Orientação a Objetos 1

Interfaces

Prof. Dr. Vinícius Camargo Andrade

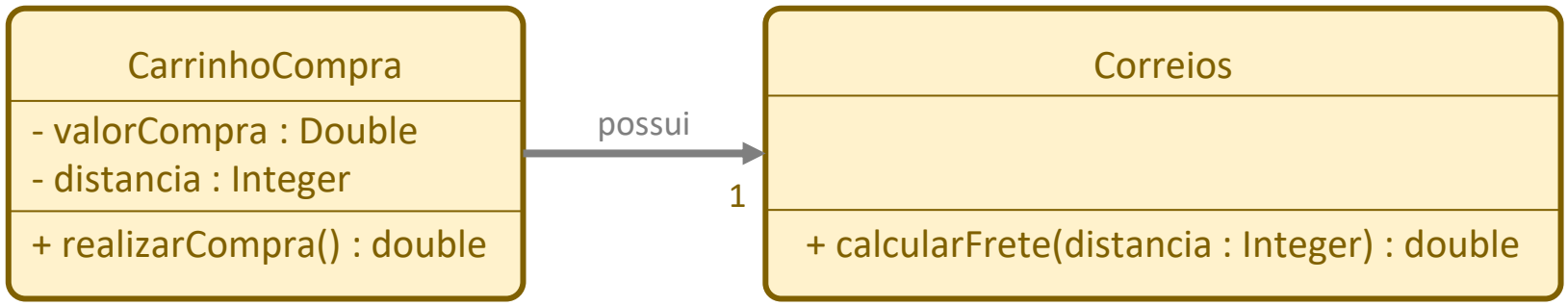
vcandrade@utfpr.edu.br

Departamento Acadêmico de Informática
Universidade Tecnológica Federal do Paraná

Cenário

Em um sistema de E-commerce, o carrinho de compras é formado pelo valor total dos produtos, distância da distribuidora até o endereço do cliente e empresa que efetuará a entrega, neste caso, Correios. Além disso, no carrinho de compras é possível realizar a compra, que efetuará a soma do valor total dos produtos com o frete, calculado pelos correios.

Cenário


CarrinhoCompra
- valorCompra : Double
- distancia : Integer
+ realizarCompra() : double
possui
1
Correios
+ calcularFrete(distancia : Integer) : double

```java
public class CarrinhoCompra {

    private Double valorCompra;
    private Integer distancia;
    private Correios correios;

    public CarrinhoCompra(Integer distancia, Double valorCompra) {

        this.valorCompra = valorCompra;
        this.distancia = distancia;
        this.correios = new Correios();
    }

    public double realizarCompra() {

        double valorFrete = this.getCorreios().calcularFrete(this.getDistancia());

        return this.getValorCompra() + valorFrete;
    }
```

```java
public class Correios {

    public double calcularFrete(Integer distancia) {

        return distancia * 0.35 + 13;
    }
}
```

Cenário

```
public class CarrinhoCompraTeste {

    public static void main(String[] args) {

        CarrinhoCompra carrinhoCompra = new CarrinhoCompra(130, 100.00);
        System.out.println("Valor Total da Compra: R$ " + carrinhoCompra.realizarCompra());
    }
}
```

```
run:
Valor Total da Compra: R$ 158.5
CONSTRUÍDO COM SUCESSO (tempo total: 1 segundo)
```

Problema

Problema

A classe CarrinhoCompra está altamente dependente da classe Correios, neste caso, diz-se que há um alto acoplamento entre as classes.

Problema

Isso é um problema pois este E-commerce não poderá ter outro modo de entregar suas mercadorias, a não ser pelo serviço dos Correios.

