Academia Xideral

Regina Rodriguez Campo Garrido

# “Decorator Pattern”

08/31/2024

# Decorator

Decorator is a structural design pattern that lets you attach new behaviors to objects by placing these objects inside special wrapper objects that contain the behaviors.

A spa chain offers various basic. We wants to allow customers to customize these services with additional add-ons. Each add-on comes with an additional cost.

We create our client class with the attributes we want

```java
package Decorate;

public class Client {

    private String name;
    private int cost;

    public Client(String name) { //constructor
        super();
        this.name = name;

    }

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getCost() {
        return cost;
    }
    public void setCost(int cost) {
        this.cost = cost;
    }


}
```

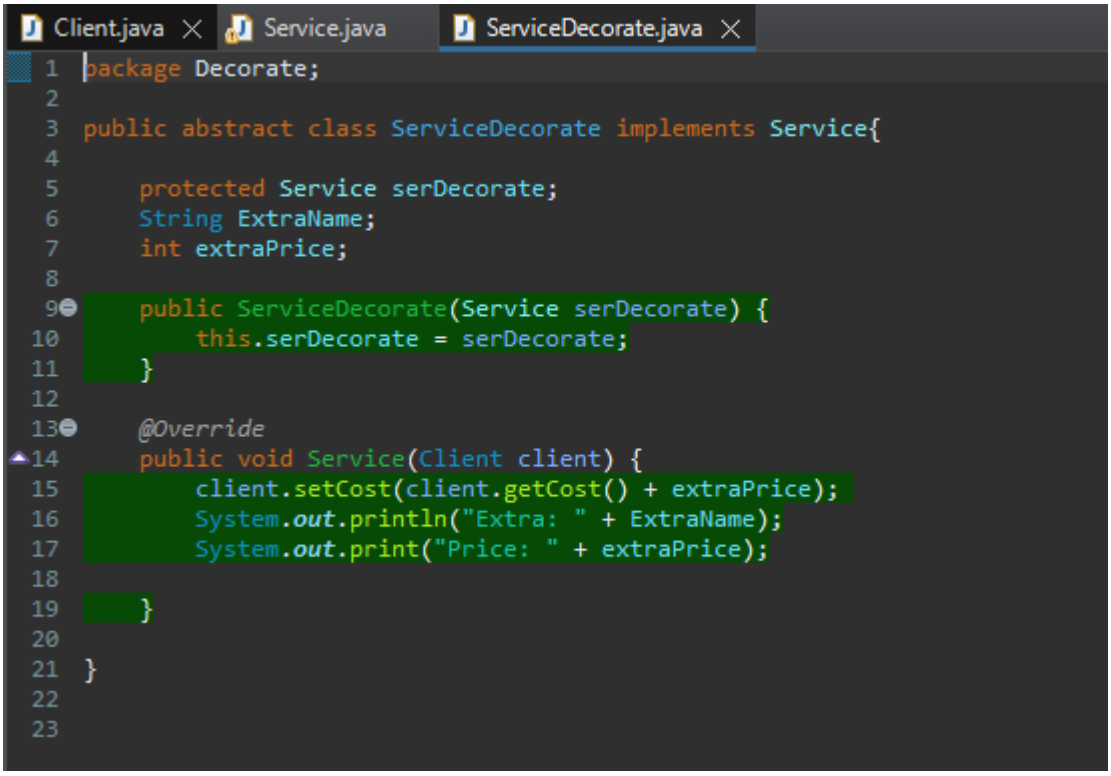
The Service interface defines a method. This method is intended to be

Implemented by classes that provide different types of services.

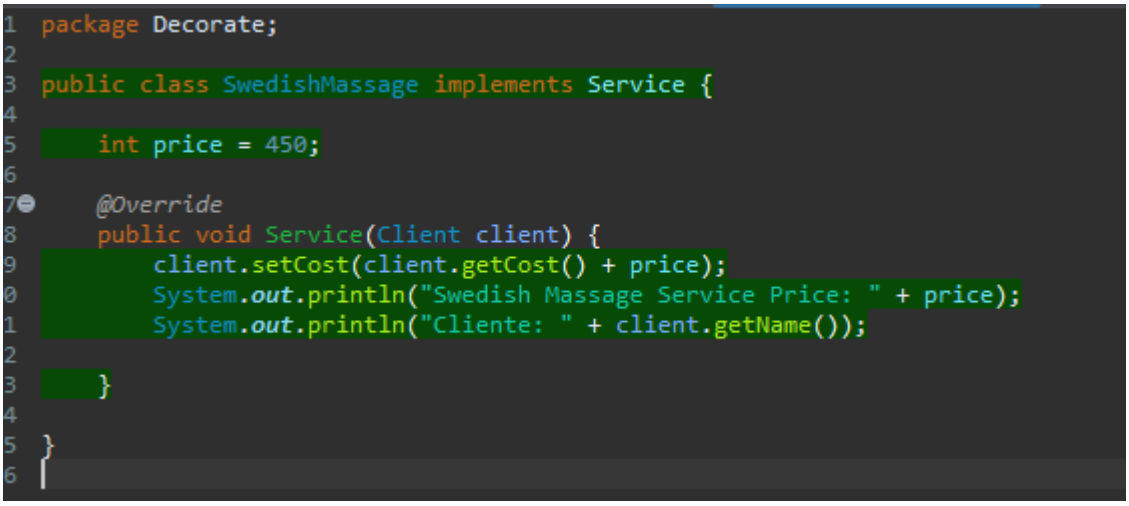
```java
package Decorate;

public interface Service {

    void Service(Client client);

}
```

The Service Decorate abstract class implements the Service

Interface and serves as a base for concrete decorators.




