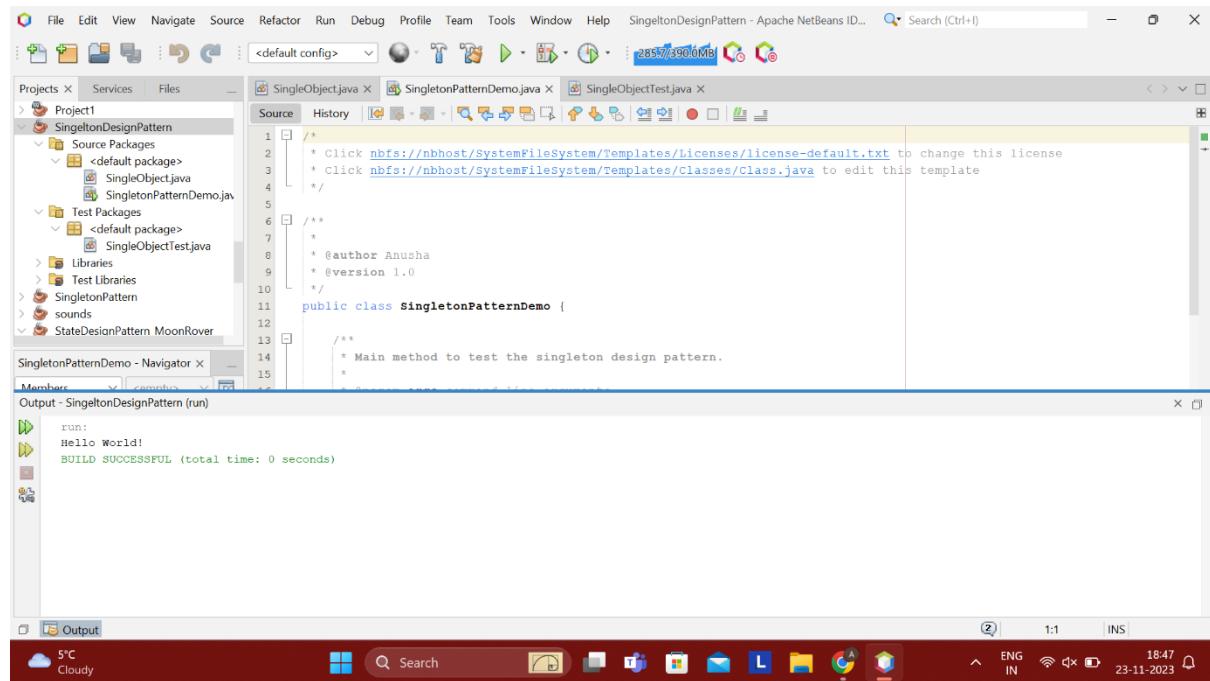


ASSIGNMENT – 4

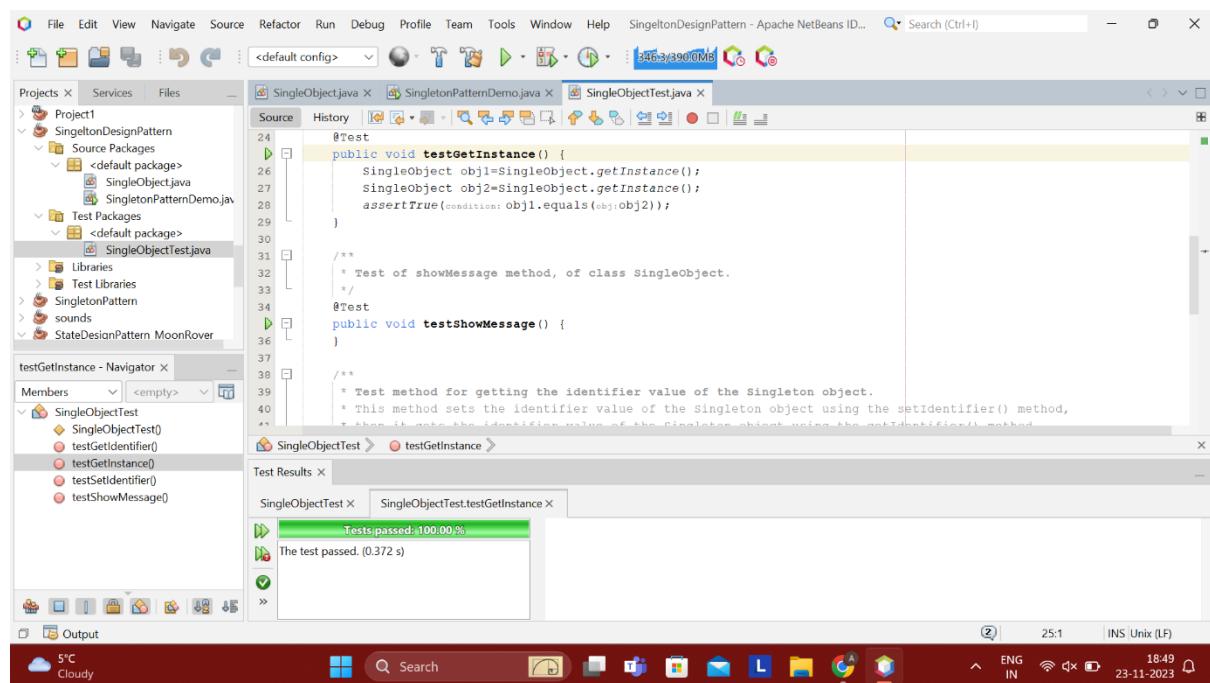
NU ID: 002644927

