

Classes e Objetos

Parte 2

O que são Variáveis Estáticas?

Definição: Variáveis associadas à classe, e não a instâncias individuais.

Propriedade: Existe uma única cópia da variável, compartilhada por todas as instâncias da classe.

```
public class Exemplo {  
    public static int contador = 0;  
  
    public Exemplo() {  
        contador++;  
    }  
}
```

O que são Métodos Estáticos?

Definição: Métodos que pertencem à classe em vez de a uma instância específica.

Uso: Acessam e manipulam variáveis estáticas, não podem acessar variáveis de instância.

```
public class Utilidades {  
    public static void imprimirMensagem(String mensagem) {  
        System.out.println(mensagem);  
    }  
}
```

Visibilidade de Variáveis

Modificadores de Acesso:

- `private`: Acessível apenas dentro da própria classe.
- `protected`: Acessível dentro da classe e suas subclasses.
- `public`: Acessível de qualquer lugar.
- `default` (pacote-privado): Acessível apenas dentro do mesmo pacote.

```
public class Exemplo {  
    private int variavelPrivada;  
    protected int variavelProtegida;  
    public int variavelPublica;  
    int variavelPacote;  
}
```

Visibilidade de Métodos

Modificadores de Acesso:

- `private`: Usado apenas dentro da classe.
- `protected`: Usado dentro da classe e subclasses.
- `public`: Usado por qualquer classe.
- `default` (pacote-privado): Usado por classes do mesmo pacote.


```
public class Exemplo {  
    private void metodoPrivado() {  
        // Código aqui  
    }  
  
    protected void metodoProtegido() {  
        // Código aqui  
    }  
  
    public void metodoPublico() {  
        // Código aqui  
    }  
  
    void metodoPacote() {  
        // Código aqui  
    }  
}
```

Herança

- Permite a criação de novas classes baseadas em classes existentes.
- Subclasse: herda de uma superclasse.
- Superclasse: classe base.
- Herança Simples: uma subclasse de uma superclasse.
- Herança Múltipla: uma subclasse de múltiplas superclasses (não suportado em Java).

```
class Animal {  
    void fazerSom() {  
        System.out.println("Som do animal");  
    }  
}  
  
class Cachorro extends Animal {  
    @Override  
    void fazerSom() {  
        System.out.println("Latido");  
    }  
}
```


Polimorfismo

- Capacidade de diferentes classes serem tratadas como instâncias da mesma classe.
- Polimorfismo de Subtipo: instâncias de subclasses usadas onde superclasses são esperadas.

```
Animal meuAnimal = new Cachorro();  
meuAnimal.fazerSom(); // Output: Latido
```

Classes Abstratas

- Classes Abstratas: não podem ser instanciadas; contêm métodos com ou sem implementação.

```
abstract class Forma {  
    abstract void desenhar();  
}  
  
class Circulo extends Forma {  
    void desenhar() {  
        System.out.println("Desenhando um círculo");  
    }  
}
```

Interfaces

- Interfaces: declaram métodos que devem ser implementados pelas classes.

```
interface Movivel {  
    void mover();  
}  
  
class Carro implements Movivel {  
    public void mover() {  
        System.out.println("Carro está se movendo");  
    }  
}
```

Interfaces X Classes Abstratas

- Classe Abstrata: Uma classe que não pode ser instanciada e pode conter métodos abstratos (sem implementação) e métodos concretos (com implementação).
- Interface: Um contrato que define métodos que devem ser implementados pelas classes que a "assinam". Interfaces não contêm implementação de métodos (exceto métodos default em Java 8+).

Encapsulamento

- Restrição do acesso direto aos atributos de um objeto.
- Getters e Setters: métodos para acessar e modificar atributos privados.

```
class Pessoa {  
    private String nome;  
  
    public String getNome() {  
        return nome;  
    }  
  
    public void setNome(String nome) {  
        this.nome = nome;  
    }  
}
```


Composição

- Uma classe composta de instâncias de outras classes.
- Relação "Has-a": uma classe contém instâncias de outras classes.

```
class Motor {  
    void ligar() {  
        System.out.println("Motor ligado");  
    }  
}  
  
class Carro {  
    private Motor motor = new Motor();  
  
    void ligarCarro() {  
        motor.ligar();  
    }  
}
```

Perguntas?

Exercicios

Exercício 1: Definição de Classe

Objetivo: Criar uma classe simples e instanciar objetos.

Instruções:

- Crie uma classe chamada Carro.
- Adicione os atributos marca, modelo e ano.
- Crie um método detalhes que imprime os detalhes do carro.

Exercício 2: Getters e Setters

Objetivo: Implementar métodos getters e setters.

Instruções:

- Adicione os métodos getMarca, setMarca, getModelo, setModelo, getAno e setAno na classe Carro.
- Utilize esses métodos no método main(psvm) para definir e obter os valores dos atributos.

Exercício 3: Usando Construtores

Objetivo: Aprender a criar e usar construtores em uma classe.

Instruções:

- Crie uma classe chamada Carro.
- Adicione os atributos marca (String) e modelo (String).
- Crie um construtor que inicializa esses atributos.
- Crie um método chamado exibirDetalhes que imprime a marca e o modelo do carro.

BONUS: Crie também um construtor vazio

Exercício 4: Criando Métodos com Parâmetros

Objetivo: Aprender a criar métodos que aceitam parâmetros.

Instruções:

- Crie uma classe chamada Calculadora.
- Adicione um método chamado somar que aceita dois parâmetros inteiros e retorna a soma deles.
- Crie um método main para testar o método somar.

Exercício 5: Trabalhando com Múltiplas Classes

Objetivo: Aprender a trabalhar com múltiplas classes e instâncias de objetos.

Instruções:

- Crie uma classe chamada Endereco com os atributos rua (String), numero (int) e cidade (String).
- Adicione uma classe chamada Pessoa que tenha um atributo endereco do tipo Endereco.
- Crie métodos para definir e exibir os detalhes da pessoa e do endereço.