Facultatea Calculatoare, Informatica si
Microelectronica

Universitatea Tehnica a Moldovei

Analiza Proiectarea si Programarea
Orientata pe Obiecte

Lucrarea de laborator#2

Principiile SOLID

Autor:
Dumitrita Garaba
lector asistent:
Mihail Pecari

Lucrarea de laborator #2

0.1 Scopul lucrarii de laborator

Studierea a doua principii SOLID.

0.2 Obiective

- Single Responsability Principle
  - Interface Segregation Principle

0.3 Efectuarea lucrarii de laborator

0.3.1 Analiza lucrarii de laborator

In aceasta lucrare de laborator am elaborat o aplicatie in limbajul Java, IDEA-ul folosit fiind Intellij. Aplicatia creata reda o florarie online. Pe ramura master s-a elaborat aplicatia simpla, in branch1 aplicatia conform principiilor.

Scopul lucrarii de laborator a fost implementarea celor 2 principii SOLID: Single Responsability si Interface Segregation.

S-au creat 3 clase FlowerShop, Flowers si FlowerProcessing ce au cite o singura responsabilitate.

Listing 1: Clasa FlowerShop

```
package com.company;

import javax.imageio.ImageIO;
import javax.swing.*;
import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.File;

public class FlowerShop implements Shop {
    JFrame editorFrame;

    ImageIcon imageIcon;

    BufferedImage image;

    JLabel jLabel;

    @Override
    public void view() {

        editorFrame = new JFrame("MintFlower");
```

```
        editorFrame.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
        image=null;
        try{
            image = ImageIO.read(new File("D:/mintflower.jpg"));
        }
        catch (Exception e)
        {
            e.printStackTrace();
            System.exit(1);
        }
        imageIcon = new ImageIcon(image);
        jLabel = new JLabel();
        jLabel.setIcon(imageIcon);
        editorFrame.getContentPane().add(jLabel, BorderLayout.CENTER);
        editorFrame.pack();
        editorFrame.setLocationRelativeTo(null);
        editorFrame.setVisible(true);
    }
}
```

Listing 2: Clasa Flowers

```
package com.company;

class Flowers implements Flower{

    private int amount;
    private double price;
    private String name;
```

```
    public Flowers(int amount, double price, String name){

        this.amount=amount;
        this.price=price;
        this.name=name;

    }

    public double buyFlowers(FlowersPrice flowersPrice, int amount,
                             double userMoney){

       if(amount>this.amount){
       System.out.println("Nu avem at t de multe flori n stoc!");
             return 0;

            }
            int actualAmount=checkInventory(amount);

            double actualPrice=actualAmount*flowersPrice.calculate(this);

            if(userMoney<actualPrice){
      System.out.println("Nu ave i suficien i bani pentru a efectua
aceast tranzac ie!");
                return 0;

            }

            if(actualAmount>0){

                System.out.print("A i cump rat "+actualAmount+" flori de ");

                System.out.printf("%.2f",actualPrice);

                System.out.println("lei.");

                System.out.println();

                System.out.println("Acum sunt "+getAmount()+" flori n stoc.");

            }

            return actualPrice;

        }
```

```
    public double getPrice(){
        return this.price;
    }

    public int getAmount(){
        return this.amount;
    }
    public int checkInventory(int amount){
        int actualBought=0;
        for(int i=0;i<amount;i++){
            if(this.amount>0){
                this.amount--;
                actualBought++;
            }
        }
        if(this.amount==0){
            System.out.println("Aceste flori nu sunt  n stoc!");
        }
        return actualBought;
    }
}
```

Listing 3: Clasa FlowerProcessing

```
package com.company;
```

```
public class FlowerProcessing implements FlowersPrice {
    public double calculate(Flowers flowers){

        return flowers.getPrice();

    }
}
```

Sunt clienti care doresc si nu doresc sa vizualizeze poza florarii.
De aceea se creeaza doua interfete Flower si Shop.

Listing 4: Interfata Flower

```
package com.company;

public interface Flower {
   public double buyFlowers( FlowersPrice flowersPrice,
                             int amount, double userMoney);

}
```

Listing 5: Interfata Shop

```
package com.company;

public interface Shop {
    public void view();
}
```

0.3.2 Imagini

```
Cum vă numiți?
Fiodorov Ina
Câți bani doriți să cheltuiți?
200
Salut Fiodorov Ina! Bine ați venit la MintFlower! Cu ce vă pot ajuta?

Garaba Dumitrita are 0,00lei
Fiodorov Ina are 200,00lei

1-Cumpăr
2-Ies
1
Ce doriți să cumpărați?
1-Trandafiri
2-Lalele
3-Garoafe
4-Jasmin
5-Bujori
6-Un buchet
1
Câte doriți?
5
Nu aveți suficienți bani pentru a efectua această tranzacție!

Garaba Dumitrita are 0,00lei
Fiodorov Ina are 200,00lei

1-Cumpăr
2-Ies
2
Vă mulțumesc, vă mai așteptăm!
```

Figure 1: Aplicatia

Figure 2: Imaginea florariei

Concluzii

In aceasta lucrare de laborator mi-am dezvoltat abilitatile practice in aplicarea a doua din principiile SOLID-Single Responsability si Interface Segregation.

In contextul principiului Singurei Responsabilitati prin responsabilitate se intelege un motiv de a modifica . Daca pot fi gasite mai multe motive pentru a modifica o clasa, inseamna ca acea clasa are mai multe responsabilitati. Acest principiu spune faptul ca o clasa nu trebuie sa aiba mai multe responsabilitati fiindca orice modificare la nivelul cerintelor se reflecta printr-o modificare la nivelul uneia sau mai multor responsabilitati care se propaga mai departe la nivelul claselor. Astfel, daca o clasa implementeaza mai multe responsabilitati automat pentru acea clasa la un moment dat va exista mai mult de un motiv pentru a fi modificata.

Principiul Segregarii Interfetei scoate in evidenta faptul ca atunci cind se defineste o interfata trebuie de avut grija ca doar acele metode care sunt specifice interfetei sa fie puse in interfata. Daca intr-o interfata sunt adaugate metode care nu au ce cauta acolo, atunci clasele care implementeaza interfata vor trebui sa implementeze si acele metode.

Bibliography

[1] https://www.codecademy.com/learn/learn-java

[2] http://www.learnjavaonline.org/

[3] https://scotch.io/bar-talk/s-o-l-i-d-the-first-five-principles-of-object-oriented-desig

[4] https://zeroturnaround.com/rebellabs/object-oriented-design-principles-and-the-5-ways-o