1) SINGLETON PATTERN

SAMPLE RUN:



TEST CASES RUN:

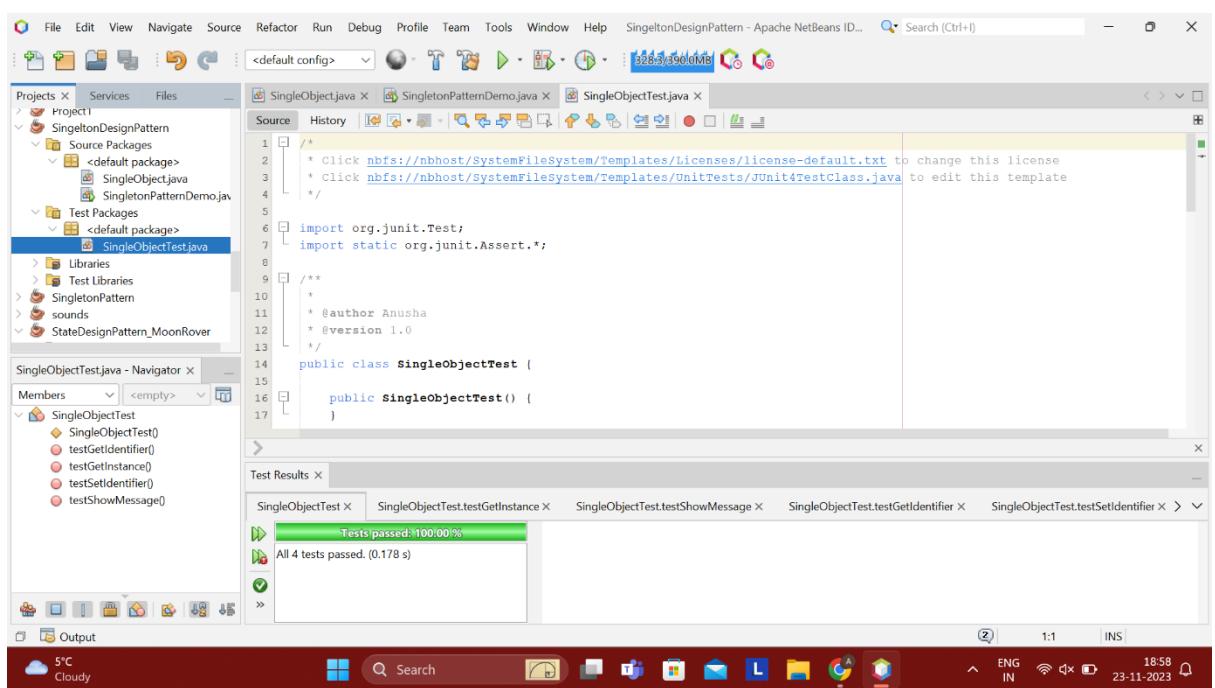
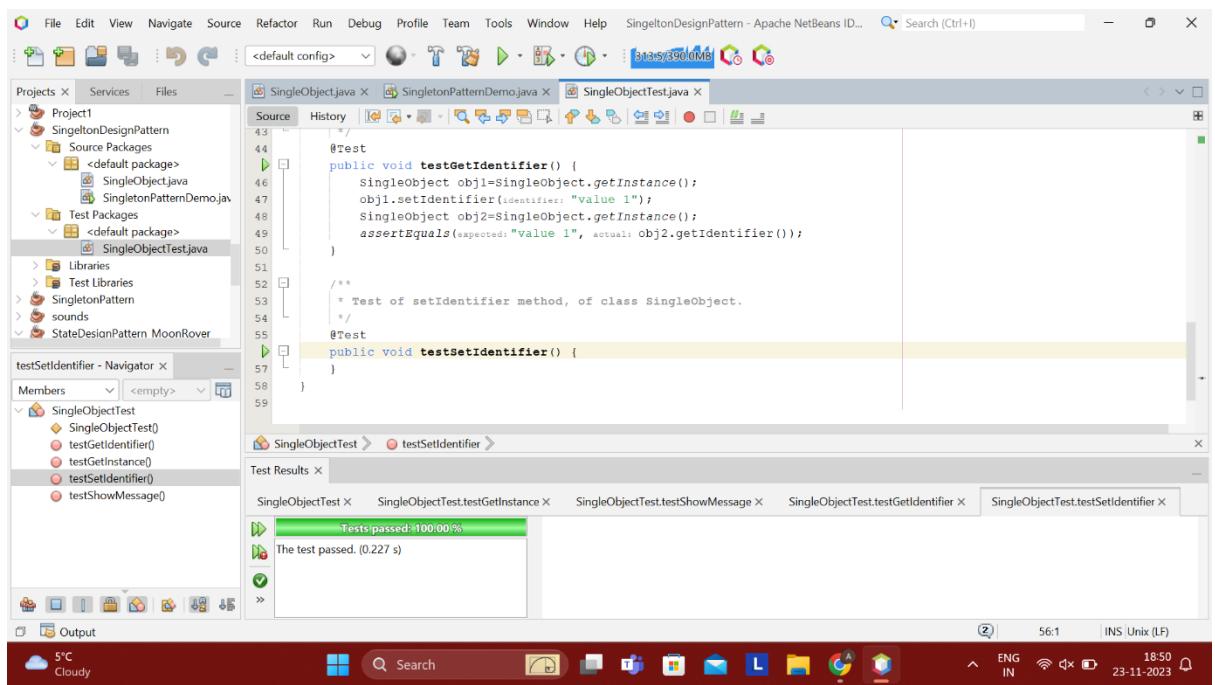