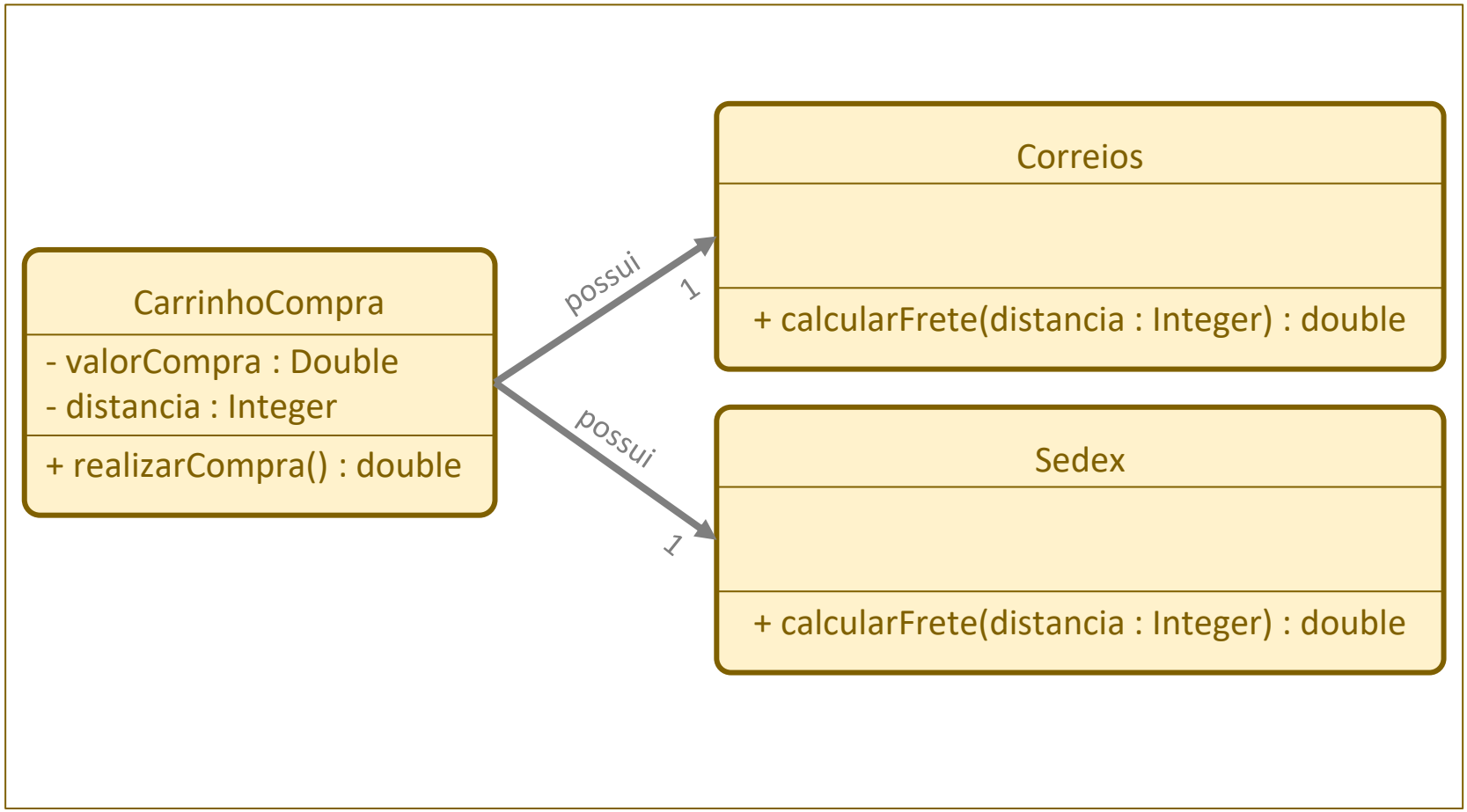
Exemplo

Problema

Exemplo

Além dos Correios, agora o E-commerce quer disponibilizar outro serviço de entrega, por exemplo, Sedex. Como implementar?

Exemplo


CarrinhoCompra
- valorCompra : Double
- distancia : Integer
+ realizarCompra() : double
possui
1
possui
1
Correios
+ calcularFrete(distancia : Integer) : double
Sedex
+ calcularFrete(distancia : Integer) : double

