Universidade de Brasília Faculdade UnB Gama

Orientação a Objetos

Aula 12 - Interfaces

Daniel Porto

daniel.porto@unb.br

APRESENTAÇÃO

Interfaces

INTERFACES

Um tipo de **recurso especial** que só possui métodos abstratos e atributos finais ou static (constantes), são conhecidas como Interfaces em Java. Por meio destas é definido um padrão a ser implementado, além do caminho público para especificação do comportamento de classes (métodos).

Independente de suas posições hierárquicas, as interfaces implementam **comportamentos comuns** entre diferentes classes.

Cada classe pode implementar uma ou mais Interfaces, definindo implementações específicas para cada um de seus métodos abstratos que possuam assinaturas definidas dentro de suas respectivas Interfaces.

INTERFACES

Interfaces são contratos...

INTERFACES

Uma interface é usada quando existe a necessidade de implementação de métodos similares em classes não relacionadas. Exemplo: tratar o peso de Carro e Elefante.

Outra razão consiste no mecanismo existente em Java para trabalhar a **herança múltipla**, essencial na solução de alguns tipos de problemas computacionais.

Esta programação, usando interface, é realizada com emprego das instruções: interface (cria uma Interface) e implements (implementa em uma classe uma ou várias interfaces), por exemplo:

```
interface Tamanhos {
        :
}
```

```
public class Animal implements Tamanhos {
        :
}
```

INTERFACES

Características das Interfaces

Todas as **declarações internas** de uma interface são públicas, podendo serem usadas no pacote onde foram declaradas. Caso a interface também seja definida como pública, qualquer outro pacote poderá implementá-la, fazendo uso de seus componentes.

Atributos nas Interfaces:

- Atributos nas interfaces são definidos como constantes
- Implicitamente os atributos são sempre final e static, não sendo necessário o uso destas instruções em sua definição

INTERFACES

Métodos nas Interfaces:

- Métodos em uma interface são sempre públicos
- Todos são abstratos e possuem somente a definição de suas assinaturas
- **public e abstract**, geralmente não são usados na declaração de métodos em interface, pois já são implícitos
- Métodos de interface nunca podem ser **static**, pois indicam que são específicos de uma classe e nunca serão **abstract**
- Cada classe que implementa a interface deve implementar todos os seus métodos, definindo suas ações (seu corpo)
- As classes que implementam somente alguns dos métodos de uma interface devem ser declaradas como abstratas

INTERFACES

Classes Abstratas e Interfaces

A definição de uma **classe abstrata** pode conter métodos concretos (implementados) e abstratos (só assinaturas a serem implementadas em suas subclasses), enquanto que na **interface** todos os métodos só podem ser abstratos (só possui assinaturas a serem implementadas em suas subclasses).

Apesar de algumas diferenças, as interfaces possuem em comum, com as classes, 2 características importantes:

- ambas definem **tipos** a serem usados na programação
- classes e interfaces também podem declarar **novos métodos**, porém as interfaces não podem implementá-los, pois são abstratos (só assinaturas)

INTERFACES

A declaração de uma interface inclui um novo tipo, mas nele não existem valores, pois nas interfaces não são definidas, realmente, a implementação de nenhum método, sendo especificado apenas suas assinaturas.

A criação de objetos de classes, que implementem a interface, devem definir as ações de processamento que cada método realizará em seu acionamento.

A seguir, observe o exemplo que esta utilizando a classe abstrata Veiculo, bem encapsulada no exemplo de código anterior. No entanto, a subclasse Carro esta sendo elaborada novamente por receber modificações relevantes, incluindo a inclusão de uma interface.

INTERFACES

```java
// Inclua neste projeto a classe abstrata Veiculo

/** Síntese da Interface
 *     Atributo: limite
 *     Método: controlaVelocidade()
 */

public interface Rodovia {
    final float LIMITE = 120;   // atributo constante
    // todo método é abstrato na interface (assinatura)
    public boolean controlaVelocidade(float velocidade);
```

INTERFACES

```java
/**Síntese
 *   Atributos:
 *   Métodos: liga(), desliga(), getStatus() , mover()
 *         setStatus(boolean),controlaVelocidade(float)
 */
import javax.swing.JOptionPane;
public class Carro extends Veiculo implements Rodovia {
    private boolean status;

    public boolean getStatus() {
        return status;
        }
    public void setStatus(boolean status) {
        this.status = status;
    }
    public void liga() {
        status = true;
    }
    public void desliga() {
        status = false;
    }

    public int mover() {
        final String combustivel = "gasolina";
        return(JOptionPane.showConfirmDialog(null,
           "O veículo está abastecido com " +
            combustivel + "?","Abastecimento",
           JOptionPane.YES_NO_OPTION,
           JOptionPane.QUESTION_MESSAGE));
    }
```

