Programação Avançada - AULA 04

Matheus Moresco
Engenharia de Software - 5º Período
2025/01

Classes Abstratas e Interfaces

Classe Abstrata

- Modelo para outras classes, pode ter métodos com e sem implementação.
- Usada quando várias classes compartilham comportamento comum.

Interface

- Define um contrato que as classes devem seguir, sem implementação.
- Usada quando diferentes classes precisam da mesma funcionalidade, sem herança direta.

O que é uma Classe Abstrata?

Uma classe abstrata é uma classe que não pode ser instanciada diretamente e pode conter métodos concretos (com implementação) e métodos abstratos (sem implementação).

- Características das Classes Abstratas
  - ✓Pode ter métodos abstratos (sem corpo) e métodos concretos (com corpo).
  - ✓Pode ter atributos e construtores.
  - ✓Pode ter modificadores de acesso (private, protected, public).
  - ✓Uma classe concreta deve herdar (extends) de uma classe abstrata e implementar seus métodos abstratos.

Classe Abstrata

- Exemplo de uma classe abstrata

```java
abstract class Animal {
    protected String nome;

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

    public void dormir() { // Método concreto (tem corpo)
        System.out.println(nome + " está dormindo...");
    }

    public abstract void fazerSom(); // Método abstrato (sem corpo)
}
```

Classe Abstrata

- Exemplo de Classe Concreta Herdando a Classe Abstrata

```
class Cachorro extends Animal {
    public Cachorro(String nome) {
        super(nome);
    }

    @Override
    public void fazerSom() {
        System.out.println(nome + " está latindo: Au au!");
    }
}

public class Teste {
    public static void main(String[] args) {
        // Animal a = new Animal("Genérico"); // ERRO! Classe abstrata não pode ser instanci
        Cachorro dog = new Cachorro("Rex");
        dog.fazerSom(); // Rex está latindo: Au au!
        dog.dormir(); // Rex está dormindo...
    }
}
```

Classe Abstrata

- **Resumo:** Classes abstratas são usadas quando queremos definir um comportamento comum para várias classes, mas deixando a implementação de certos métodos para as subclasses.

O que é uma Interface?

Uma interface é um contrato que define um conjunto de métodos sem implementação. Qualquer classe que implementa uma interface é obrigada a fornecer implementações para os métodos definidos nela.

- Características das Interfaces
  - ✓Não pode ter atributos de instância, apenas constantes (public static final).
  - ✓Não pode ter construtores.
  - ✓Todos os métodos são abstratos por padrão (até o Java 7).
  - ✓Uma classe pode implementar várias interfaces, mas só pode estender uma única classe.

Interface

- Exemplo de interface

```
interface Voador {
    void voar();
}
```

```
class Passaro implements Voador {
    @Override
    public void voar() {
        System.out.println("O pássaro está voando.");
    }
}
```

```
class Aviao implements Voador {
    @Override
    public void voar() {
        System.out.println("O avião está decolando.");
    }
}
```

Interface

- **Resumo:** Interfaces são usadas quando queremos definir um conjunto de comportamentos que podem ser compartilhados por várias classes **sem necessidade de herança**.

```
public class Teste {
    public static void main(String[] args) {
        Voador v1 = new Passaro();
        Voador v2 = new Aviao();

        v1.voar(); // O pássaro está voando.
        v2.voar(); // O avião está decolando.
    }
}
```

Classe Abstrata *vs* Interface

Característica	Classe Abstrata	Interface
Pode ser instanciada?	Não	Não
Pode ter métodos concretos?	Sim	Até Java 7 (Java 8+ permite métodos default)
Pode ter atributos?	Sim	Apenas constantes (final static)
Permite múltiplas heranças?	Não (somente uma classe)	Sim (várias interfaces podem ser implementadas)
Modificadores de acesso	private, protected, public	Apenas public

Quando Usar Cada Um?

- **Use uma classe abstrata** quando deseja definir **atributos e métodos comuns** para várias classes e permitir reuso de código.

- **Use uma interface** quando deseja definir um **contrato** que pode ser compartilhado por várias classes sem relação de herança.

- Se você precisa de uma hierarquia de classes com comportamento padrão, **use classes abstratas**. Se precisa apenas de um conjunto de métodos sem estado, **use interfaces**.

Exemplo Prático

1. Usar a Classe Carro;
2. Criar a classe Veiculo (Classe Abstrata): Define atributos comuns (marca, modelo, ano) e um método abstrato acelerar().
3. Criar uma classe Moto e herdar de veículo.
4. Criar a classe Bicicleta e herdar de veículo;
5. Combustivel (Interface): Define um contrato para o método abastecer().
6. Implementar Combustivel, nas classes Carro e Moto, adiciona a funcionalidade abastecer().

Exercício

1. Usar exemplo da ContaBancaria, torna-la abstrata.
2. Criar classes herdando da ContaBancaria:
    1. ContaCorrente
        1. Adicionar o atributo tarifa, é um número real que indica o valor descontado para cada saque.
        2. Sobrescrever o método saque para descontar o valor da tarifa e imprimir na tela o valor descontado no saque,
    2. ContaPoupanca
        1. Adicionar o atributo rendimentoMensal, que informa a porcentagem que a conta rende no mês.
        2. Crie o método aplicarRendimentoMensal que adiciona o rendimento do mês ao saldo.
3. Criar interface Tributavel com o método calcularIR(), implemente esta interface na classe ContaPoupanca.