CS251: Software Engineering I

Lecture Practice 7

Cairo University, Faculty of Computers and Information

Readings and Videos: Lecture 16 and 17 – videos, Readings 10, 11 and 12

Q1: Strategy: Can you tell me any design pattern which you have used recently in your project, except Singleton? This is one of the popular question from various Java interviews in recent years. I think, this actually motivated many Java programmers to explore more design patterns and actually look at original 23 patterns introduced by GOF. Strategy pattern is one of the useful pattern you can mention while answering such question

For the given code: (1) What is the output?**(2)** Rewrite it without Strategy Pattern. **(3)** Compare the 2 versions.

```
public class Test {
    public static void main(String args[]) throws InterruptedException {
        // we can provide any strategy to do the sorting
        int[] var = {1, 2, 3, 4, 5 };
        Context ctx = new Context(new BubbleSort());
        ctx.arrange(var);

        // we can change the strategy without changing Context class
        ctx = new Context(new QuickSort());
        ctx.arrange(var);
    }

}

interface Strategy {
    public void sort(int[] numbers);
}

class BubbleSort implements Strategy {
    @Override
    public void sort(int[] numbers) {
        System.out.println("sorting array using bubble sort strategy");
    }
}

class InsertionSort implements Strategy {
    @Override
    public void sort(int[] numbers) {
        System.out.println("sorting array using insertion sort strategy");
    }
}

class QuickSort implements Strategy {
    @Override
    public void sort(int[] numbers) {
        System.out.println("sorting array using quick sort strategy");
    }
}

class MergeSort implements Strategy {
    @Override
    public void sort(int[] numbers) {
        System.out.println("sorting array using merge sort strategy");
    }
}
```

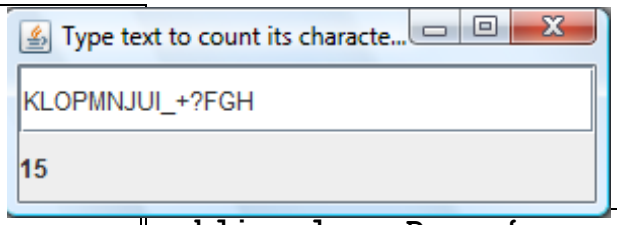
```
class Context {
    private final Strategy strategy;

    public Context(Strategy strategy) {
        this.strategy = strategy;
    }

    public void arrange(int[] input) {
        strategy.sort(input);
    }
}
```

CS251: Software Engineering I

Lecture Practice 7

Cairo University, Faculty of Computers and Information

Q2: MVC: Given the following incomplete implementation of a program that counts the characters you type using the MVC control architecture (or pattern), **fill in** the missing parts for the pattern to work correctly. **Write Java code**. Assume **import** statements are already included. The user interface of the program is shown.




```
public class Model extends Observable // Model
{
   private int nChar =  0;
   public void setCount (String newText) {
     nChar = newText.length();
     ..... // (1) Add code here
   }
   public int getCount () {return nChar;}
}
```

```
public class Demo {
  public static void
    main(String[] s) {
      new View(new Model());
  }
}
```

```
public class View extends JFrame implements Observer        // View
{
   private JTextField text = new JTextField();
   private JLabel counter  = new JLabel();
   private Model model;
   public View(Model givenModel) {

      this.model = givenModel;

      ..... // (2) Add code here
      setTitle("Type text to count its characters.");
      setSize(300, 100);
      setLayout (new GridLayout(2,1));

      text.addActionListener (new Controller());

      add (text);
      add (counter);
      setVisible(true);
   }
  ..... // (3) Add code here

   private class Controller implements ActionListene    // Controller
   {
      public void actionPerformed(ActionEvent e) {
        model.setCount (text.getText());
      }
   }
}
```