INTERFACES

```
31      // continuação do exemplo anterior
32
33      // Implementação obrigatória do método da interface
34      public boolean controlaVelocidade(float velocidade) {
35          if(velocidade < LIMITE)
36              return true;
37          else
38              return false;
39      }
40  }
```

```
1   /** Síntese
2    *    Objetivo: movimenta um carro
3    *    Entrada:  situação da aceleração e combustível
4    *    Saída:    mostra movimentação do carro
5    */
6   import javax.swing.JOptionPane;
7   public class UsaNovoCarro {
8       public static void main(String[] args) {
9           Carro auto1 = new Carro();
10          auto1.setMarca("FORD");
11          auto1.setVelocidade(0);
12          auto1.setStatus(false);
```

INTERFACES

```
13          if(auto1.mover()==0) {
14              int continua;
15              auto1.liga();
16              auto1.acelera();
17              System.out.println("Carro em movimento");
18              mostraVelocidade(auto1);
19              do {
20                  for(int aux=0; aux<10; aux++) {
21                      auto1.acelera();
22                      mostraVelocidade(auto1);
23                  }
24                  System.out.println();
25                  continua = JOptionPane.showConfirmDialog(null,
26                          "Deseja continuar acelerando?",
27                           "Aceleração", JOptionPane.YES_NO_OPTION,
28                            JOptionPane.QUESTION_MESSAGE);
29              } while (continua == 0 &&
30                  auto1.controlaVelocidade(auto1.getVelocidade()));
31          }
```

INTERFACES

```
32              else {
33                  System.out.print("Velocidade = ");
34                  mostraVelocidade(auto1);
35                  JOptionPane.showMessageDialog(null, "Carro " +
36                          " precisa de gasolina para se  mover.",
37                          "Informação",JOptionPane.WARNING_MESSAGE);
38              }
39          }
40
41          public static void mostraVelocidade(Veiculo auto) {
42              System.out.print("\t" + auto.getVelocidade());
43          }
44      }
```

INTERFACES

Herança em Interfaces

Similar as classes, que podem herdar métodos e atributos, as interfaces também podem herdar assinaturas de métodos e constantes de outras interfaces, sendo esta herança indicada pela instrução **extends** entre interfaces.

Apesar das Interfaces não serem partes de uma hierarquia de classes, elas podem:

- relacionar-se entre si (estendendo uma interface - superinterface e subinterface) através da **extends**
- permitir que classes não relacionadas (sem herança direta) implementem a mesma interface

INTERFACES

Suponha a extensão da interface Rodovia para uma subinterface chamada Rua, onde o limite permitido é 50% da velocidade máxima definida na Rodovia.

Esta nova interface possuiria um novo método (só assinatura) que obrigaria suas classes implementadoras a averiguarem o local de movimento do Carro em questão (rodovia ou rua) para acompanhar seu limite.

```
public interface Rua extends Rodovia {
        public float ajustaLimite(float maxVelocidade);
}
```

O uso da nova interface exige alterações no projeto, mas permite que qualquer classe não relacionada com Carro ou mesmo Veiculo possam utilizá-las para análise de velocidade, mesmo não sendo relacionadas com este projeto.

INTERFACES

Novos Tipos

As classes derivadas e as interfaces implementadas em uma classe progenitora são coletivamente denominadas supertipos, enquanto as novas classes estendidas deste supertipo são um subtipo.

O tipo completo desta nova classe inclui todos os seus supertipos, de forma que uma referência a um de seus objetos pode ser utilizado polimorficamente, com referência a qualquer um de seus supertipos (classes ou interfaces).

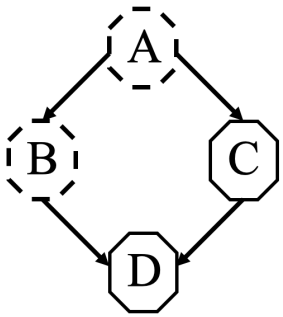
As definições de interface criam novos tipos, como ocorre com as classes, podendo estes tipos serem usados para definir variáveis, que recebem objetos que implementem estas interfaces (polimorfismo).

INTERFACES

Herança Múltipla em Java

Java disponibiliza a herança simples, que impede a implementação de alguns projetos que necessitam das características vinculadas a herança múltipla.

No entanto, Java incorpora a herança múltipla de forma alternativa, com o uso de interface, evitando um dos principais problemas de implementação deste tipo de herança, que pode ser representado no “losango da herança”.