We create the classes that will be the spa services. Those The classes implements the Service interface and represents a concrete service.



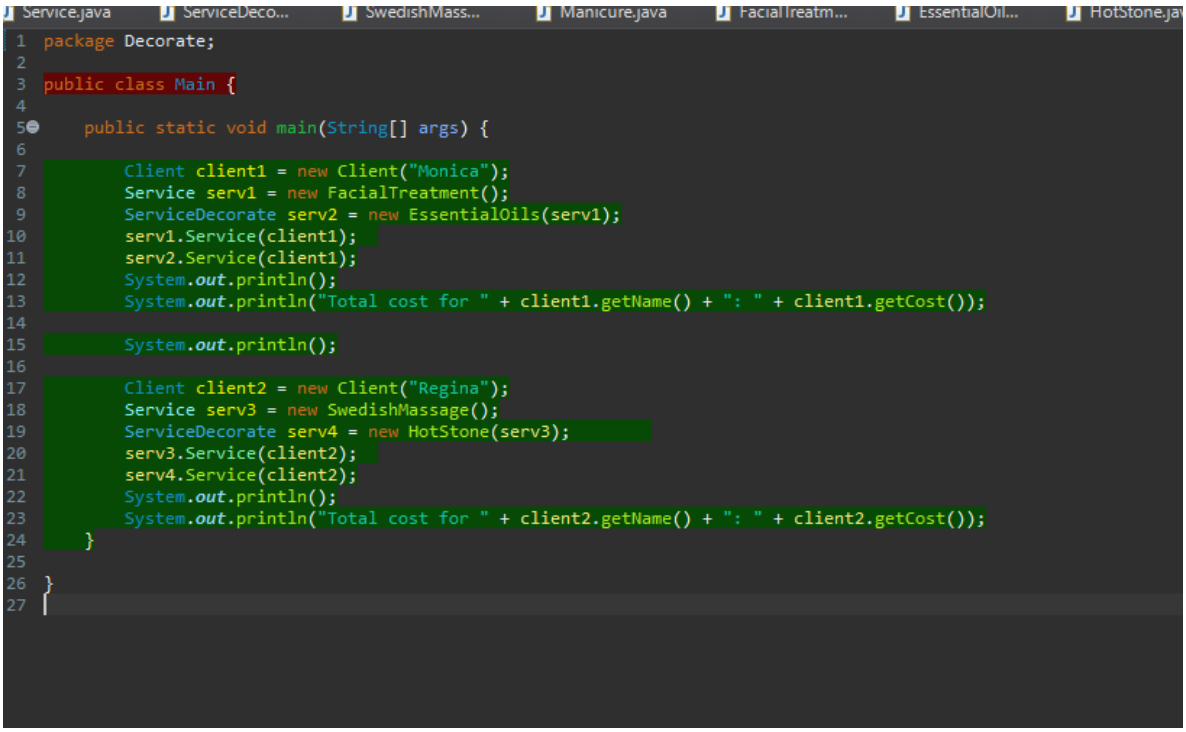
```java
package Decorate;

public class FacialTreatment implements Service{

    int price = 500;

    @Override
    public void Service(Client client) {
        client.setCost(client.getCost() + price);
        System.out.println("Facial Treatment Service Price: " + price);
        System.out.println("Client: " + client.getName());

    }

}
```

Create extra class extends Service Decorate and represents a concrete decorator that adds extra features to the base service.

```java
package Decorate;

public class EssentialOils extends ServiceDecorate {

    public EssentialOils(Service serviceDecorate) {
        super(serviceDecorate);
        ExtraName = "Essential Oils";
        extraPrice = 135;
    }
}
```

```java
package Decorate;

public class HotStone extends ServiceDecorate {

    public HotStone(Service serviceDecorate) {
        super(serviceDecorate);
        ExtraName = "Essential Oils";
        extraPrice = 50;
    }
}
```

We create in main where the client makes the packages of the services they want.




```java
package Decorate;

public class Main {

    public static void main(String[] args) {

        Client client1 = new Client("Monica");
        Service serv1 = new FacialTreatment();
        ServiceDecorate serv2 = new EssentialOils(serv1);
        serv1.Service(client1);
        serv2.Service(client1);
        System.out.println();
        System.out.println("Total cost for " + client1.getName() + ": " + client1.getCost());

        System.out.println();

        Client client2 = new Client("Regina");
        Service serv3 = new SwedishMassage();
        ServiceDecorate serv4 = new HotStone(serv3);
        serv3.Service(client2);
        serv4.Service(client2);
        System.out.println();
        System.out.println("Total cost for " + client2.getName() + ": " + client2.getCost());
    }

}
```