

Vitor Dias de Albuquerque

Kauan Santos Oliveira

Padrões:

Strategy:

O strategy foi utilizado para escolher a forma de pagamento, ele foi utilizado nas classes dos 3 pagamentos, na interface pagamento, na main e no pedido.

Decorator: O decorator foi utilizado para colocar adicionais (add-ons) no sorvete, ele foi utilizado na main, nas classes das 3 coberturas e na filldecorator.

Main:

```
package com.company;

import com.company.coberturas.*;

import com.company.pagamentos.*;
import com.company.sorvetes.*;

public class Main {
    /** Rodar o programa */
    public static void main(String[] args) {

        IItem sorveteBaunilha = new SorveteBaunilha();
        IItem sorveteChocolate = new SorveteChocolate();
        IItem sorveteMorango = new SorveteMorango();

        sorveteMorango = new CoberturaChocolate(sorveteMorango);
        sorveteMorango = new CoberturaMorango(sorveteMorango);
        sorveteMorango = new CoberturaGranulado(sorveteMorango);

        sorveteChocolate = new
CoberturaChocolate(sorveteChocolate);
        sorveteChocolate= new CoberturaMorango(sorveteChocolate);
        sorveteChocolate= new
CoberturaGranulado(sorveteChocolate);

        sorveteBaunilha = new CoberturaChocolate(sorveteBaunilha);
        sorveteBaunilha= new CoberturaMorango(sorveteBaunilha);
        sorveteBaunilha= new CoberturaGranulado(sorveteBaunilha);

        Pedido pedidoun = new Pedido(sorveteMorango);
        Pedido pedidodois = new Pedido(sorveteChocolate);
        Pedido pedidotres = new Pedido(sorveteBaunilha);

        pedidoun.setPagamento(new Pix());
        pedidodois.setPagamento(new Dinheiro());
        pedidotres.setPagamento(new CartaoCredito());
    }
}
```

```

        pedidoun.pagar();
        pedidodois.pagar();
        pedidotres.pagar();
    }
}

```

Pedido:

```

package com.company;

/** Pedido utilizando */
public class Pedido {

    private IItem item;
    private IPagamento pagamento;

    public Pedido(IItem item){

        this.item = item;
    }

    public void setPagamento(IPagamento pagamento){
        this.pagamento = pagamento;
    }

    public void pagar(){

        System.out.println("O seu pedido ficou: "
+item.getDescription());
        System.out.println("Valor total: " +item.getPrice());
        pagamento.pagamento(item.getPrice());
    }

}

```

IPagamento:

```

package com.company;

/** Pagamento interface */
public interface IPagamento {

    void pagamento(double valor);
}

```

IItem:

```

package com.company;

/** Interface do item */
public interface IItem {

    String getDescription();
}

```

```
double getPrice();  
}
```

FillDecorator:

```
package com.company;  
/** Classe abstrata pro decorator */  
public abstract class FillDecorator implements IItem{  
  
    protected IItem FillItem;  
  
    public FillDecorator(IItem FillItem){  
        this.FillItem = FillItem;  
    }  
  
    @Override  
    public String getDescription() {  
        return FillItem.getDescription();  
    }  
  
    @Override  
    public double getPrice() {  
        return FillItem.getPrice();  
    }  
}
```

Sorvetes:

SorveteMorango:

```
package com.company.sorvetes;  
  
import com.company.IItem;  
/** Sorvete de morango */  
public class SorveteMorango implements IItem {  
  
    @Override  
    public String getDescription() {  
        return "Sorvete de Morango";  
    }  
  
    @Override  
    public double getPrice() {  
        return 8.00;  
    }  
}
```

SorveteChocolate:

```
package com.company.sorvetes;
```

```
import com.company.IItem;
/** Sorvete de chocolate */
public class SorveteChocolate implements IItem {

    @Override
    public String getDescription() {
        return "Sorvete de Chocolate";
    }

    @Override
    public double getPrice() {
        return 10.00;
    }

}
```

SorveteBaunilha:

```
package com.company.sorvetes;

import com.company.IItem;
/** Sorvete de baunilha */
public class SorveteBaunilha implements IItem {

    @Override
    public String getDescription() {
        return "Sorvete de Baunilha";
    }

    @Override
    public double getPrice() {
        return 7.00;
    }

}
```

Pagamentos:

Pix:

```
package com.company.pagamentos;

import com.company.IPagamento;

public class Pix implements IPagamento {

    /** Metodo de pagametro de boleto (strategy) */
    @Override
    public void pagamento(double valor) {
        System.out.println("Pagamento por pix");
    }

}
```

Dinheiro:

```
package com.company.pagamentos;

import com.company.IPagamento;

public class Dinheiro implements IPagamento {
    /** Metodo de pagamento de dinheiro (strategy) */
    @Override
    public void pagamento(double valor) {
        System.out.println("Pagamento por dinheiro");
    }
}
```