Exemplo

```
public class Sedex {

    public double calcularFrete(Integer distancia) {

        return distancia * 0.45 + 15;
    }
}
```

```
public class Correios {

    public double calcularFrete(Integer distancia) {

        return distancia * 0.35 + 13;
    }
}
```

```java
public class CarrinhoCompra {

    private Double valorCompra;
    private Integer distancia;
    private Correios correios;
    private Sedex sedex;

    public CarrinhoCompra(Integer distancia, Double valorCompra) {

        this.valorCompra = valorCompra;
        this.distancia = distancia;
        this.correios = new Correios();
        this.sedex = new Sedex();
    }

    public double realizarCompra() {

        double valorFreteCorreios = this.getCorreios().calcularFrete(this.getDistancia());
        double valorFreteSedex    = this.getSedex().calcularFrete(this.getDistancia());

        return this.getValorCompra() + valorFrete;
    }
```

```java
public class CarrinhoCompra {

    private Double valorCompra;
    private Integer distancia;
    private Correios correios;
    private Sedex sedex;

    public CarrinhoCompra(Integer distancia, Double valorCompra) {

        this.valorCompra = valorCompra;
        this.distancia = distancia;
        this.correios = new Correios();
        this.sedex = new Sedex();
    }

    public double realizarCompra() {

        double valorFreteCorreios = this.getCorreios().calcularFrete(this.getDistancia());
        double valorFreteSedex    = this.getSedex().calcularFrete(this.getDistancia());

        return this.getValorCompra() + valorFrete;
    }
```

