Programação Orientada a Objetos

Prof. Dr. Alan Souza

alan.souza@unama.br

2020

Interface

- Não é interface de GUI. É a palavra-reservada Java `interface`;
- As interfaces se livram do losango mortal (herança múltipla) tornando todos os métodos abstratos (sem corpo);
- Portanto, a subclasse **É OBRIGADA** a implementar os métodos na primeira subclasse concreta;
- É criada para definir uma função que as classes possam desempenhar a sua maneira;
- Uma classe pode implementar várias interfaces;
- Quando bem utilizada, aumenta a flexibilidade do projeto.

Interface

- Criando uma interface - Exemplo 1

```
public interface Pet {
...
}
```

```
public interface Pet {
        public abstract void serAmigavel( );
        public abstract void brincar( );
}
```

  - Todos os métodos devem ser abstratos, portanto, DEVEM terminar com ponto e vírgula. **Eles não têm corpo! (possuem somente a assinatura)**

Interface

- Usando a interface criada - Exemplo 1

Herança

```
public class Cachorro extends Canino implements Pet {

    // erros acontecem, porque os métodos
    // PRECISAM ser implementados!!!

}
```

Interface

Interface

- Usando a interface criada (*continuação...*) - Exemplo 1

```
public class Cachorro extends Canino implements Pet {
        // atributos da classe Cachorro
        // métodos da classe Cachorro
        public void serAmigavel( ) {
                // implementação obrigatória...
        }
        public void brincar( ) {
                // implementação obrigatória...
        }
}
```

Interface

- Implementando várias interfaces - Exemplo 1

```
public class Cachorro extends Canino
                implements Pet, Farejador, Guia {

        ...

}
```

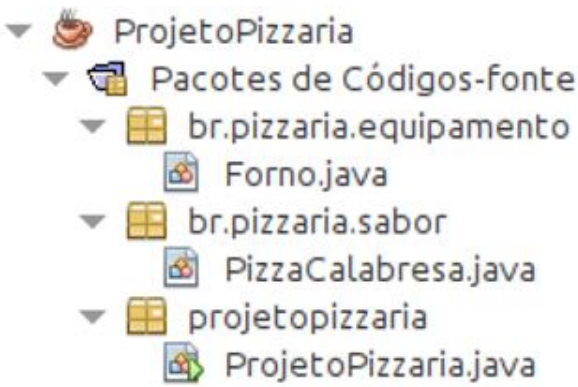
Interface

- **Exemplo 2 - Fazer junto com o professor!**

1. Criar um projeto no Netbeans. Nome: **ProjetoEx2Interface_TURMA**
2. Criar um pacote chamado **br.geometria.formas**;
3. Criar as classes **Quadrado, Retangulo, Circulo**, seus atributos, construtores e encapsulamento dentro desse pacote;
4. Criar a interface **ICalcGeometria** dentro desse pacote;
5. Dentro da interface, declarar os métodos abstratos **calcArea** e **calcPerimetro** que retornam um valor double;
6. Implementar a interface criada na classe Quadrado;
7. Implementar a interface criada na classe Retangulo;
8. Implementar a interface criada na classe Circulo;
9. Criar objetos na classe que contém o método main para testar o projeto e as interfaces.

Interface

- **Exemplo 3 - Projeto Pizzaria**



Interface

```
package br.pizzaria.sabor;
public class PizzaCalabresa {
    public void preparar() {
        System.out.println("molho, queijo, calabresa, cebola, tomate");
    }
    public void assar() {
        System.out.println("15 minutos");
    }
    public void cobrar() {
        System.out.println("R$ 32,00");
    }
}
```

Interface

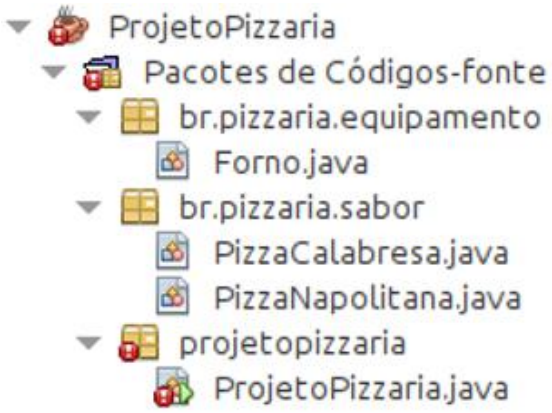
```
package br.pizzaria.equipamento;
import br.pizzaria.sabor.PizzaCalabresa;
public class Forno {
    public void fabricar(PizzaCalabresa calabresa) {
        calabresa.preparar();
        calabresa.assar();
        calabresa.cobrar();
    }
}
```

Interface

```
package projetopizzaria;
import br.pizzaria.equipamento.Forno;
import br.pizzaria.sabor.PizzaCalabresa;
public class ProjetoPizzaria {
    public static void main(String[] args) {
        Forno forno = new Forno();
        PizzaCalabresa calabresa = new PizzaCalabresa();
        forno.fabricar(calabresa);
    }
}
```

