Universidade Estadual de Maringá
Departamento de Informática

Polimorfismo

Classes abstratas e Interfaces

Conteúdo baseado nos materiais dos Professor
Marcos Aurélio Domingues (DIN/UEM)

Prof.ª Juliana Keiko Yamaguchi
abril de 2019

Objetivos

- Estudar e compreender o conceito de métodos abstratos em duas abordagens:
  - Classes abstratas;
  - Interfaces.

Polimorfismo

- No polimorfismo, o programador precisa escrever os métodos polimórficos.
- Mas se o programador esquecer de escrever os métodos?
- Como garantir que o programador não se esqueça de escrever os métodos polimórficos?

Polimorfismo
Métodos abstratos

- Há duas formas de garantir que o programador não se esqueça de implementar os métodos polimórficos:
  - Classes abstratas;
  - Interfaces.

Classes abstratas

- Uma classe abstrata é uma classe que não pode ser instanciada, ou seja, não pode ser diretamente utilizada para criar objetos.
- O propósito da criação de uma classe abstrata é de fornecer uma superclasse apropriada para que outras classes a utilizem como base (herança).

Classes abstratas

- Uma classe abstrata pode conter:
  - Métodos com implementação;
  - Métodos abstratos, ou seja, métodos sem implementação.
- A técnica de especificar métodos abstratos permite que o programador decida quais são os comportamentos que as subclasses devem ter, mas sem determinar como tais comportamentos serão implementados.

Classes abstratas
Exemplo (genérico)

```
public abstract class NomeClasseAbstrata{

   public String metodo1(){
      return "String";
   }

   public abstract void metodo2();

   public abstract int metodo3();
}
```

- A classe que herdar de NomeClasseAbstrata deverá fornecer a implementação dos métodos abstratos.

Classes abstratas
Exemplo (genérico)

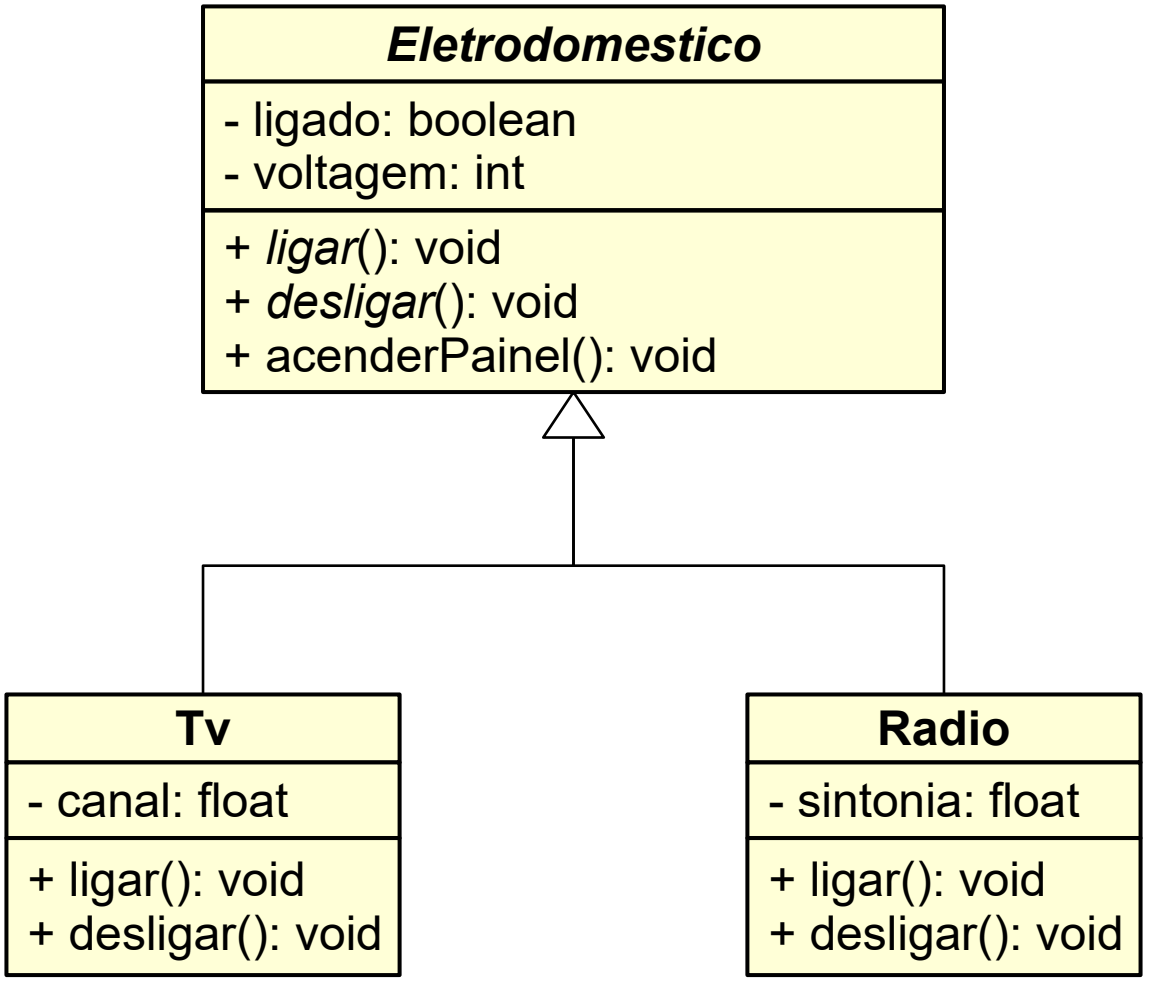
```
public class NomeClasse extends NomeClasseAbstrata{

   public void metodo2(){
      //implementação obrigatória
   }

   public int metodo3(){
      //implementação obrigatória
   }
}
```