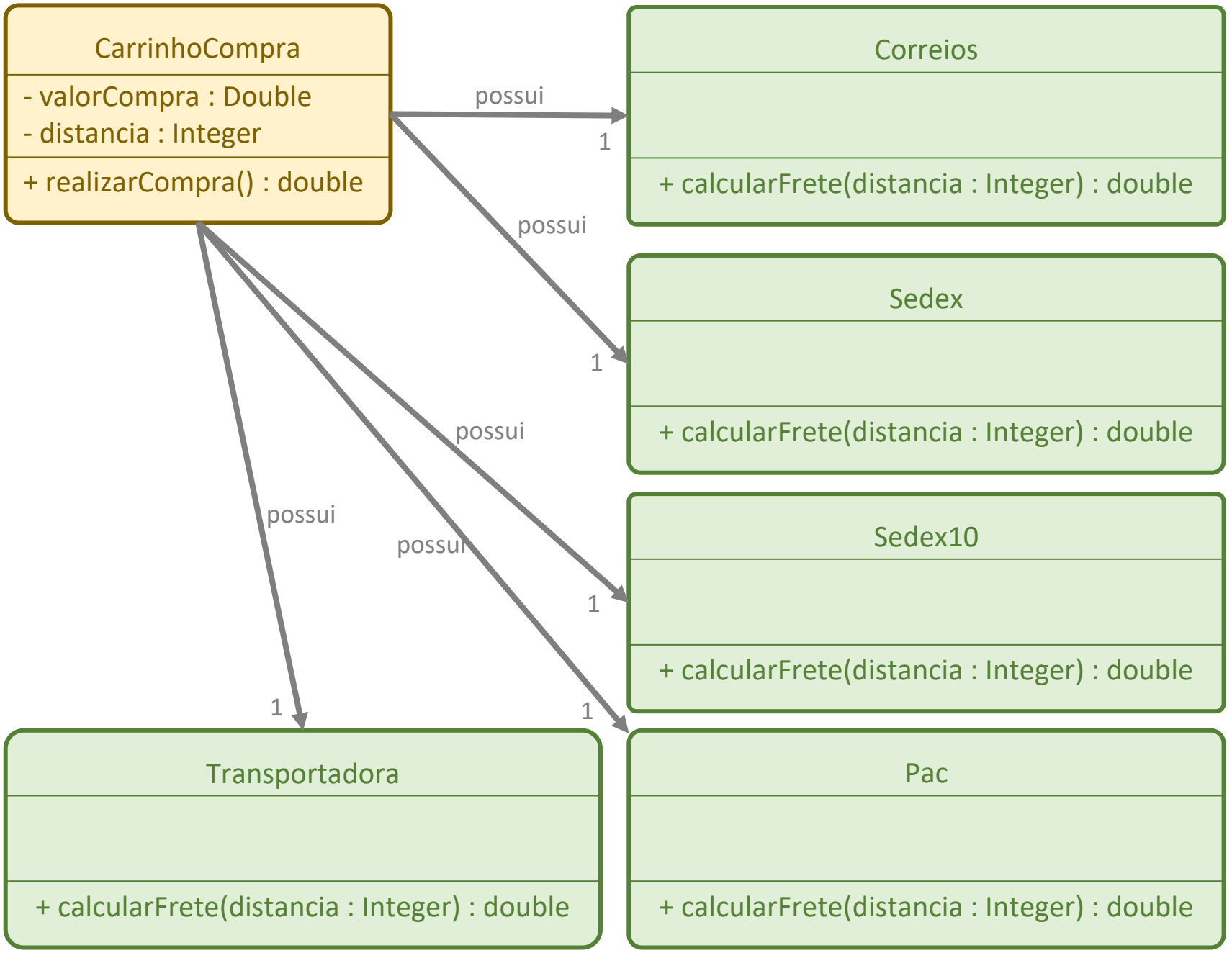
Problema

Como escolher o qual tipo de frete calcular?

```
public class CarrinhoCompra {

    private Double valorCompra;
    private Integer distancia;
    private Correios correios;
    private Sedex sedex;

    public CarrinhoCompra(Integer distancia, Double valorCompra) {

        this.valorCompra = valorCompra;
        this.distancia = distancia;
        this.correios = new Correios();
        this.sedex = new Sedex();
    }

    public double realizarCompra() {

        double valorFreteCorreios = this.getCorreios().calcularFrete(this.getDistancia());
        double valorFreteSedex    = this.getSedex().calcularFrete(this.getDistancia());

        return this.getValorCompra() + valorFrete;
    }
```

Problema

E se, além dos Correios e Sedex, o E-commerce decide disponibilizar mais serviços de fretes, como por exemplo, Sedex10, PAC, transportadoras particulares, entre outros?


CarrinhoCompra
- valorCompra : Double
- distancia : Integer
+ realizarCompra() : double
Correios
+ calcularFrete(distancia : Integer) : double
Sedex
+ calcularFrete(distancia : Integer) : double
Sedex10
+ calcularFrete(distancia : Integer) : double
Transportadora
+ calcularFrete(distancia : Integer) : double
Pac
+ calcularFrete(distancia : Integer) : double
possui
1
possui
1
possui
1
possui
1
possui
1

Problema

```java
public class CarrinhoCompra {

    private Double valorCompra;
    private Integer distancia;
    private Correios correios;
    private Sedex sedex;
    private Sedex10 sedex10;
    private Pac pac;
    private Transportadora transportadora;

    public CarrinhoCompra(Integer distancia, Double valorCompra) {

        this.valorCompra = valorCompra;
        this.distancia = distancia;
        this.correios = new Correios();
        this.sedex = new Sedex();
        this.sedex10 = new Sedex10();
        this.pac = new Pac();
        this.transportadora = new Transportadora();
    }
```

Solução

Solução

Para resolver esse problema, deve-se promover o desacoplamento (independência) entre as classes e aumentar o nível de flexibilidade do sistema, ou seja, permitir que novos serviços (classes) de fretes sejam incorporados sem grandes modificações no código já escrito.

Solução

Para isso, pode-se utilizar o conceito de Realização. Na linguagem Java, conhecido como Interface.

Interface

Associação de Realização

Interface

É um tipo que define um conjunto de operações (métodos) que uma classe deve implementar.

Interface

A interface estabelece um contrato que a classe deve cumprir.

Objetivos

Objetivos

As interfaces propiciam a criação de sistemas com baixo acoplamento e flexíveis.

