SE 3030 – Software Architecture

Lab 2 – Design Patterns Lab Spot Test (2 hours)

Prerequisites

- Java SDK 7 (or above)
- Eclipse IDE
- Knowledge on Design Patterns

Exercise 1 – Bridge Pattern

1. Use one remote controller for two types of **TVs** (**LG tv** and **Sony tv**) Implement how you can proceed with two types of TVs for both.
2. Create two interfaces for **TV** and **RemoteController** and implements operation **on(), off()** and **tune(int channel)**
3. Now implement 2 concrete classes for **LGtv** and **SonyTv** and implement above **on(), off()** and **tune(int channel)** operations in each class
4. Now implement the **RemoteContrlolerImpl** class
5. Now create a Test class as follows and display the outputs below. Your implementation of above concrete classes should satisfy below outputs

```
3  public class Test {
4      public static void main(String[] args) {
5          TV lgLv = new LGTV();
6          TV sontTv = new SonyTV();
7
8          new RemoteControllerImpl(lgLv).on();
9          new RemoteControllerImpl(lgLv).off();
10         new RemoteControllerImpl(lgLv).tune(10);
11         new RemoteControllerImpl(sontTv).on();
12         new RemoteControllerImpl(sontTv).off();
13         new RemoteControllerImpl(sontTv).tune(20);
14     }
15 }
```

Console Problems Javadoc Declaration

```
<terminated> Test (3) [Java Application] C:\Program Files\Java\jre1.8.0_20\
Switch on LG TV
Switch off LG TV
Switch on chanel in LG TV is: 10
Switch on Sony TV
Switch off Sony TV
Switch on chanel in Sony TV is: 20
```

Exercise 2 – Adaptor Pattern

1. Create a **Person** class with NIC, First Name, Last Name, Date of Birth and Phone Number as properties
2. Create a **Student** interface with getter method signatures to **getFullName()**, **getAge()** and **getContactNumber()**
3. Create a **UniversityStudent** Class that implements Student
    a. Add Full Name, Age and Contact Number properties
    b. Implement the getters and setters
4. Create an Adapter that converts Person object to get details of Student
5. Create a Test class to test out the pattern functionality

Exercise 3 – Observer Pattern

1. Create an Interfaces for **Observer** and **Subject.** You should update the subject that is visible to all observers
    a. Implement a **ObserverImpl** class to get the updated state from the **Subject**
2. Create a **SubjectImpl** class and implement the below operations
    a. You should register all observers in the subject class. Use **registerObserver()** method
    b. You should be able to remove the observer using **removeObserver()** method
    c. Then implement **setStatus()** and **getStatus()** method to update the status in Subject class that should be reflected to all observers
    [Hint: Use String value to set or update the status]
    d. When you are going to change or set **new status** that should be **notified to all observers.** Implement the **notifyObservers()** method
3. Implement a test class to test the observer as follows

```
 4 public class TestObserver {
 5
 6     public static void main(String[] args) {
 7
 8         Observer observer1 = new ObserverImpl("Observer 1");
 9         Observer observer2 = new ObserverImpl("Observer 2");
10         Observer observer3 = new ObserverImpl("Observer 3");
11         Observer observer4 = new ObserverImpl("Observer 4");
12         Observer observer5 = new ObserverImpl("Observer 5");
13
14         Subject subject = new SubjectImpl();
15         subject.registerObserver(observer1);
16         subject.registerObserver(observer2);
17         subject.registerObserver(observer3);
18         subject.registerObserver(observer4);
19         subject.registerObserver(observer5);
20
21         subject.setStatus("status 111111111");
22     }
23 }
```

```
Console  Problems  Javadoc  Declaration  Servers  Data Source Explorer  Debug
<terminated> TestObserver [Java Application] C:\Program Files\Java\jre1.8.0_20\bin\javaw.exe (Feb 10, 2019, 3:49:04 PM)
Observer recieved state change of subject ID is = Observer 1 Status = status 111111111
Observer recieved state change of subject ID is = Observer 2 Status = status 111111111
Observer recieved state change of subject ID is = Observer 3 Status = status 111111111
Observer recieved state change of subject ID is = Observer 4 Status = status 111111111
Observer recieved state change of subject ID is = Observer 5 Status = status 111111111
```

Exercise 4 – Decorator Pattern

1. Create an **Abstract class** called **Employee** and implement another **sub classes** called **Consultant, Manager and Engineer** and **extends** the **Employee** class.

2. Employee should consist an **abstract method getSalary()** which override in each subclass to display their salary for the designation
    a. Consultant salary is 70,000/=
    b. Manager salary is 80,000/=
    c. Engineer salary is 60,000/=

3. Create another abstract class called **EmployeeQualification** that has **abstract** method **getDescription()** and extends Employee class as well

4. Then create a class **EmployeeCertification** and **extends EmployeeQualification**
    a. Override getDescription and getSalary() methods
    b. If employee has certification add an additional 30,000/= for the salary

5. Similarly create a class **WorkExperience** and **extends EmployeeQualification**
    a. Override getDescription and getSalary() methods
    b. If employee has enough work experience add an additional 20,000/= for the salary

6. Now create a **TestPattern** class as follows and display the output below. Your implementation of above classes should satisfy below outputs

```
3  public class TestPattern {
4
5      public static void main(String[] args) {
6
7          Employee employee = new Engineer();
8          employee = new EmployeeCertification(employee);
9          employee = new WorkExperience(employee);
10
11         System.out.println(employee.description + " " + employee.getSalary());
12
13         Employee employee2 = new Consultant();
14         employee2 = new WorkExperience(new EmployeeCertification(employee2));
15
16         System.out.println(employee2.description + " " + employee2.getSalary());
17
18         System.out.println("PM salary = "
19                 + new WorkExperience(new EmployeeCertification(new Manager()))
20                         .getSalary());
21     }
22 }
```

```
Console  Problems  Javadoc  Declaration  Servers  Data Source Explorer  Debug
<terminated> TestPattern [Java Application] C:\Program Files\Java\jre1.8.0_20\bin\javaw.exe (Feb 10, 2019, 4:24:43 PM)
General employee details 110000.0
General employee details 120000.0
PM salary = 130000.0
```

==========THE END ==========