D

Justificativa: O construtor definido com parâmetro é invocado no momento da criação do objeto com new, inicializando corretamente o atributo nome.

Questão 13: Analise as afirmações sobre encapsulamento em Java:

I. Atributos privados só podem ser acessados por métodos da mesma classe.
II. Encapsulamento favorece a proteção do estado interno do objeto.
III. Métodos públicos expõem funcionalidades controladas ao mundo externo.
IV. A ausência de encapsulamento pode comprometer a consistência dos dados.

Assinale a alternativa correta:

A. Apenas I, II e III são verdadeiras.
B. Apenas II, III e IV são verdadeiras.
C. Apenas I, II e IV são verdadeiras.
D. Apenas I e IV são verdadeiras.
E. Todas são verdadeiras.

Gabarito: E

Justificativa: Todas as afirmações estão corretas. O encapsulamento é um pilar essencial para integridade, segurança e organização dos objetos.

Questão 14: Complete corretamente: Um _____ é um método especial utilizado para inicializar objetos, enquanto o _____ é um método herdado da classe Object, executado pelo coletor de lixo antes da remoção do objeto da memória.

A. construtor / destrutor

B. finalize / initialize

C. init / delete

D. new / static

E. criação / exclusão

Gabarito: A

Justificativa: O construtor é usado para inicializar objetos, e o método finalize() atua como um destrutor implícito no Java.

Questão 15: Complete corretamente: O encapsulamento permite que os detalhes internos de uma classe sejam _____ e o acesso a eles seja feito por meio de _____.

A. abertos / atributos

B. ocultos / métodos públicos

C. públicos / construtores

D. privados / modificadores static

E. externos / referências

Gabarito: B

Justificativa: Encapsular é ocultar os dados e controlar seu acesso via métodos públicos que implementam regras.

Questão 16: Complete corretamente: A _____ ocorre quando um objeto utiliza outro sem controlar seu ciclo de vida, enquanto a _____ representa um vínculo mais forte, no qual o objeto composto não existe sem o seu todo.

A. agregação / composição

B. composição / herança

C. associação / agregação

D. composição / agregação

E. herança / associação

Gabarito: A

Justificativa: A agregação é uma forma mais fraca de composição, pois o objeto agregado pode existir independentemente. A composição, por outro lado, implica dependência total.

Questão 17: Considere o seguinte código Java:

```
class Produto {
    String nome;
    double preco;

    void reajustarPreco(double percentual) {
        preco += preco * percentual / 100;
    }
}

Produto p = new Produto();
p.nome = "Caneta";
p.preco = 2.0;
p.reajustarPreco(10);
```

Com base neste código, analise as afirmativas:

I. O atributo 'nome' armazena o estado do objeto.

II. O método reajustarPreco manipula diretamente o atributo preco.

III. A chamada ao método reajustarPreco modifica o estado interno do objeto p.

IV. O método reajustarPreco deve retornar o novo valor do preço para ser válido.

Assinale a alternativa correta:

A. Apenas I, II e III estão corretas.

B. Apenas I, II e IV estão corretas.

C. Apenas II, III e IV estão corretas.

D. Apenas a afirmativa IV está incorreta.

E. Todas as afirmativas estão corretas.

Gabarito: C e D

Justificativa: As afirmativas I, II e III estão corretas: atributos armazenam estado, métodos podem modificá-los e a mudança ocorre via chamada de método. A afirmativa IV está incorreta, pois um método void pode modificar atributos sem necessidade de retorno.

Questão 18: Dado o seguinte código:

```
class Temperatura {
    double graus;

    void aumentar(double valor) {
        graus += valor;
    }
    void zerar() {
        graus = 0;
    }
}

Temperatura t = new Temperatura();
t.aumentar(10);
t.zerar();
t.aumentar(5);
System.out.println(t.graus);
```

O valor impresso será:

A. 0

B. 5

C. 10

D. 15

E. Erro de compilação

Gabarito: B

Justificativa: O atributo graus é modificado por dois métodos: primeiro definido como 10, depois zerado, e finalmente incrementado em 5. Valor final: 5.

Questão 19: Associe os elementos da coluna A aos exemplos da coluna B:

Coluna A	Coluna B
1. Atributo	( ) preco
2. Método	( ) Produto
3. Objeto	( ) p.reajustarPreco(10)
4. Classe	( ) p

Assinale a alternativa com a sequência correta:

A. 2 – 1 – 4 – 3
B. 1 – 3 – 4 – 2
C. 3 – 4 – 1 – 2
D. 4 – 1 – 3 – 2
E. 1 – 4 – 2 – 3

Gabarito: E

Justificativa: 'preco' é um atributo (1), 'Produto' é a classe (4), 'p.reajustarPreco(10)' representa um método sendo chamado (2), e 'p' é o objeto (3).

Questão 20: Considere as seguintes classes:

```
class Produto {
    String nome;
    double preco;
}

class ProdutoDAO {
    void salvar(Produto p) {
        // lógica de persistência
    }
}

class ProdutoView {
    void exibir(Produto p) {
        System.out.println(p.nome + " - R$ " + p.preco);
    }
}
```

A organização acima representa corretamente:

A. O uso de herança múltipla em diferentes camadas do sistema.
B. Uma violação do encapsulamento, pois as classes se conhecem mutuamente.
C. Um padrão exclusivo da linguagem Java para classes anônimas.
D. A separação de responsabilidades entre lógica de apresentação, domínio e acesso a dados.
E. A ausência total de reutilização de código.

Gabarito: C

Justificativa: O código apresenta uma separação clara de responsabilidades, comum em arquiteturas orientadas a objetos: domínio (Produto), persistência (ProdutoDAO) e apresentação (ProdutoView).