Interfaces

Instituto Federal do Espirito Santo – Ifes
Campus Colatina – Programação Orientada a Objetos I
Professor: Giovany Frossard Teixeira

Herança Múltipla

- Herança Múltipla: é possível que uma classe descenda de várias classes. Imaginemos as seguintes classes:
  - Pessoa
  - Atleta
  - Nadador
  - Corredor
  - Ciclista
  - Triatleta

Um Atleta é uma Pessoa (Atleta extends Pessoa).

Um Nadador é um Atleta (Nadador extends Atleta).

Um Corredor é um Atleta (Corredor extends Atleta).

Um Ciclista é um Atleta (Ciclista extends Atleta).

Um Triatleta é um Nadador, Corredor e Ciclista (Triatleta extends Nadador, Corredor, Ciclista).

Herança Múltipla

- O detalhe é que Java não implementa herança múltipla (C++ por exemplo possui herança múltipla).

- Mas como Java pode trabalhar o comportamento da herança múltipla, sem herança múltipla ?
  - Através de Interfaces.

Interfaces

- **Interface:** é um conjunto de declaração de métodos, sem as respectivas implementações.

- Uma interface pode apenas conter:
  - Um identificador (nome)
  - Um conjunto de atributos public static final.
  - Um conjunto de métodos implicitamente (não precisa escrever) públicos e abstratos.

Interfaces

- Voltando ao nosso exemplo é necessário resolver o problema da herança múltipla da classe Triatleta.
- Através de interfaces é possível fazer o seguinte:
  - Podemos definir as interfaces:
    - Atleta
    - Nadador extends Atleta
    - Corredor extends Atleta
    - Ciclista extends Atleta
  - Podemos definir as seguintes classes:
    - Pessoa
    - Triatleta extends Pessoa implements Nadador, Corredor, Ciclista

Obs1. Notar que é possível fazer herança de interfaces.

Obs2. O operador implements determina que a classe Triatleta implementa as interfaces Nadador, Corredor e Ciclista.

Interfaces – Implementando o exemplo

- Implementando as Interfaces:

```
public interface Atleta {
    public static final int i = 0;
    public abstract void aquecer();
}

public interface Nadador extends Atleta {
    public void nadar();
}
```

```
public interface Corredor extends Atleta {
    public void correr();
}

public interface Ciclista extends Atleta {
    public void correrDeBicicleta();
}
```

Interfaces – Implementando o exemplo

- Classe Pessoa

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

Interfaces – Implementando o exemplo

- Classe Triatleta

```
public class Triatleta extends Pessoa implements
      Nadador, Corredor, Ciclista {

        public Triatleta(String nome) {

          super(nome);

        }

        public void aquecer() {

          System.out.println(this.getNome()  + " está
         aquecendo");

        }

        public void nadar() {

          System.out.println(this.getNome() + " está
          nadando");

        }

        public void correr() {

          System.out.println(this.getNome() + " está
          correndo");

        }

        public void correrDeBicicleta() {

          System.out.println(this.getNome() + " está
         correndo de bike");

        }

}
```

Obs. É obrigatório implementar todos os métodos das interfaces listadas.

Interfaces – Implementando o exemplo

```
public class TesteInterfaces {

    public static void main(String[] args) {

        Triatleta triatleta = new Triatleta("Fulano");

        triatleta.aquecer();

        triatleta.nadar();

        triatleta.correrDeBicicleta();

        triatleta.correr();

        Ciclista ciclista = new Triatleta("Ciclano");

        ciclista.aquecer();

        ciclista.correrDeBicicleta();

        Nadador nadador = new Triatleta("Giovany");

        nadador.aquecer();

        nadador.nadar();

    }

}
```

- Obs. Apesar de sempre ser criado um Triatleta, nos casos onde são declaradas variáveis interfaces não é permitido acessar métodos que não sejam da respectiva interface escolhida.

Interfaces e Padrões de Projeto

- O uso de interfaces é recomendável no desenvolvimento de sistemas para fornecer um contexto menos acoplado e mais simplificado de programação.
- Vamos supor, por exemplo, que temos uma interface responsável pela comunicação com banco de dados, dessa forma, qualquer classe que implementar a interface responderá a todas as funcionalidades para acesso a banco.
- Suponhamos que um novo banco seja elaborado e que desejemos fazer a troca do banco antigo por esse banco novo, será necessário apenas elaborar a classe que implemente a interface de acesso a esse banco novo e ao invés de utilizarmos um objeto da classe antiga utilizaremos um objeto da nova classe elaborada.

Dúvidas?