Exemplo

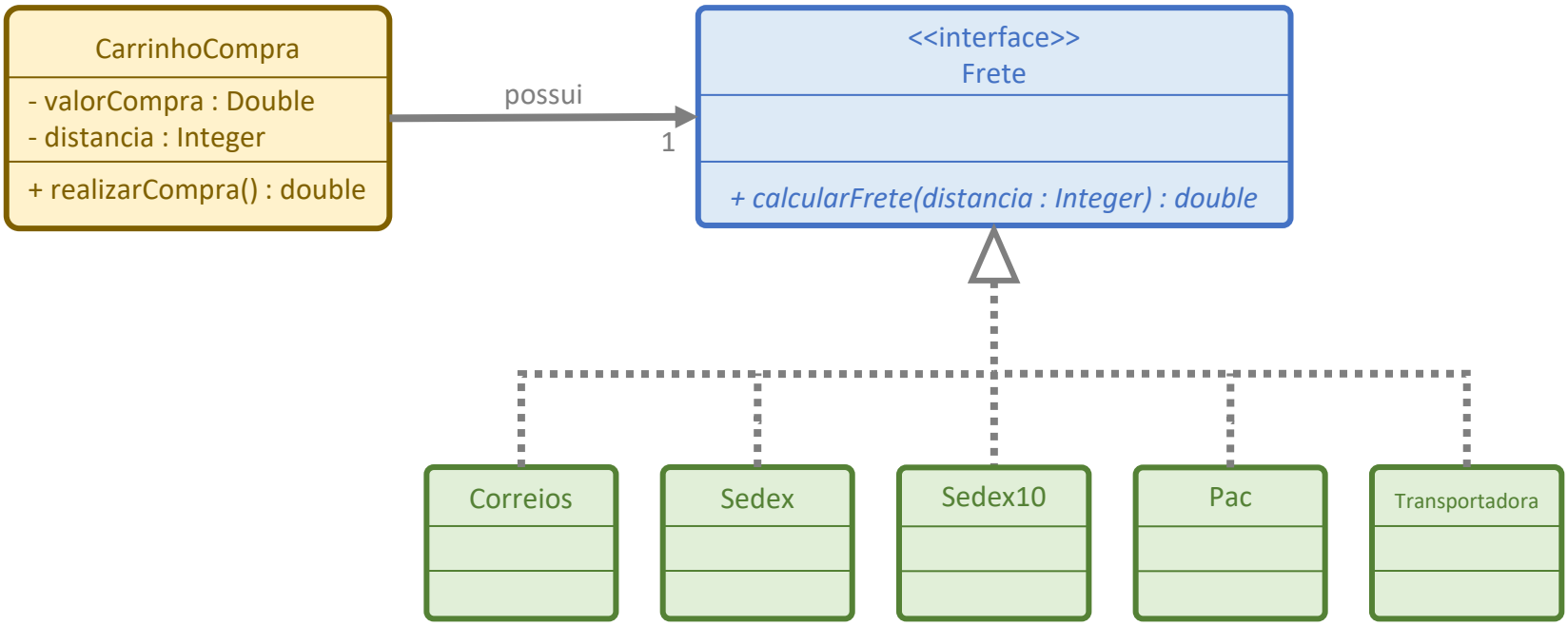
Exemplo

Considerando o exemplo abordado anteriormente. Em vez do objeto da classe CarrinhoCompra possuir uma instância da classe Correios, o que gera um forte acoplamento, a classe CarrinhoCompra vai possuir uma interface genérica, neste caso, chamada de Frete.

Exemplo

Na interface Frete serão definidos os métodos abstratos que deverão ser implementados pelas classes de frete (Correios, Sedex, Sedex10, Pac, Transportadora, entre outras)

