SLIIT
Discover Your Future

BSc (Hons) in Information Technology
Year 2

Lab 06 – Design Patterns

IT2020 - Software Engineering Semester 1, 2025

In this Lab we will look into Design Patterns. You will use the NetBeans IDE.

Activity 1

Suppose we are building a cricket app that notifies viewers about the information such as current score, run rate etc. Suppose we have made two display elements CurrentScoreDisplay and AverageScoreDisplay. CricketData has all the data (runs, bowls etc.) and whenever data changes the display elements are notified with new data and they display the latest data accordingly.

a. If you are to develop the above functionality using design patterns, what is the design pattern that you will use?

b. Implement a simple application to demonstrate the above scenario.

Note: We will not implement an actual Client/Server environment. We will simply simulate the concept of using the design patterns.

Follow the below steps.

1. Open NetBeans IDE.
2. Create a new Java Application and name it as "observer1 application".
3. Add a new class to the “observer1 application " package called "Subject.java" which is an interface. Add the method signatures that will be implemented by the concrete subject class as the next step. Method signatures include “ registerObserver(Observer o)” and “unregisterObserver(Observer o)” which will take an Observer object as a parameter. In addition, “notifyObservers()”, “ dataChanged()”, “getLatestRuns()”, “getLatestWickets()”, and “getLatestOvers()” are included in the Subject interface. You can place the code relevant for the Subject interface.
4. Add a new class to the “observer1 application " package called "CricketData" which is the concrete class of the Subject interface. Implement all the methods defined in the

Subject interface. Implement the constructor to initialize an Array List. " registerObserver(Observer o)" method will add observers to the array list while "unregisterObserver(Observer o)" method will remove observers from the array list. "dataChanged()" method will be invoked by the main method, which will then invoke the "notifyObservers()" method. "notifyObservers()" method will notify all the observers about the change that happened. "GetLatestRuns()", "GetLatestWickets()" and "GetLatestOvers()" methods will return latest runs, wickets and overs respectively. You can place the code relevant for the "CricketData" class.

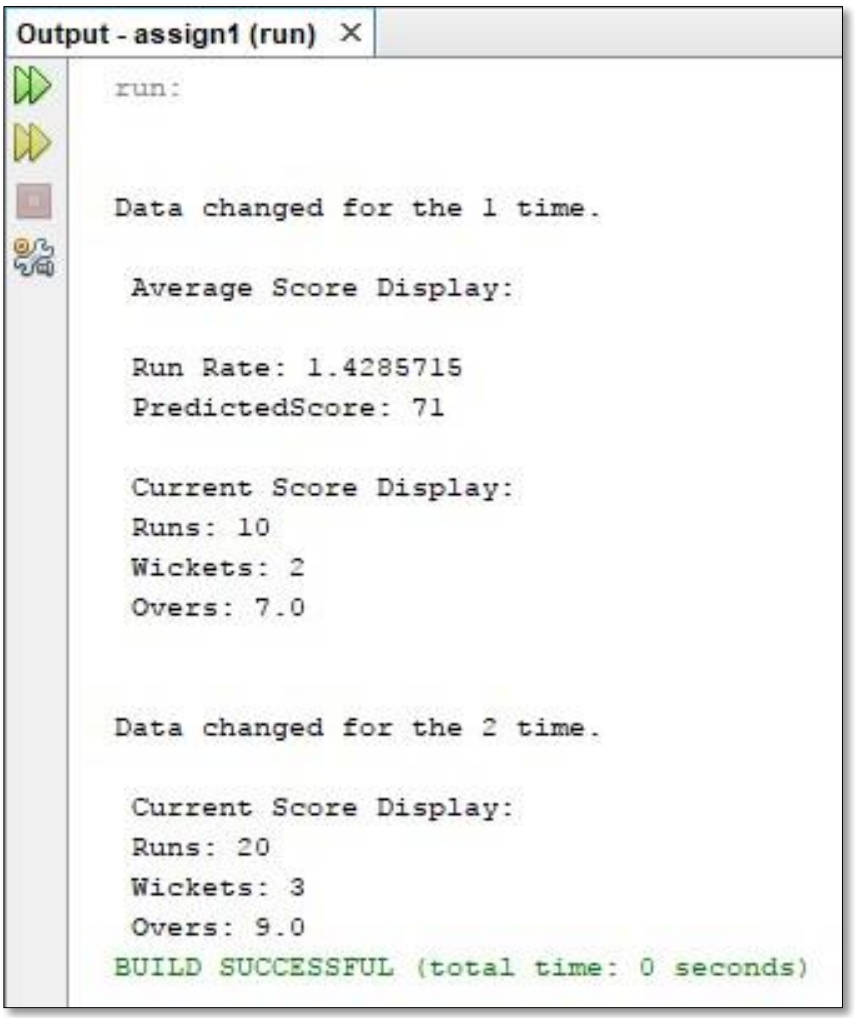
5. Next add a new class to the "observer1 application " package called "Observer.java" which is an interface. Add the method signature for "update (Subject subject)" method that will be implemented by the concrete observer classes as the next step. "update()" method will take a Subject type object as a parameter. You can place the code relevant for the Observer interface.

