Academia Xideral Java Monterrey

Daniel Ivan Anaya Alvarez

Activity Decorator

This activity consist of doing a program focused on the decorator methor, this method basically as its name refers adds decoration to a specific objects. The way it works is by defining the object in my case what I did was choosing a washer and put some related ítems it could have like soap and softener, also in a more cotidian way can have some coins on top.

Decorator.java

```
package ActDecorator;

public abstract class Decorator extends Component {

    public Decorator(Component component) {
        super(component);
    }
}
```

Component.java

```
package ActDecorator;

public abstract class Component {

    // Private attributes for encapsulation
    private Component component;
    String accessoryName;
    double accessoryPrice;
    String type; // Add type attribute

    // Constructors
    Component() {}

    Component(Component component) {
        this.component = component;
    }

    // Getter and setter for the component
    public Component getComponent() {
        return component;
    }

    public void setComponent(Component component) {
        this.component = component;
    }

    // Methods to get accessory name, total cost, and type
```

```java
    public String getAccessoryName() {
        if (component == null) {
            return accessoryName;
        } else {
            return component.getAccessoryName() + "\n" + accessoryName;
        }
    }

    public double getTotalCost() {
        if (component == null) {
            return accessoryPrice;
        } else {
            return component.getTotalCost() + accessoryPrice;
        }
    }

    public String getType() {
        if (component == null) {
            return "The type of accessory is " + type;
        } else {
            return component.getType() + "\n" + "The type of accessory is " +
type;
        }
    }

    // Setter for type
    public void setType(String type) {
        this.type = type;
    }
}
```

Washer.java

```java
package ActDecorator;

public class Washer extends Component {

    public Washer() {
        accessoryName = "Washer";
        accessoryPrice = 5000.0;
        type = "Cleaning";
    }

    @Override
    public String getAccessoryName() {
        return accessoryName;
    }

    @Override
    public double getTotalCost() {
        return accessoryPrice;
    }

    @Override
```

```
    public String getType() {
        return "The type of accessory is " + type;
    }
}
```

Coins.java

```
package ActDecorator;

public class Coins extends Decorator {

    public Coins(Component component) {
        super(component);
        accessoryName = "Coins";
        accessoryPrice = 20.5;
        type = "Currency";
    }
}
```

Soap.java

```
package ActDecorator;

public class Soap extends Decorator {

    public Soap(Component component) {
        super(component);
        accessoryName = "Soap";
        accessoryPrice = 50.5;
        type = "Cleaning";
    }
}
```

Softener.java

```
package ActDecorator;

public class Softener extends Decorator {

    public Softener(Component component) {
        super(component);
        accessoryName = "Softener";
        accessoryPrice = 50.5;
        type = "Cleaning";
    }
}
```

Principal.java

```java
package ActDecorator;

public class Principal {

    public static void main(String[] args) {
        Component component = new Soap(
            new Coins(
                new Softener(
                    new Washer()
                )
            )
        );

        System.out.println(component.getAccessoryName());
        System.out.println("Total Cost: " + component.getTotalCost());
        System.out.println(component.getType());
    }
}
```

Print

```
Console ×
<terminated> Principal (11) [Java Application] C:\Users\HP\.p2\pool\plugins\org.ecli
Washer
Softener
Coins
Soap
Total Cost: 5121.5
The type of accessory is Cleaning
The type of accessory is Cleaning
The type of accessory is Currency
The type of accessory is Cleaning
```

Conclusion

In my opinion this method was kind of confusing for me, I understand the point of the program but its something that I would like to avoid if possible. At the end it can prove its own usefullness like relating objects one to another and basically pointing a “main” object and its on “sub” objects