INSTITUTO FEDERAL
Triângulo Mineiro
Campus Paracatu

Disciplina: Programação Orientada a Objetos	Professor: Igor de Oliveira	Entrega: 17/01/2025

Nome: Luiz Otávio Nazar Silva

Instruções de entrega:

- Provas iguais serão zerados por inteiro;
- Responda na própria prova.

Prova – Java, abstração, arraylist e herança

1. **Sobre herança em Java, analise as afirmações a seguir e marque (V) para verdadeiro ou (F) para falso:**

a) (F) Uma classe em Java pode herdar de mais de uma classe ao mesmo tempo.
b) (V) O uso do modificador protected permite que subclasses em pacotes diferentes acessem os métodos da classe pai.
c) (F) O método super() deve sempre ser a primeira instrução dentro de um construtor de uma subclasse.
d) (V) Uma classe filha pode sobrescrever métodos da classe pai para alterar seu comportamento.

2. **Considere os seguintes códigos:**

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

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

```
public class Gato extends Animal {
    @Override
    void emitirSom() {
        System.out.println("Miau");
    }
}
```

Qual será a saída do código a seguir?

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

a) "Miau"
b) "Latido"
c) "Som genérico de animal"
d) O código não compilará, pois a classe Animal não pode ser instanciada.

3. **Sobre construtores em Java, analise as afirmações abaixo:**

a) (V) Um construtor de uma subclasse pode acessar o construtor da superclasse usando super().
b) (V) Se uma classe não declarar explicitamente um construtor, o compilador criará um construtor padrão sem argumentos.
c) (F) Um construtor pode ser herdado de uma classe pai.
d) (F) É obrigatório declarar pelo menos um construtor em todas as classes Java.

4. ~~**Considere o código:**~~

```
public class Pessoa {
    private String nome;

    public Pessoa(String nome) {
        this.nome = nome;
    }

    public String getNome() {
        return nome;
    }
}
```

```
public class Funcionario extends Pessoa {
    private double salario;

    public Funcionario(String nome, double salario) {
        super(nome);
        this.salario = salario;
    }

    public double getSalario() {
        return salario;
    }
}
```

O que acontece ao executar o seguinte código?

a) O código não compila, pois Funcionário não implementa o método getNome().
b) O código compila e exibe "João".
c) O código não compila, pois não é permitido atribuir uma subclasse a uma

superclasse.
d) O código compila, mas gera um erro de execução ao acessar o método getNome().

5. **Sobre o uso do ArrayList em Java, analise em verdadeiro(V) ou falso(F):**

a) (V) Um ArrayList aceita elementos de diferentes tipos, desde que sejam especificados como Object.
b) (V) O método add() do ArrayList insere um elemento na última posição da lista.
c) (V) Para acessar um elemento do ArrayList, utiliza-se o método get(), fornecendo o índice desejado.
d) (F) O ArrayList tem tamanho fixo, definido no momento da sua criação.

6. **Considere o seguinte código:**

```
import java.util.ArrayList;

public class Teste {
    public static void main(String[] args) {
        ArrayList<String> lista = new ArrayList<>();
        lista.add("A");
        lista.add("B");
        lista.add("C");
        lista.remove(1);
        System.out.println(lista);
    }
}
```

Qual será a saída ao executar o programa?

a) [A, B, C]
b) [A, C]
c) [B, C]
d) [A, B]

7. **Sobre modificadores de acesso em Java:**

a) (V) O modificador public permite que a classe ou membro seja acessível de qualquer lugar do programa.
b) (V) O modificador private restringe o acesso ao membro apenas à classe onde ele foi declarado.
c) (F) O modificador protected permite acesso apenas dentro do mesmo pacote.
d) (F) Construtores sempre usam o modificador public.

8. **Observe o código:**

```
public class Animal {
    public void emitirSom() {
        System.out.println("Animal faz um som");
    }
}
```

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

    public void correr() {
        System.out.println("Cachorro está correndo");
    }
}
```

Complete os seguintes trechos de código:

1. A classe Cachorro deve herdar da classe Animal.
2. O método correr deve ser declarado como público.

9. Observe o código:

```
public class Pessoa {
    private String nome;

    public Pessoa(String nome) {
        this.nome = nome;
    }

    public void exibirNome() {
        System.out.println("Nome: " + nome);
    }
}
```

```
public class Funcionario extends Pessoa {
    private double salario;

    public Funcionario(String nome, double salario) {
        super(nome);
        this.salario = salario;
    }

    @Override
    public void exibirNome() {
        System.out.println("Nome do Funcionário: " + _____);
    }

    public void exibirInformacoes() {
        exibirNome();
        System.out.println("Salário: " + salario);
    }
```

```
    public void trabalhar() {
        System.out.println("Funcionário está trabalhando");
    }
}
```

Complete os seguintes trechos de código:

1. O parâmetro do construtor da classe Pessoa.
2. Os parâmetros do construtor da classe Funcionario.
3. O método trabalhar na classe Funcionario.

10. Observe o código:

```
import java.util.ArrayList;

import java.util.Iterator;



public class Biblioteca {

    private ArrayList<String> livros;



    public Biblioteca() {

        livros = new ArrayList<>();

    }



    public void adicionarLivro(String livro) {

        livros.add(livro);

    }



    public void exibirLivros() {

        Iterator<String> iterator = livros.iterator();
```

```
        while (iterator.hasNext()) {
            System.out.println(iterator.next());
        }
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        Biblioteca biblioteca = new Biblioteca();
        biblioteca.adicionarLivro("Dom Casmurro");
        biblioteca.adicionarLivro("O Alienista");
        biblioteca.exibirLivros();
    }
}
```

Complete os seguintes trechos de código:

1. Inicializar a variável livros na classe Biblioteca.
2. Adicionar um livro à ArrayList na classe Biblioteca.
3. Inicializar o Iterator na classe Biblioteca.