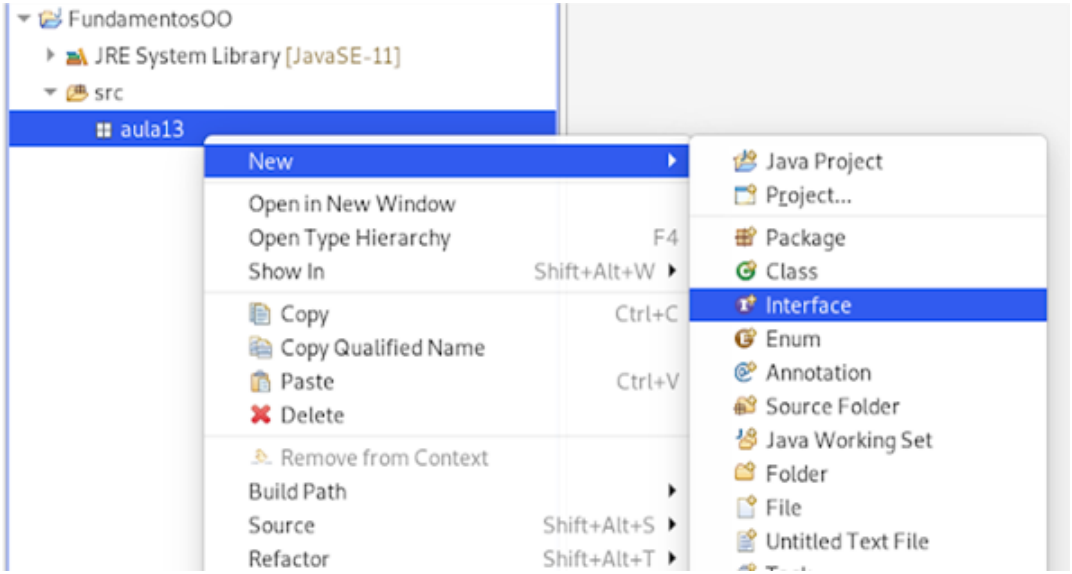
Cenário


CarrinhoCompra
- valorCompra : Double
- distancia : Integer
+ realizarCompra() : double
possui
1
<<interface>>
Frete
+ calcularFrete(distancia : Integer) : double
Correios
Sedex
Sedex10
Pac
Transportadora

Implementação

Implementação

Para criar uma interface, clique com o botão direito no pacote e selecione: New > Interface



Implementação

Todos os métodos declarados em uma interface são implicitamente abstratos e públicos.

```
public interface Frete {

    public double calcularFrete(Integer distancia);

}
```

Implementação

Na linguagem Java, para que uma classe implemente os métodos de uma interface, usa-se a palavra reservada implements na assinatura da classe.

```java
public class Correios implements Frete {

    @Override
    public double calcularFrete(Integer distancia) {

        return distancia * 0.35 + 13;
    }
}
```

```java
public class Sedex implements Frete {

    @Override
    public double calcularFrete(Integer distancia) {

        return distancia * 0.45 + 15;
    }
}
```

```java
public class Sedex10 implements Frete {

    @Override
    public double calcularFrete(Integer distancia) {

        return distancia * 0.48 + 16;
    }
}
```

```java
public class Pac implements Frete {

    @Override
    public double calcularFrete(Integer distancia) {

        return distancia * 0.30 + 10;
    }
}
```

Implementação

```
public class CarrinhoCompra {

    private Double valorCompra;
    private Integer distancia;
    private Frete frete;

    public CarrinhoCompra(Integer distancia, Double valorCompra, Frete frete) {

        this.valorCompra = valorCompra;
        this.distancia = distancia;
        this.frete = frete;
    }

    public double realizarCompra() {

        double valorFrete = this.getFrete().calcularFrete(this.getDistancia());

        return this.getValorCompra() + valorFrete;
    }
```

Assim, a classe CarrinhoCompra possui uma referência para a interface, que pode ser qualquer objeto concreto (Correios, Sedex, Sedex10, Pac ou Transportadora) que implementa a interface Frete.

```
public class CarrinhoCompra {

    private Double valorCompra;
    private Integer distancia;
    private Frete frete;

    public CarrinhoCompra(Integer distancia, Double valorCompra, Frete frete) {

        this.valorCompra = valorCompra;
        this.distancia = distancia;
        this.frete = frete;
    }

    public double realizarCompra() {

        double valorFrete = this.getFrete().calcularFrete(this.getDistancia());

        return this.getValorCompra() + valorFrete;
    }
```

Perguntas

Perguntas

- *Que objeto concreto o objeto frete irá receber?*
- *Qual classe irá instanciar o objeto concreto?*

```
public class CarrinhoCompra {

    private Double valorCompra;
    private Integer distancia;
    private Frete frete;

    public CarrinhoCompra(Integer distancia, Double valorCompra, Frete frete) {

        this.valorCompra = valorCompra;
        this.distancia = distancia;
        this.frete = frete;
    }

    public double realizarCompra() {

        double valorFrete = this.getFrete().calcularFrete(this.getDistancia());

        return this.getValorCompra() + valorFrete;
    }
```

Respostas

A classe responsável por instanciar a classe CarrinhoCompra deve, no momento da instanciação, enviar uma instância concreta de frete, como por exemplo, correios, sedex, sedex10, pac ou transportadora.

Respostas

Há duas técnicas utilizadas:

- *Inversão de Controle;*
- *Injeção de Dependência.*

Inversão de Controle

Consiste em retirar da classe a responsabilidade de instanciar suas dependências.