Interface

- **Exemplo 3 - Projeto Pizzaria**



Interface

```
package br.pizzaria.sabor;
public class PizzaNapolitana {
    public void preparar() {
        System.out.println("molho, queijo, presunto, tomate, orégano");
    }
    public void assar() {
        System.out.println("19 minutos");
    }
    public void cobrar() {
        System.out.println("R$ 38,00");
    }
}
```

Interface

```
package projetopizzaria;
import br.pizzaria.equipamento.Forno;
import br.pizzaria.sabor.*;
public class ProjetoPizzaria {
    public static void main(String[] args) {
        Forno forno = new Forno();
        PizzaCalabresa calabresa = new PizzaCalabresa();
        forno.fabricar(calabresa);
        PizzaNapolitana napolitana = new PizzaNapolitana();
        forno.fabricar(napolitana);
    }
}
```

Erro!

Interface

```
// package e import ocultados
public class Forno {
    public void fabricar(PizzaCalabresa calabresa) {
        calabresa.preparar();
        calabresa.assar();
        calabresa.cobrar();
    }
    public void fabricar(PizzaNapolitana napolitana) {
        napolitana.preparar();
        napolitana.assar();
        napolitana.cobrar();
    }
}
```

POLIMORFISMO (SOBRESCRITA)

Interface

Exemplo 3 - Projeto Pizzaria

Problemas:

- É necessário tornar o projeto mais flexível;
- Já pensou se o cardápio tiver 30 sabores de pizza?
  - Teria que criar 30 versões do método “fabricar” diferentes...
- Muito código duplicado;
- Manutenção prejudicada.

Interface

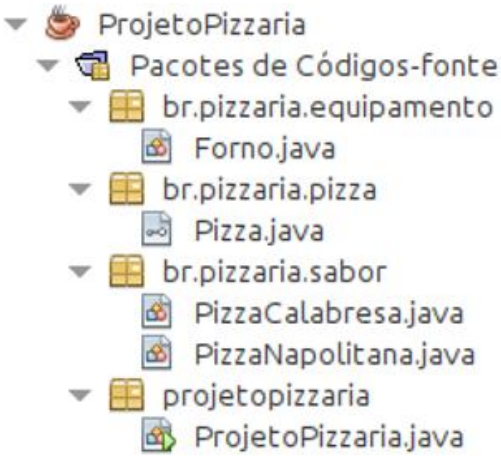
Exemplo 3 - Projeto Pizzaria

Soluções:

- Criar uma interface;
- Implementar a interface nas classes dos sabores da pizza;
- Sobrescrever os métodos;
- Alterar a classe Forno (“pulo do gato”).

Interface

- **Exemplo 3 - Projeto Pizzaria**



Interface

```
package br.pizzaria.pizza;

public interface Pizza {
    public void preparar();
    public void assar();
    public void cobrar();
}
```

Interface

```
package br.pizzaria.sabor;
import br.pizzaria.pizza.Pizza;
public class PizzaCalabresa implements Pizza {
    public void preparar() {
        System.out.println("molho, queijo, calabresa, cebola, tomate");
    }
    public void assar() {
        System.out.println("15 minutos");
    }
    public void cobrar() {
        System.out.println("R$ 32,00");
    }
}
```

Interface

```
package br.pizzaria.sabor;
import br.pizzaria.pizza.Pizza;
public class PizzaNapolitana implements Pizza {
    public void preparar() {
        System.out.println("molho, queijo, presunto, tomate, orégano");
    }
    public void assar() {
        System.out.println("19 minutos");
    }
    public void cobrar() {
        System.out.println("R$ 38,00");
    }
}
```

Interface

```
package br.pizzaria.equipamento;
import br.pizzaria.pizza.Pizza;

public class Forno {
    public void fabricar(Pizza pizza) {
        pizza.preparar();
        pizza.assar();
        pizza.cobrar();
    }
}
```

Interface

```
package projetopizzaria;
// imports ocultados
public class ProjetoPizzaria {
    public static void main(String[] args) {
        Forno forno = new Forno();
        PizzaCalabresa calabresa = new PizzaCalabresa();
        forno.fabricar(calabresa);

        PizzaNapolitana napolitana = new PizzaNapolitana();
        forno.fabricar(napolitana);
    }
}
```

Interface

- **Exemplo 4 - Continuação do Projeto Pizzaria**

a) Crie mais uma classe no projeto referente a um determinado tipo de pizza a sua escolha;
b) Implemente a interface Pizza;
c) Sobrescreva os métodos de acordo com a pizza que você escolheu;
d) Crie um objeto da classe da sua pizza no método main;
e) Chame o método de fabricação de pizza;
f) Execute o projeto.
g) Descreva o que torna esse projeto flexível e fácil de ser mantido.

Referência

Franzini, F. Pizzaria Polimórfica. Disponível em <https://fernandofranzini.wordpress.com/2010/07/07/pizzaria-polimorfica-2/>. Último acesso maio/2020. Publicado em julho/2010.