- A subclasse herda metodo1() mas deve implementar metodo2() e metodo3().

Classes abstratas

- Somente classes abstratas podem ter métodos abstratos.
- Classes abstratas garantem o polimorfismo, já que todos os métodos abstratos polimórficos devem ser implementados pela subclasse.
  - Dessa forma, a classe abstrata serve como um “modelo” para outras classes que herdarem dela.

Classes abstratas
Exemplo



- Crie um projeto chamado Aparelhos
  - Crie uma classe chamada Eletrodomestico.

Classes abstratas
Exemplo

```
public abstract class Eletrodomestico {
    private boolean ligado;
    private int voltagem;

    public abstract void ligar();
    public abstract void desligar();

    public void acenderPainel(Eletrodomestico e){
         System.out.println("Mostrando painel de " +
e.getClass().getSimpleName());
     }
    public boolean isLigado() {
        return ligado;
    }
    public void setLigado(boolean ligado) {
        this.ligado = ligado;
    }
    public int getVoltagem() {
        return voltagem;
    }
    public void setVoltagem(int voltagem) {
        this.voltagem = voltagem;
    }
}
```

Classes abstratas
Exemplo

```
public class Tv extends Eletrodomestico {
    private float canal;
    @Override
    public void ligar() {
        if (!isLigado()) {
            setLigado(true);
            System.out.println("Tv ligada.");
        }
    }
    @Override
    public void desligar() {
        if (isLigado()) {
            super.setLigado(false);
            System.out.println("Tv desligada.");
        }
    }
    public float getCanal() {
        return canal;
    }
    public void setCanal(float canal) {
        this.canal = canal;
    }
}
```

Classes abstratas
Exemplo – Atividade

- Implemente a classe Radio.
  - Radio é um Eletrodoméstico.
  - Quando um rádio for ligado:
    - uma frequência é sintonizada automaticamente, isto é, o atributo sintonia deve ser inicializado com um valor;
    - o painel do rádio deve acender (chamar o método **`acenderPainel()`**).
- Crie objetos da classe Tv e Radio e invoque seus métodos.
  - Onde esses objetos devem ser criados?

Exercício 01

- Suponha que, no exemplo anterior, todo objeto do tipo Eletrodoméstico deva emitir um som assim que ligado.
- Refatore as classes do projeto Aparelhos, de modo que o método emitirSom() seja invocado quando todo eletrodoméstico for ligado.
  - Nesse caso, para efeito de teste, emitirSom() pode imprimir uma mensagem de texto.

Exercício 02

- Crie novas subclasses que herdam de Eletrodoméstico.
- Crie objetos das novas subclasses e invoque seus métodos.

Classes abstratas
Resumindo

- Classes abstratas implementam parcialmente um tipo de dado ou entidade.
- Uma classe abstrata pode:
  - definir métodos com implementação (padrão), prontos para serem usados para quem herda;
  - declarar métodos sem implementação, para que as subclasses obrigatoriamente as implementem, de acordo com sua especialidade;
  - declarar variáveis e métodos privados, sobrescrever construtores, assim como uma classe comum.