Injeção de Dependência

Maneira de realizar a inversão de controle: Um componente externo instancia a dependência, que é então injetada no objeto.

Implementação

Implementação

```
public class CarrinhoCompraTeste {

    public static void main(String[] args) {

        Correios correios = new Correios();

        CarrinhoCompra carrinhoCompra1 = new CarrinhoCompra(130, 100.00, correios);
        System.out.println("Valor Total da Compra: R$ " + carrinhoCompra1.realizarCompra());

        Sedex sedex = new Sedex();

        CarrinhoCompra carrinhoCompra2 = new CarrinhoCompra(130, 100.00, sedex);
        System.out.println("Valor Total da Compra: R$ " + carrinhoCompra2.realizarCompra());
    }
}
```

Implementação

```
public class CarrinhoCompraTeste {

    public static void main(String[] args) {

        Correios correios = new Correios();

        CarrinhoCompra carrinhoCompra1 = new CarrinhoCompra(130, 100.00, correios);
        System.out.println("Valor Total da Compra: R$ " + carrinhoCompra1.realizarCompra());

        Sedex sedex = new Sedex();

        CarrinhoCompra carrinhoCompra2 = new CarrinhoCompra(130, 100.00, sedex);
        System.out.println("Valor Total da Compra: R$ " + carrinhoCompra2.realizarCompra());
    }
}
```

Implementação

```
public class CarrinhoCompraTeste {

    public static void main(String[] args) {

        Correios correios = new Correios();

        CarrinhoCompra carrinhoCompra1 = new CarrinhoCompra(130, 100.00, correios);
        System.out.println("Valor Total da Compra: R$ " + carrinhoCompra1.realizarCompra());

        Sedex sedex = new Sedex();

        CarrinhoCompra carrinhoCompra2 = new CarrinhoCompra(130, 100.00, sedex);
        System.out.println("Valor Total da Compra: R$ " + carrinhoCompra2.realizarCompra());
    }
}
```

Otimização de Código

Otimização de Código

```
public class CarrinhoCompraTeste {

    public static void main(String[] args) {

        Correios correios = new Correios();

        CarrinhoCompra carrinhoCompra1 = new CarrinhoCompra(130, 100.00, correios);
        System.out.println("Valor Total da Compra: R$ " + carrinhoCompra1.realizarCompra());

        Sedex sedex = new Sedex();

        CarrinhoCompra carrinhoCompra2 = new CarrinhoCompra(130, 100.00, sedex);
        System.out.println("Valor Total da Compra: R$ " + carrinhoCompra2.realizarCompra());
    }
}
```

Otimização de Código

```
public class CarrinhoCompraTeste {

    public static void main(String[] args) {

        CarrinhoCompra carrinhoCompra1 = new CarrinhoCompra(130, 100.00, new Correios());
        System.out.println("Valor Total da Compra: R$ " + carrinhoCompra1.realizarCompra());

        CarrinhoCompra carrinhoCompra2 = new CarrinhoCompra(130, 100.00, new Sedex());
        System.out.println("Valor Total da Compra: R$ " + carrinhoCompra2.realizarCompra());
    }
}
```

Otimização de Código

```
public class CarrinhoCompraTeste {

    public static void main(String[] args) {

        CarrinhoCompra carrinhoCompra1 = new CarrinhoCompra(130, 100.00, new Correios());
        System.out.println("Valor Total da Compra: R$ " + carrinhoCompra1.realizarCompra());

        CarrinhoCompra carrinhoCompra2 = new CarrinhoCompra(130, 100.00, new Sedex());
        System.out.println("Valor Total da Compra: R$ " + carrinhoCompra2.realizarCompra());
    }
}
```

Exercício

Exercício

Desenvolva um sistema de tributos que possibilitará efetuar diferentes cálculos de impostos, como por exemplo: IRPF, INSS, IPVA e IOF. Para efetuar o cálculo dos impostos, o método de cálculo deve receber o valor do bem ou mercadoria, então o sistema deverá calcular o valor do imposto sobre este valor e retornar a soma do valor acrescido do imposto calculado. Os valores para cada imposto são:

- *IRPF: 7,5%;*
- *INSS: 11%;*
- *IPVA: 3%;*
- *IOF: 6%.*