

PROVA — PROGRAMAÇÃO ORIENTADA A OBJETOS COM JAVA

Disciplina: Programação Orientada a Objetos

Linguagem: Java

Duração: 2 horas

Valor: 10,0 pontos

Aluno: _____

Data: _/_/

Turma: ____

INSTRUÇÕES

1. Leia atentamente todas as questões antes de iniciar.
2. A prova contém **20 questões**, sendo:
3. **12 questões objetivas;**
4. **8 questões discursivas/práticas.**
5. Nas questões objetivas, assinale apenas uma alternativa.
6. Nas questões discursivas, apresente o raciocínio e, quando solicitado, escreva o código Java.
7. Não é permitido consultar materiais externos.
8. A duração da prova é de **2 horas**.

PARTE I — QUESTÕES OBJETIVAS

Questão 1 — Classes e objetos

(0,4 ponto)

Em Programação Orientada a Objetos, uma classe pode ser entendida como:

- A) Uma variável utilizada para armazenar objetos.
- B) Um modelo que define características e comportamentos dos objetos.
- C) Um método responsável por criar bancos de dados.
- D) Um arquivo que contém apenas métodos `static`.
- E) Um tipo especial de exceção.

Questão 2 — Instanciação

Considere o código:

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

Qual das alternativas cria corretamente um objeto da classe `Pessoa`?

A)

```
Pessoa = new Pessoa();
```

B)

```
new Pessoa p;
```

C)

```
Pessoa p = new Pessoa();
```

D)

```
Pessoa p();
```

E)

```
object Pessoa = Pessoa();
```

Questão 3 — Encapsulamento

Analise a classe:

```
public class Conta {  
  
    private double saldo;  
  
    public double getSaldo() {  
        return saldo;  
    }  
  
    public void depositar(double valor) {  
        if (valor > 0) {
```

```
        saldo += valor;
    }
}
}
```

O principal objetivo do uso de `private` no atributo `saldo` é:

- A) Impedir que a classe seja instanciada.
- B) Tornar o atributo automaticamente `static`.
- C) Permitir acesso direto de qualquer classe.
- D) Controlar o acesso ao estado interno do objeto.
- E) Transformar o atributo em uma constante.

Questão 4 — Construtores

Considere:

```
public class Produto {

    private String nome;
    private double preco;

    public Produto(String nome, double preco) {
        this.nome = nome;
        this.preco = preco;
    }
}
```

Qual alternativa cria corretamente um objeto?

A)

```
Produto p = new Produto();
```

B)

```
Produto p = Produto("Mouse", 50.0);
```

C)

```
Produto p = new Produto("Mouse", 50.0);
```

D)

```
new Produto p("Mouse", 50.0);
```

E)

```
Produto p("Mouse", 50.0);
```

Questão 5 — this

Observe:

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

Nesse contexto, `this.nome` representa:

- A) Uma variável local criada pelo construtor.
- B) O atributo `nome` pertencente ao objeto atual.
- C) Uma variável `static`.
- D) O nome da classe.
- E) Um método chamado `nome`.

Questão 6 — Herança

Considere:

```
public class Animal {  
    public void emitirSom() {  
        System.out.println("Som");  
    }  
}  
  
public class Cachorro extends Animal {  
    public void latir() {  
        System.out.println("Au au");  
    }  
}
```

Sobre o código, é correto afirmar que:

- A) `Animal` herda de `Cachorro`.
 - B) `Cachorro` não pode utilizar métodos de `Animal`.
 - C) `Cachorro` é uma subclasse de `Animal`.
 - D) As duas classes são independentes.
 - E) `extends` indica composição.
-

Questão 7 — Sobrescrita de métodos

Considere:

```
class Animal {  
  
    public void emitirSom() {  
        System.out.println("Som genérico");  
    }  
}  
  
class Cachorro extends Animal {  
  
    @Override  
    public void emitirSom() {  
        System.out.println("Au au");  
    }  
}
```

O conceito demonstrado pelo método `emitirSom()` redefinido em `Cachorro` é:

- A) Sobrecarga.
 - B) Encapsulamento.
 - C) Composição.
 - D) Sobrescrita de método.
 - E) Agregação.
-

Questão 8 — Polimorfismo

Analise:

```
Animal animal = new Cachorro();  
animal.emitirSom();
```

Considerando as classes da questão anterior, a saída será:

- A) Som genérico
- B) Cachorro

- C) Animal
 - D) Au au
 - E) Erro de compilação
-

Questão 9 — Sobrecarga

Considere:

```
public class Calculadora {  
  
    public int somar(int a, int b) {  
        return a + b;  
    }  
  
    public double somar(double a, double b) {  
        return a + b;  
    }  
}
```

O código demonstra:

- A) Herança múltipla.
 - B) Sobrescrita.
 - C) Sobrecarga de métodos.
 - D) Encapsulamento.
 - E) Abstração.
-

Questão 10 — static

Considere:

```
public class Contador {  
  
    private static int quantidade = 0;  
  
    public Contador() {  
        quantidade++;  
    }  
  
    public static int getQuantidade() {  
        return quantidade;  
    }  
}
```

Após executar:

```
Contador c1 = new Contador();  
Contador c2 = new Contador();  
Contador c3 = new Contador();  
  
System.out.println(Contador.getQuantidade());
```

Qual será a saída?

