Universidade de Brasília Faculdade UnB Gama

UNIVERSIDADE DE BRASÍLIA - UnB
FACULDADE DE CIÊNCIAS E TECNOLOGIAS EM ENGENHARIA

Nome: _____ **Matrícula:** _____
Data: ____/____/_____ **Professor:** Lucas Boaventura

P1 Teórica - OO - 2025.1 - T06

Gabarito:

Questão 1	Questão 2	Questão 3	Questão 4	Questão 5

1. Uma aplicação de gerenciamento de pagamentos precisa oferecer suporte a diferentes métodos, como boleto, cartão de crédito e Pix. Para isso, a classe pagamento foi projetada com um método abstrato chamado processarPagamento(), que deve ser implementado de maneira específica por cada classe correspondente a um tipo de pagamento. Essa abordagem foi escolhida para aproveitar o conceito de polimorfismo. Nesse contexto, qual é a principal vantagem proporcionada pelo uso desses conceitos em relação à expansão e manutenção do sistema? (1 ponto)

a. Centralizar a lógica de todos os métodos em uma única classe, reduzindo a duplicação de código.
b. Reduzir a complexidade ao permitir que todos os métodos utilizem a mesma implementação genérica.
c. Facilitar a adição de novos tipos de pagamento sem alterar o código existente.
d. Garantir que todos os tipos de pagamento utilizem a mesma estrutura interna de dados.

2. Sobre herança na orientação a objetos podemos afirmar que: (1 ponto)

1) Por meio do recurso de herança, uma classe pode ser especializada.
2) É possível criar novas classes (filhas) a partir de uma classe já existente (mãe), reaproveitando seus atributos e operações.
3) Na relação de classes do tipo mãe-filha, chamamos a classe mãe de superclasse e as classes filhas de subclasses.

A. Somente 1 e 2 estão corretos.
B. Somente 2 e 3 estão corretos.
C. Somente 1 e 3 estão corretos.
D. Todas estão erradas.
E. Todas estão corretas.

3. Considerando os princípios da Programação Orientada a Objetos, como herança e polimorfismo, analise o seguinte trecho de código em Java e, em seguida, assinale a alternativa que mostra a saída da execução desse código: (1 ponto)

```java
class Disciplina {
    public void aplicar_prova() {
        System.out.println("Prova sendo aplicada...");
    }
}
class OrientacaoObjetos extends Disciplina {
    @Override
    public void aplicar_prova() {
        System.out.println("Prova de Orientação a Objetos sendo
aplicada...");
    }
}
class Calculo1 extends Disciplina {
    @Override
    public void aplicar_prova() {
        System.out.println("Prova de Cálculo 1 sendo aplicada...");
    }
}
public class Teste {
    public static void main(String[] args) {
        Disciplina minhaDisciplina = new OrientacaoObjetos();
        Disciplina outraDisciplina = new Calculo1();
        minhaDisciplina = outraDisciplina;
        minhaDisciplina.aplicar_prova();
    }
}
```

a. Erro de Execução.
b. Erro de Compilação.
c. Prova de Orientação a Objetos sendo aplicada…
d. Prova de Cálculo 1 sendo aplicada…
e. Prova sendo aplicada...

4. Métricas para Sistemas Orientados a Objetos (OO) são, fundamentalmente, diferentes dos sistemas desenvolvidos usando métodos convencionais. As métricas para análise são organizadas em categorias que refletem importantes características do sistema. Para a categoria de encapsulamento, tem-se a seguinte métrica: (1 ponto)

a. Número de Classes Raiz.
b. Número de Scripts de Cenário.
c. Falta de Coesão nos Métodos.
d. Número de filhos da Superclasse.

5. Sobre encapsulamento em classes Java. O que caracteriza o encapsulamento em uma classe Java?: (1 ponto).

A. Expor todos os atributos com public para acesso fácil.
B. Criar métodos estáticos para acessar atributos.
C. Tornar atributos privados e fornecer métodos públicos para acesso controlado.
D. Utilizar herança para compartilhar atributos entre classes.
E. Garantir que todos os atributos sejam constantes (final).

6. Considerando a orientação a Objetos na linguagem Java, faça: (5,5 pontos)

a. O que é uma classe? **(0,5 pontos)**

b. Faça um exemplo de uma classe em Java com 3 atributos (**String marca, int memoriaRAM, float preco)** e seus respectivos métodos de acesso **(Get e Set)**. **(1 ponto)**

c. Faça dois construtores para a classe 'Computador', um recebendo parâmetros e outro sem parâmetros. **(1 ponto)**

d. Implemente, dentro da sua classe, uma sobrescrita (override) do método toString() para que ele retorne uma string formatada contendo todas as informações (atributos) da sua classe. **(1 ponto)**

e. Agora faça uma **main** que cria duas instâncias da classe que você implementou, uma com parâmetros e outra sem parâmetros, na instância com parâmetros passe a marca sendo seu primeiro nome, o preço será seus 4 primeiros dígitos da matrícula e a memória seus últimos 2 dígitos da matrícula (por exemplo, se a matrícula for 20251234 → preço: 2025, memória: 34). Em seguida, altere um dos atributos da instância que foi criada **com parâmetros.** Por fim, chame o método **toString()** e exiba o resultado no terminal usando **System.out.println()**. **(1,5 pontos)**

f. O que seu código vai imprimir na tela? **(0,5 pontos)**

Main.java

```
public class Main {
    public static void main(String[] args) {









    }
}
```

Computador.java