Guia de Referência Rápida – Java

Rodrigo Rafael Villarreal Goulart v.i.1

1. Classe

- Definição: Modelo ou blueprint para criar objetos.
- Exemplo:

```
public class Carro {
    String cor;
    String modelo;
    int ano;
}
```

2. Objeto

- Definição: Instância de uma classe.
- Exemplo:

```
Carro meuCarro = new Carro();
```

3. Método

- Definição: Função definida dentro de uma classe.
- Exemplo:

```
public class Carro {
    public void acelerar() {
        System.out.println("O carro está acelerando.");
    }
}
```

4. Construtor

- Definição: Método especial usado para inicializar objetos.
- Exemplo:

```
public class Carro {
    String modelo;

    public Carro(String modelo) {
        this.modelo = modelo;
    }
}
```

5. Encapsulamento

- Definição: Restrição do acesso direto a alguns dos componentes de um objeto.
- Exemplo:

```
public class Pessoa {
    private String nome;

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }
}
```

6. Herança

- Definição: Mecanismo onde uma classe herda atributos e métodos de outra classe.
- Exemplo:

```
public class Veiculo {
    // atributos e métodos
}

public class Carro extends Veiculo {
    // atributos e métodos adicionais
}
```

7. Abstração

- Definição: Processo de ocultar os detalhes complexos e mostrar apenas a funcionalidade essencial.
- Exemplo:

```
public abstract class Animal {
    public abstract void som();
}

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

8. Polimorfismo

- Definição: Capacidade de uma variável, objeto ou função assumir múltiplas formas.
- Exemplo:

```
Veiculo meuVeiculo = new Carro();
```

9. Sobrecarga de Métodos

- Definição: Definir múltiplos métodos com o mesmo nome, mas com diferentes parâmetros.
- Exemplo:

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

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

10. Sobrescrita de Métodos

- Definição: Redefinir um método da superclasse na subclasse.
- Exemplo:

```
public class Animal {
    public void som() {
        System.out.println("Som do animal");
    }
```

```
}

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

11. Contratos (Interfaces)

- Definição: Definem métodos que uma classe deve implementar.
- Exemplo:

```
public interface Animal {
    void som();
}

public class Cachorro implements Animal {
    public void som() {
        System.out.println("Latido");
    }
}
```

12. Coleções

- Definição: Estruturas de dados que armazenam grupos de objetos.
- Exemplo:

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;

public class ExemploColecao {
    public static void main(String[] args) {
        List<String> lista = new ArrayList<>();
        lista.add("Banana");
        lista.add("Maçã");
        lista.add("Laranja");

        // Ordenação usando Collections
        Collections.sort(lista);
        System.out.println("Ordenado com Collections: " + lista);

        // Ordenação usando método sobrescrito
        List<Carro> carros = new ArrayList<>();
        carros.add(new Carro("BMW", 2020));
        carros.add(new Carro("Audi", 2018));
        carros.add(new Carro("Mercedes", 2019));

        Collections.sort(carros);
        System.out.println("Ordenado com método sobrescrito: " + carros);

        // Ordenação usando Comparator
        Collections.sort(carros, new Comparator<Carro>() {
            @Override
            public int compare(Carro c1, Carro c2) {
                return c1.modelo.compareTo(c2.modelo);
            }
        });
        System.out.println("Ordenado com Comparator: " + carros);
    }
}

class Carro implements Comparable<Carro> {
    String modelo;
    int ano;

    public Carro(String modelo, int ano) {
        this.modelo = modelo;
        this.ano = ano;
    }

    @Override
    public int compareTo(Carro outroCarro) {
        return this.ano - outroCarro.ano;
    }

    @Override
    public String toString() {
        return modelo + " (" + ano + ")";
    }
}
```

13. Exceções

- Definição: Mecanismo para lidar com erros e outras condições excepcionais.
- Exemplo:

```
public class ExemploExcecao {
    public static void main(String[] args) {
        try {
            int resultado = 10 / 0;
        } catch (ArithmeticException e) {
            System.out.println("Erro: Divisão por zero.");
        }
    }
}
```

14. Tipos de Encapsulamento

- **Private**: Acesso apenas dentro da própria classe.
- **Default (Package-Private)**: Acesso dentro do mesmo pacote.
- **Protected**: Acesso dentro do mesmo pacote e por subclasses.
- **Public**: Acesso de qualquer lugar.