Interfaces

- Uma interface é como se fosse uma classe 100% abstrata, ou seja, uma coletânea de métodos sem implementação.
- Em outras palavras, pode-se dizer que uma interface é como se fosse um “contrato” cujas cláusulas são as assinaturas de métodos.
- Toda a classe que implementar uma interface deve implementar todos os seus métodos.

Interfaces

- Em Java, uma classe pode ter uma única superclasse (herança simples), mas pode implementar várias interfaces (herança múltipla).
- Assim como o uso de classes abstratas, o uso de interfaces garante o polimorfismo.
- Uma interface não pode ser instanciada.
  - Uma interface deve ser implementada.

Interfaces
Exemplo (genérico)

```
public interface NomeInterface{

   tipo NOME_CONSTANTE = valor;

   public abstract void metodo1();
   public int metodo2();
   String metodo3(String parametro1);
}
```

- Em uma interface, os métodos são implicitamente públicos e abstratos.
  - Por isso, as palavras public e abstract podem ser omitidas.

Interfaces
Exemplo (genérico)

```
public interface NomeInterface{

    tipo NOME_CONSTANTE = valor;

    public abstract void metodo1();
    public int metodo2();
    String metodo3(String parametro1);
}
```

- Em uma interface, somente constantes podem ser declaradas.
  - As constantes devem ser inicializadas;
  - Toda constante é, implicitamente, *public*, *static* e *final*.

Interfaces
Exemplo (genérico)

```
public class MinhaClasse implements NomeInterface{
    @Override
    public void metodo1(){
        //TODO - implementar
    }
    @Override
    public int metodo2(){
        //TODO - implementar
    }
    @Override
    public String metodo3(String parametro1){
        //TODO - implementar
    }
}
```

- Toda a classe que implementar NomeInterface deve fornecer comportamento a todos os métodos da interface.

Interfaces

- O uso de Interfaces é útil em grandes projetos, para obrigar os desenvolvedores a seguir um padrão do projeto.
  - Padrão de assinatura dos métodos:
    - Nomenclatura padronizada;
    - Tipo de retorno;
- Em vez de programar para uma implementação, programar para uma interface.
  - Desse modo, o contrato deve ser seguido, isto é, todos os métodos da interface devem ser implementados.