6. Implement the concrete Observer classes which have the implementation for the "update (Subject subject)" method. In this example, we will take two observer classes named "CurrentScoreDisplay" and "AverageScoreDisplay". "Update()" method will invoke the "getLatestRuns()", "getLatestWickets()", and "getLatestOvers()" methods in CricketData class to retrieve data and will display current scores and average score respectively by "CurrentScoreDisplay" and "AverageScoreDisplay".

7. Complete the main method in the "TestObserver" class. Create observer objects from concrete observer classes and subject type objects from concrete subject class. Register all the

observer objects through “registerObserver(Observer o)” method of concrete subject class. Then simulate the change, by invoking the “dataChanged()” method of “CricketData” class.

8. Execute and run the application. Understand how the change is notified to the observers, “CurrentScoreDisplay” and “AverageScoreDisplay”.



SLIIT
Discover Your Future

BSc (Hons) in Information Technology
Year 2

Lab 06 – Design Patterns

IT2020 - Software Engineering Semester 1, 2025

Activity 2

Suppose we are building an application which has three display elements, “BinaObserver”, “OctaObserver” and “HexaObserver” which will convert a given integer into binary format, octal format and hexadecimal format respectively. Whenever the user input an integer value, receiving of different values to the application (data change) is notified to all three display elements. Then the display elements will display the respective value after the conversion.

a) If you are to develop the above functionality using design patterns, what is the design pattern that you will use?

b) Implement a simple application to demonstrate the above scenario.

Follow the below steps.

1. Open NetBeans IDE.
2. Create a new Java Application and name it as "observer2 application".
3. Add a new class to the “observer2 application " package called "Subject.java" which is an interface. Add the method signatures that will be implemented by the concrete subject class as the next step. Method signatures include “ registerObserver(Observer o)” and “unregisterObserver(Observer o)” which will take an Observer object as a parameter. In addition, “notifyObservers()”, “ setState(int value)”, “getState()” are included in the Subject interface. You can place the code relevant for the Subject interface.

4. Add a new class to the “observer2 application " package called "SubjectImpl" which is the concrete class of the Subject interface. Implement all the methods defined in the Subject interface. Implement the constructor to initialize an Array List. “registerObserver(Observer o)” method will add observers (display elements) to the

array list while “unregisterObserver(Observer o)” method will remove observers (display elements) from the array list. “setState()” method will be invoked by the main method, which will then invoke the “notifyObservers()” method. “notifyObservers()” method will notify all the observers/display elements about the change that happened. “getState()” method will return the value user input. You can place the code relevant for the "SubjectImpl" class.

5. Next add a new class to the “observer2 application " package called "Observer.java" which is an interface. Add the method signature for “update (Subject subject)” method that will be implemented by the concrete observer classes as the next step. “update(Subject subject)” method will take a Subject type object as a parameter. You can place the code relevant for the Observer interface.

6. Implement the concrete Observer classes which have the implementation for the “update (Subject subject)” method. In this example, we will take three observer classes for the three display elements named “BinaObserver”, “OctaObserver” and “HexaObserver”. “Update()” method will invoke the “getState() method in “SubjectImpl” class to retrieve data and will display binary format, octal format and hexadecimal format respectively by “BinaObserver”, “OctaObserver” and “HexaObserver”

7. Complete the main method in the “TestObserver” class. Create observer objects from concrete observer classes and subject type objects from concrete subject class. Register all the observer objects through “registerObserver(Observer o)” method of concrete subject class. Then simulate the change, by invoking the “setState(int value)” method of “SubjectImpl” class.

8. Execute and run the application. Understand how the change is notified to the observers, "BinaObserver", "OctaObserver" and "HexaObserver".