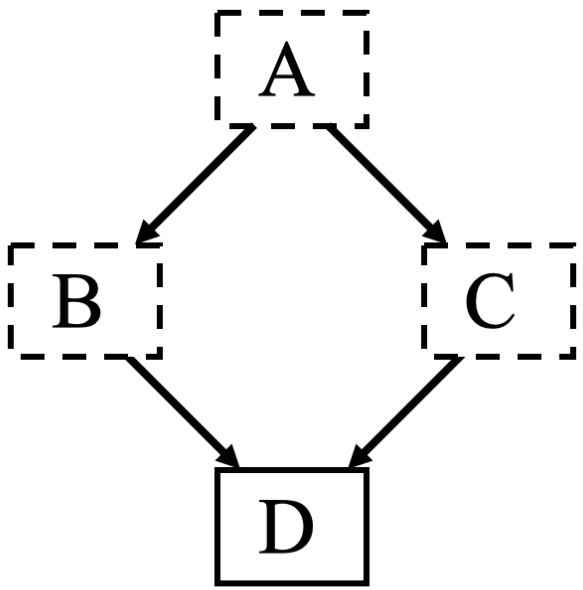


Suponha **A** e **B** interfaces que não possuem implementação e não geram conflitos sobre os componentes herdados para um objeto de **D** que ainda herda da classe **C**.

INTERFACES

Uma outra situação relacionada a este losango poderia ser a criação de uma nova classe D proveniente somente das interfaces A, B e C (interfaces gerando nova classe).

```
interface A {
        : // codificação adequada
}
interface B extends A {
        : // codificação adequada
}
interface C extends A {
        : // codificação adequada
}
class D implements B, C {
        : // codificação da classe
}
```



Assim, seria criada uma única classe (D) pertencente a hierarquia das classes, sendo ela baseada somente em interfaces.

INTERFACES

Diferentemente das classes, uma interface não possui uma **raiz**, como **Object** para classes, mas qualquer tipo de interface pode ser passada como parâmetro para um método que tenha um argumento do tipo **Object**.

Isso é possível porque a implementação da interface exigi a criação de um objeto e este deve ser de alguma classe para ser implementado.

Sendo todas as classes subclasses de **Object**, esta operação é válida e funcionará corretamente pelo polimorfismo.

INTERFACES

Uma implementação comum em Java, que possibilita a herança múltipla, consiste em classes derivadas que implementam uma ou mais interfaces, por exemplo:

```
public class Aluno extends Pessoa implements Frequencia, Avaliacao
{
        : // componentes e programação da classe
}
```

Aluno é subclasse direta de Pessoa e recebe sua herança.

Aluno implementa as interfaces Frequencia e Avaliacao em sua codificação (instruções dos métodos abstratos nas interfaces sendo definidos na própria classe Aluno).

Apesar de Frenquencia e Avaliacao não pertencerem a hierarquia de classes, Pessoa e sua nova subclasse Aluno pertencem.

INTERFACES

Quando Usar Interface

Definição de padrões de métodos para diferentes classes.

Implementação das características de herança múltipla.

A necessidade de extensão de uma classe, abstrata ou não, deve ser implementada como interface facilitando possíveis extensões.

As diferenças entre a implementação de classes ou interfaces, normalmente direcionam a escolha adequada de qual o melhor recurso a ser usado em uma implementação específica.

A classe abstrata permite o fornecimento de algumas ou todas implementações, facilitando sua herança, enquanto que as interfaces exigem seu repasse, por vezes tedioso e propenso a erros de implementação.

EXERCÍCIOS DE FIXAÇÃO

1) Implemente interfaces que sejam herdadas da superinterface Rodovia, apresentada nos exemplos desta aula. Depois evolua o exemplo da aula para permitir que um Carro qualquer, com dados informados pelo usuário, possa transitar por uma rodovia com limite já definido; por uma rua com 50% da velocidade limite da rodovia e por uma avenida com limite máximo de 80 km/h. Permita que o usuário cadastre um carro especifico e acelere o quanto quiser, podendo o mesmo abastecer o carro quando este estiver sem gasolina.

Faça uso de todos os conceitos de POO estudados até o momento e use a interface gráfica para interagir com o usuário como o exemplo de código anterior já vem fazendo.

EXERCÍCIOS DE FIXAÇÃO

2) Faça um programa que armazene dados de Empregados cadastrando sua matrícula funcional, nome completo e salário. Implemente também um controle para Terrenos na região, onde deverá ser armazenado os dados do endereço em uma única StringBuilder com o endereço completo, a área ocupada em um valor inteiro de metros quadrados e valor atual em reais (R$). Elabore uma interface, denominada Analise, que poderá verificar o menor e maior valor do tipo real, a existencia ou não de duplicidade entre valores inteiros, o somatório de qualquer tipo de dado numérico e a média de quaisquer valores numéricos.

Forneça opções de menu para o usuário aplicar esta interface e fazer análises correspondentes a qualquer empregado ou terreno, conforme ele deseje e faça tal análises somente por meio da interface Analise. Se o usuário quiser encerrar o programa sua solução deverá mostrar todos os dados cadastrados na console como um relatório tabelar.