



8.00

## Avaliação A2

Nome: Ericksen Giesel Müller Turma: T1:☒ T2:☐

### Orientações:

1. A prova pode ser feita a lápis, porém o professor se dará ao direito de não aceitar reclamações relativas à correção.
2. Coloque o seu nome nas folhas de respostas.
3. Manter celulares/dispositivos eletrônicos desligados!

### Boa sorte!

0,75 **Questão 1 – (1,0 pontos)** Com relação aos modificadores `final` e interfaces em Java, assinale Verdadeiro (V) ou Falso (F):

- ☒ (F) Em Java, uma classe pode implementar apenas uma interface.
- ☒ (V) Uma classe declarada como `final` não pode derivar características para outras subclasses.
- ☒ (F) Um campo declarado como `final` pode ser alterado após sua inicialização.
- ☒ (F) Uma interface pode conter métodos concretos, ou seja, com implementação.

**Questão 2 – (1,5 pontos)** Sobre classes e métodos abstratos, assinale Verdadeiro (V) ou Falso (F):

- 0,5 ☒ (V) Um método abstrato pode existir em uma classe que não seja definida como abstrata.
- ☒ (V) Uma classe abstrata não pode ser instanciada diretamente.
- ☒ (V) Métodos abstratos são métodos declarados que não possuem implementação.
- ☒ (F) Uma classe que contém pelo menos um método abstrato deve ser declarada como uma classe abstrata.
- ☒ (F) Uma classe abstrata pode ter métodos concretos (com implementação).
- ☒ (V) Uma classe que herda características de uma classe abstrata, é opcional implementar/sobrescrever os métodos abstratos herdados.

**Questão 3 – (1,0 pontos)** Sobre o trecho de código abaixo, faça uma breve análise do uso de exceções e assinale a alternativa correta em relação a análise obtida:

```
public class X{
    public static void main(String[] args) {
        try {
            int[] array = new int[5];
            array[10] = 50;
            System.out.println("Valor: " + array[10]);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Erro: Acesso a índice inválido do array!");
        } finally {
            System.out.println("Bloco finally executado.");
        }
    }
}
```

- a) O código compilará, mas lançará uma NullPointerException em tempo de execução.
- b) O código compilará, mas lançará uma ArithmeticException em tempo de execução.
- c) O código compilará e executará sem lançar nenhuma exceção.
- d) O código não compilará devido ao acesso inválido ao índice do array.
- ☒ e) O código compilará, lançará uma ArrayIndexOutOfBoundsException e executará o bloco finally.

**Questão 4 – (1,0 pontos)** Considerando os tipos de **Polimorfismo** em Java, e seus recursos, relacione as colunas:

1 Por Subtipagem/Inclusão	(3) Ocorre quando métodos com mesmo nome porém com assinaturas diferentes são definidos em uma mesma classe.
2 Paramétrico	(4) double d = 1;
3 Por Sobrecarga / Overloading	(1) Permite que objetos de diferentes classes sejam tratados como objetos de uma classe comum.
4 Por Coerção	(2) Permite a definição de classes e métodos com tipos genéricos, preservando a segurança dos tipos;

- a) 1, 2, 3, 4;
- b) 3, 2, 4, 1;
- ☒ c) 3, 4, 1, 2
- d) 3, 4, 2, 1;
- e) Nenhuma das anteriores.



Questão 5 – (1,0 pontos) Sobre Desenvolvimento Visual e Persistência de Dados em Java, assinale Verdadeiro (V) ou Falso (F).

(F) Em Java, a persistência de dados só pode ser realizada em bancos de dados relacionais; não é possível usar arquivos.

(V) JDBC (Java Database Connectivity) é uma API em Java que permite a conexão e execução de operações em bancos de dados;

(V) O Java Swing é uma biblioteca de desenvolvimento visual para criar interfaces gráficas de usuário (GUI) em Java;

(F) No desenvolvimento de GUIs com Swing, é obrigatório manipular todos os componentes gráficos diretamente no código, sem ferramentas de design visual

Questão 6 – (1,5 pontos) Considerando o uso de Coleções e Mapas (HashSet, LinkedHashSet, TreeSet, ArrayList, Vector, LinkedList, PriorityQueue, HashTable, LinkedHashMap, HashMap, TreeMap), preencha as lacunas com a melhor implementação para cada situação:

a) <i>LinkedList</i>	Tipo de arranjo de dados onde cada nó contém um dado e uma referência para o próximo nó.
b) <i>Array List</i>	Tipo de arranjo de dados que permite repetição de números e crescimento dinâmico.
c) <i>Hash Table</i>	Permite armazenar chaves e valores nulos e não garante ordenação dos dados
d) <i>Hash Map</i>	Tipo de lista utilizada em cenários que requerem maior controle e thread-safety.
e) <i>TreeSet</i>	Oferece a inserção de elementos únicos e ordenados.
f) <i>Hash Set</i>	Oferece a inserção de elementos de forma rápida, permite a inserção somente de elementos únicos e não garante a ordem dos elementos inseridos.



2.75 • **Questão 7 – (3,0 pontos)** Desenvolva um sistema para gerenciar veículos em uma frota, modelando as classes Veiculo, Carro e Moto. Para a resolução do exercício, considere:

- a) **(0,5 pontos)** As classes Carro e Moto devem herdar características da classe Veiculo; ✓
- b) **(0,5 pontos)** A classe Veiculo deve incluir ao menos dois atributos comuns, que possam ser utilizados pelas classes Carro e Moto; ✓
- c) **(0,5 pontos)** A classe Veiculo deve ser modelada de modo que objetos não podem ser criados com o tipo Veículo, somente objetos derivados das classes Carro e Moto; ✓
- d) **(0,5 pontos)** Ao menos as classes especializadas devem fazer uso de construtor; ✓
- e) **(0,5 pontos)** Utilize boas práticas de nomenclatura e encapsulamento para definir as classes, atributos e métodos; ✓
- f) **(0,5 pontos)** Em uma classe de teste, crie instâncias de Carro e outra de Moto, configure alguns atributos e demonstre a funcionalidade de um método que exiba as informações dos veículos. ✓



7-)

~~public class~~

public abstract class Veiculo {

int ano;

String cor;

public abstract void detalhes();

construtor

}  
public class Carro extends Veiculo {

int numPortas;

public Carro(ano, cor, numPortas) {

super(ano, cor); // esquece o construtor da super

this.numPortas = numPortas;

}

@Override

public void detalhes() {

System.out.println("ano: " + this.ano +

"cor: " + this.cor +

"Portas: " + this.numPortas);

}

}

1000 1000 = 10  
80 1000 = 10  
20 1000 = 10  
01 1000 = 10

```
public Veiculo (ano, cor) {  
    this.ano = ano;  
    this.cor = cor; }  
}
```

```
public class Moto extends Veiculo {
```

```
    int potencia;
```

```
    public Moto (ano, cor, potencia) {
```

```
        super (ano, cor);
```

```
        this.potencia = potencia; }
```

```
@Override
```

```
public void detalhes () {
```

```
    System.out.println("ano: " + this.ano +
```

```
        "cor: " + this.cor +
```

```
        "potencia: " + this.potencia);
```

```
}
```

```
}
```

```
public class Main {
```

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

```
        Carro c1 = new Carro (2011, "branco", 4);
```

```
        Moto m2 = new Moto (2018, "branco", 150);
```

```
        c1.detalhes();
```

```
        m2.detalhes();
```

```
}
```

```
}
```