CartaoCredito:

```
package com.company.pagamentos;

import com.company.IPagamento;

public class CartaoCredito implements IPagamento {
    /** Metodo de pagamento de cartão de credito (strategy) */
    @Override
    public void pagamento(double valor) {
        System.out.println("Pagamento por Cartao de Credito");
    }
}
```

Coberturas:

CoberturaMorango:

```
package com.company.coberturas;

import com.company.FillDecorator;
import com.company.IItem;
/** Cobertura de Morango para o decorator */
public class CoberturaMorango extends FillDecorator {

    public CoberturaMorango(IItem FillItem) {
        super(FillItem);
    }

    /** Decorator*/
    @Override
    public double getPrice() {
        return FillItem.getPrice() + 1.50;
    }

    @Override
    public String getDescription() {
        return FillItem.getDescription() + " com cobertura de morango";
    }
}
```

CoberturaGranulado:

```
package com.company.coberturas;

import com.company.FillDecorator;
import com.company.IItem;
/** Cobertura de granulado para o decorator */
public class CoberturaGranulado extends FillDecorator {

    public CoberturaGranulado(IItem FillItem) {
        super(FillItem);
    }

    /** Decorator*/
    @Override
    public double getPrice() {
        return FillItem.getPrice() + 1.00;
    }

    @Override
    public String getDescription() {
        return FillItem.getDescription() + " com granulado";
    }
}
```

CoberturaChocolate:

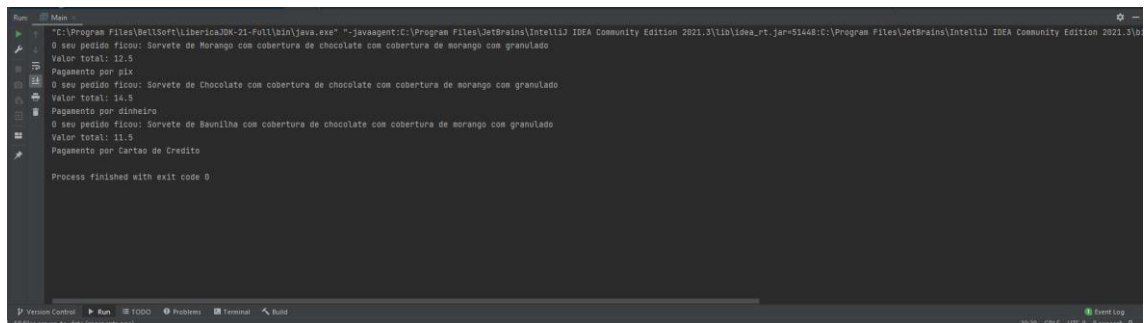
```
package com.company.coberturas;

import com.company.FillDecorator;
import com.company.IItem;
/** Cobertura de chocolate para o decorator */
public class CoberturaChocolate extends FillDecorator {

    public CoberturaChocolate(IItem FillItem) {
        super(FillItem);
    }
    /** Decorator*/
    @Override
    public double getPrice() {
        return FillItem.getPrice() + 2.00;
    }

    @Override
    public String getDescription() {
        return FillItem.getDescription() + " com cobertura de chocolate";
    }
}
```

Rodando:



```
Run - Main
"C:\Program Files\BellSoft\JDK-21-Full\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2021.3\lib\idea_rt.jar=51448:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2021.3\bin"
0 seu pedido ficou: Sorvete de Morango com cobertura de chocolate com cobertura de morango com granulado
Valor total: 12.5
Pagamento por pix
0 seu pedido ficou: Sorvete de Chocolate com cobertura de chocolate com cobertura de morango com granulado
Valor total: 14.5
Pagamento por dinheiro
0 seu pedido ficou: Sorvete de Baunilha com cobertura de chocolate com cobertura de morango com granulado
Valor total: 11.5
Pagamento por cartão de crédito
Process finished with exit code 0
```

Javadoc:

com.company

ArquivoC:/Users/dtu/Desktop/javadocfeito/com/company/package-summary.html

GmailYouTubeMaps

OVERVIEWPACKAGECLASS TREEINDEXHELP

PACKAGE DESCRIPTION | RELATED PACKAGES | CLASSES AND INTERFACES

SEARCHSearch

Package com.company

package com.company

Related Packages

Package	Description
com.company.coberturas	
com.company.pagamentos	
com.company.sorvetes	

All Classes and Interfaces

InterfacesClasses

Class	Description
FillDecorator	Classe abstrata pro decorator
Item	Interface do item
IPagamento	Pagamento interface
Main	
Pedido	Pedido utilizando

Professora verificou durante a aula.