Academia Xideral

Regina Rodriguez Campo Garrido

“Observer Pattern”

08/31/2024

Observer

Observer is a behavioral design pattern that lets you define a subscription mechanism to notify multiple objects about any events that happen to the object they're observing. Here we made an exercise:

A cosmetics company, launches new makeup products. The company wants to keep its customers informed whenever a new product becomes available. Customers can subscribe to receive notifications.

First, we create a method for the observers.

```java
package Observer;

public interface Observer {

    void update(); //method

}
```

We create the `Subject` class where we can subscribe and receive notifications about the makeup we are interested in.

```java
package Observer;

import java.util.ArrayList;

public abstract class Subject {

    List<Observer> observers = new ArrayList<>();

    void subscription(Observer o) {
        observers.add(o); //subscribe to makeup
    }

    void Cancel(Observer o) {
        observers.remove(o);
        }


    public void Notify() { //will send the notification
        for(Observer o: observers)
            o.update();
    }



}
```

We create different makeup classes that implement the `Observer` interface, as this will send notifications whenever there is something new.

```java
package Observer;

public class Base implements Observer {
    BaseBrand brand; //class atribute

    public Base(BaseBrand brand) {
        this.brand = brand;
    }


    void ShowBase() { //print the brand of the base
        System.out.println(brand);
    }


    @Override
    public void update() {
        ShowBase();  //sends the method to the observer

    }
}
```

```java
package Observer;

public class Lip implements Observer{

    LipBrand brand; //class atribute

    public Lip(LipBrand brand) {
        this.brand = brand;
    }


    void ShowLip() {//print the brand of the lip
        System.out.println(brand);
    }


    @Override
    public void update() {
         ShowLip();//print the brand of the base

    }


}
```

The `Makeup` class will send notifications to the clients.

```java
package Observer;


public class Makeup extends Subject{

    void Sub() {

        Notify();//send notification to observers
    }

}
```

Then, in our `main` method, we have a menu to choose the type of subscription we want so that we receive notifications about that makeup.

```java
package Observer;

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner number = new Scanner(System.in);
        System.out.println("ENTERTHE NUMBER OF SUBSCRIPTION THAT YOU WISH");
        System.out.println("1.- LIP");
        System.out.println("2.- BASE");

        int num = number.nextInt();

        switch(num) {
          case 1:
                Observer lip1 = new Lip(LipBrand.CHANEL);
                Makeup subs1 = new Makeup();
                subs1.subscription(lip1);
                System.out.println("The lipstick that just arrived is: ");
                subs1.Sub();


            break;

          case 2:
              Observer Base2 = new Base(BaseBrand.FOUNDATION);

                Makeup subs2 = new Makeup();
                subs2.subscription(Base2);
                System.out.println("The Base that just arrived is: ");
                subs2.Sub();
            break;


        }



    }

}
```