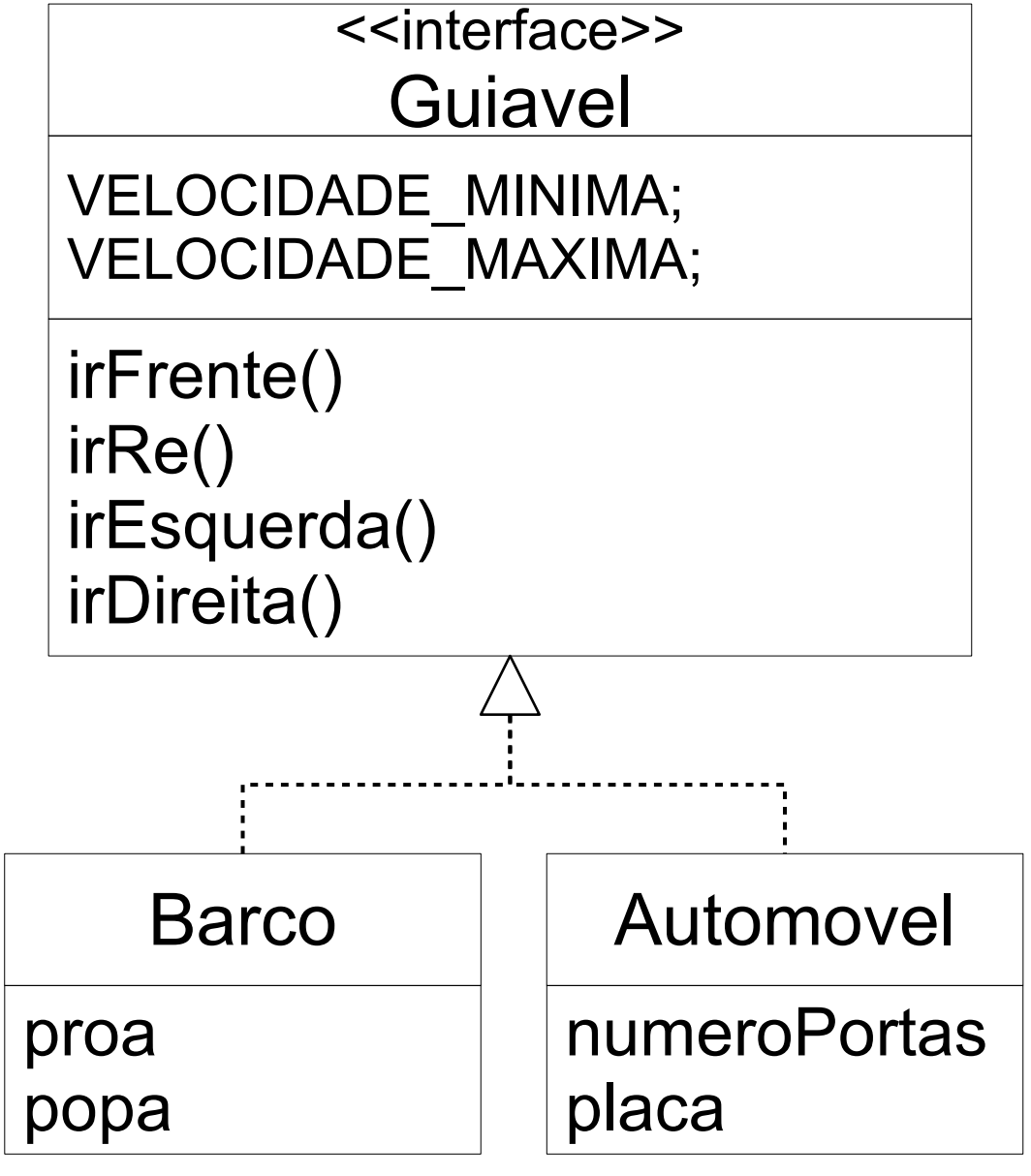
Interfaces
Exemplo

- Considere a seguinte interface:

```
public interface Guiavel{

   double VELOCIDADE_MINIMA = 10.0;
   double VELOCIDADE_MAXIMA = 300.0;

   void irFrente();
   void irRe();
   void irEsquerda();
   void irDireita();
}
```

- Podemos utilizar essa interface, para as seguintes classes:

Interfaces
Exemplo



A classe Automovel e a classe Barco implementam os comportamentos da interface Guiavel.

Portanto, em ambas as classes, devem ser implementados todos os métodos definidos na interface.

Interfaces
Exemplo

```
public class Automovel implements Guiavel{
    @Override
    public void irFrente(){
        System.out.println("Automovel em frente.");
    }
    @Override
    public void irRe(){
        System.out.println("Automovel dando re.");
    }
    @Override
    public void irEsquerda(){
        System.out.println("Automovel para esquerda.");
    }
    @Override
    public void irDireita(){
        System.out.println("Automovel para direita.");
    }
}
```

Atividade

- Crie um projeto chamado Locomocao.
  - Implemente as classes Automovel e Barco, e a interface Guiavel do exemplo anterior.
  - Crie objetos de Automovel e Barco e invoque seus métodos.
  - Na classe Barco, crie um método chamado isDeslocando() que retorna verdadeiro, se a velocidade do barco for maior que a velocidade mínima, e falso, caso contrário.
    - Utilize a constante da interface Guiavel para fazer a comparação.

Considerações finais

Classes abstratas x Interfaces

Característica	Classes abstratas	Interfaces
Herança	Uma classe pode herdar somente de uma classe abstrata (herança simples)	Uma classe pode implementar várias interfaces (herança múltipla)
Implementação padrão	Uma classe abstrata pode fornecer métodos com implementação e construtores	Uma interface determina somente assinaturas de métodos
Variáveis e constantes	Uma classe abstrata pode declarar variáveis e constantes	Uma interface pode declarar somente constantes.
Forma de uso	Uma classe concreta deve ser escrita para estender da classe abstrata	Qualquer classe pode implementar uma ou mais interface

Considerações finais

- Ambas classe abstrata e interface oferecem métodos abstratos para forçar outras classes a terem um mesmo conjunto de operações padronizadas.
- A necessidade de utilização de uma classe abstrata ou de uma interface depende do problema.

Considerações finais

- Uma classe abstrata deve ser criada quando o problema envolve relacionamentos de generalização do tipo É-UM.
  - Uma classe abstrata pode fornecer uma implementação padrão, caso não seja sobrescrita.
- Uma interface pode ser criada para definir protocolos, isto é, métodos padronizados.
  - Uma interface está focada sobre o quê, não interessa como as operações ou tarefas serão implementadas.