Academia Xideral

Regina Rodriguez Campo Garrido

“Builder Pattern”

08/31/2024

Builder

The purpose of the builder method is to construct a complex object step-by-step, allowing for greater control over the object creation process and providing a flexible way to customize the object's attributes. Here I made an exercise where we used that method

A cafe where all beverages have the same characteristics, but each can be customized depending on the type of drink you choose.

First we created our class Drink

```
package builder;

import java.util.List;

public class Drink {

	private String type;
	private int price;
	private List<String> topping;
	private String extra;

	public String getType() {
		return type;
	}
	public void setType(String type) {
		this.type = type;
	}
	public int getPrice() {
		return price;
	}
	public void set Price(int price) {
		this. Price = price;
	}
	public List<String> getTopping() {
		return topping;
	}
	public void setTopping(List<String> topping) {
		this.topping = topping;
	}
	public String getExtra() {
		return extra;
	}
	public void setExtra(String extra) {
		this.extra = extra;
	}
	@Override
	public String toString() {
		return "Drink [type=" + type + ", price=" + price + ", topping=" +
topping + ", extra=" + extra + "]";
	}
}
```

Then we create our Builder interface which defines a set of methods that any builder class must implement.

```java
package builder;

import java.util.List;

public interface Builder {
        void reset();
        void setType(String type);
        void setPrice(int price);
        void setTopping(List <String> topping);
        void setExtra(String extra);

}
```

CoffeBuilder implements the Builder interface and constructs Drink objects.

```java
package builder;

import java.util.List;

public class CoffeBuilder implements Builder {

	private Drink coffebuilder;

	@Override
	public void reset() {
		this.coffebuilder = new Drink();

	}

	@Override
	public void setType(String type) {
		coffebuilder.setType(type);

	}

	@Override
	public void setPrice(int price) {
		coffebuilder.setPrice(price);

	}

	@Override
	public void setTopping(List <String> topping) {
		coffebuilder.setTopping(topping);

	}

	@Override
	public void setExtra(String extra) {
		coffebuilder.setExtra(extra);

	}


	public Drink getResult() {
		return this.coffebuilder;

	}
}
```

TeBuilder is similar to CoffeBuilder, but it is intended to build Drink objects of type tea. It implements the same methods as CoffeBuilder but uses a private instance Te to build the Drink object.

```java
package builder;

import java.util.List;

public class TeBuilder implements Builder {

	private Drink Te;

	@Override
	public void reset() {
		this.Te = new Drink();

	}

	@Override
	public void setType(String type) {
		Te.setType(type);

	}

	@Override
	public void setPrice(int price) {
		Te.setPrice(price);

	}

	@Override
	public void setTopping(List <String> topping) {
		Te.setTopping(topping);

	}

	@Override
	public void setExtra(String extra) {
		Te.setExtra(extra);

	}


	public Drink getResult() {
		return this.Te;

	}
}
```

The Director class directs the process of constructing different types of beverages.

Coffe configures a Builder object to create a "LATTE" coffee with the specified price and toppings.

Te configures a Builder object to create a "BLACK" tea with the specified price and toppings.

```
package builder;

import java.util.ArrayList;
import java.util.List;

public class Director {
        public void Coffe(Builder builder) {
                        builder.reset();
                builder.setType("LATTE");
                builder.setPrice(65);
                List<String> toppings = new ArrayList<>();
                toppings.add("Milk");
                toppings.add("Sugar");
                builder.setTopping(toppings);
                builder.setExtra("chocolate");
    }


    public void Te(Builder builder) {
                builder.reset();
                builder.setType("BLACK");
                builder.setPrice(45);
                List<String> toppings = new ArrayList<>();
                toppings.add("sugar");
                toppings.add("Lactosed milk");
                builder.setTopping(toppings);
                builder.setExtra("- -");
    }
}
```

Main:

```
package builder;

public class Client {
        public static void main(String[] args) {



                Director director = new Director();


        CoffeBuilder coffebuilder = new CoffeBuilder();
        TeBuilder tebuilder = new TeBuilder();
        director.Coffe(coffebuilder);
        director.Te(tebuilder);
        Drink drinkCoffe = coffebuilder.getResult();
        Drink drinkTe = tebuilder.getResult();

        System.out.println( drinkCoffe);
        System.out.println( drinkTe);
        }
}
```