- A) 0
- B) 1
- C) 2
- D) 3
- E) Erro de compilação

Questão 11 — Composição

Considere:

```
public class Motor {  
}  
  
public class Carro {  
  
    private Motor motor;  
  
    public Carro() {  
        motor = new Motor();  
    }  
}
```

A relação entre `Carro` e `Motor` representa principalmente:

- A) Herança.
- B) Polimorfismo.
- C) Composição.
- D) Sobrecarga.
- E) Interface.

Questão 12 — `ArrayList`

Considere:

```
ArrayList<String> nomes = new ArrayList<>();  
  
nomes.add("Ana");
```

```
nomes.add("Carlos");
nomes.add("Maria");

System.out.println(nomes.size());
```

Qual será o resultado?

- A)
- B)
- C)
- D)

E) Erro de compilação.

PARTE II — QUESTÕES DISCURSIVAS E PRÁTICAS

Questão 13 — Conceitos fundamentais de POO

(0,8 ponto)

Explique, com suas próprias palavras, os seguintes conceitos:

- a) Classe
- b) Objeto
- c) Atributo
- d) Método
- e) Encapsulamento

Para cada conceito, apresente também um pequeno exemplo relacionado a um sistema de software.

Questão 14 — Classe, atributos, construtor e métodos

(1,0 ponto)

Crie uma classe Java chamada `Produto` contendo:

- atributo `nome` ;
- atributo `preco` ;
- atributo `quantidade` ;
- construtor para inicializar os três atributos;
- método `adicionarEstoque(int quantidade)` ;
- método `removerEstoque(int quantidade)` ;
- método `calcularTotal()` , que deve retornar:

preco × quantidade

Os atributos devem ser encapsulados.

Questão 15 — Controle de acesso

(0,8 ponto)

Explique a diferença entre os modificadores de acesso:

- `private`
- `protected`
- `public`

Depois, explique por que, em uma classe que utiliza encapsulamento, é comum declarar os atributos como `private` e disponibilizar métodos `get` e `set`.

Questão 16 — Herança e polimorfismo

(1,0 ponto)

Crie as seguintes classes:

```
Funcionario
|
+--- Gerente
|
+--- Desenvolvedor
```

A classe `Funcionario` deve possuir:

- `nome`;
- `salario`;
- método `calcularBonus()`.

A classe `Gerente` deve sobrescrever `calcularBonus()`, retornando **20% do salário**.

A classe `Desenvolvedor` deve sobrescrever `calcularBonus()`, retornando **10% do salário**.

Depois, crie um exemplo utilizando polimorfismo:

```
Funcionario funcionario = new Gerente(...);
```

e explique por que o método correto é executado mesmo a variável sendo do tipo `Funcionario`.

Questão 17 — ArrayList de objetos

(1,0 ponto)

Crie uma classe:

Aluno

com os seguintes atributos:

matricula
nome
nota

Depois, crie um programa que:

1. Crie um `ArrayList<Aluno>`;
2. Cadastre pelo menos três alunos;
3. Percorra a lista utilizando um `for`;
4. Exiba o nome e a nota de cada aluno;
5. Exiba somente os alunos aprovados, considerando nota maior ou igual a `6.0`.

Questão 18 — Composição

(1,0 ponto)

Um sistema de vendas possui:

```
Pedido
|
+--- Cliente
|
+--- Produto
```

Explique se essas relações poderiam ser representadas por **composição ou agregação**.

Em seguida, apresente um exemplo em Java que demonstre uma classe `Pedido` contendo objetos de outras classes.

Explique também por que utilizar objetos como atributos pode representar melhor o domínio do sistema do que simplesmente utilizar tipos primitivos.

Questão 19 — Análise de código

(1,0 ponto)

Analise o código abaixo:

```
public class Conta {  
  
    private double saldo;  
  
    public Conta(double saldo) {  
        this.saldo = saldo;  
    }  
  
    public void sacar(double valor) {  
        saldo -= valor;  
    }  
  
    public double getSaldo() {  
        return saldo;  
    }  
}
```

- a) Identifique um problema relacionado à regra de negócio do método `sacar()`.
 - b) Explique o que poderia acontecer se o usuário tentar sacar um valor maior que o saldo.
 - c) Reescreva o método `sacar()` de maneira que o saldo não possa ficar negativo.
 - d) Explique como o encapsulamento ajuda a proteger essa regra de negócio.
-

Questão 20 — Desenvolvimento de uma solução orientada a objetos

(1,0 ponto)

Uma empresa deseja desenvolver um sistema para controlar veículos.

Um veículo possui:

- placa;
- marca;
- modelo;
- ano.

O sistema deverá trabalhar com diferentes tipos de veículos:

```
Veiculo
|
+--- Carro
|
+--- Moto
```

O **Carro** possui a informação de quantidade de portas.

A **Moto** possui a informação de cilindradas.

Faça uma proposta de implementação utilizando os conceitos de:

- classe;
- objeto;
- encapsulamento;
- construtor;
- herança;
- sobrescrita;
- polimorfismo.

Não é necessário implementar um sistema completo. O objetivo é demonstrar, por meio do código e da explicação, como os conceitos de POO seriam aplicados nessa situação.

FIM DA PROVA

Boa prova!