The screenshot shows the Apache NetBeans IDE interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and SingeltonDesignPattern - Apache NetBeans ID... A search bar at the top right contains the text "Search (Ctrl+F)". The main window has tabs for SingleObject.java, SingletonPatternDemo.java, and SingleObjectTest.java. The SingleObjectTest.java tab is active, displaying Java test code. The code includes two test methods: testGetInstance() and testShowMessage(). The testShowMessage() method is highlighted with a yellow background. The code uses JUnit annotations like @Test and assertions like assertEquals. Below the code editor is a Test Results panel showing "Tests passed: 100.00%" and "The test passed. (0.272 s)". The bottom status bar shows the date and time as 23-11-2023 18:50, and the system status as INS (Unix LF). The taskbar at the bottom includes icons for various applications like File Explorer, Task Manager, and a Start button.

```
24     /**
25      * Test of getInstance method, of class SingleObject.
26      */
27     @Test
28     public void testGetInstance() {
29         SingleObject obj1=SingleObject.getInstance();
30         SingleObject obj2=SingleObject.getInstance();
31         assertEquals(obj1,obj2);
32     }
33
34     /**
35      * Test of showMessage method, of class SingleObject.
36      */
37     @Test
38     public void testShowMessage() {
39
40     }
41
42     /**
43      * Test method for getting the identifier value of the Singleton object.
44      * This method sets the identifier value of the Singleton object using the setIdIdentifier() method,
45      * then it gets the identifier value of the Singleton object using the getIdentifier() method.
46     */
47     @Test
48     public void testGetIdentifier() {
49         SingleObject obj1=SingleObject.getInstance();
50         obj1.setIdIdentifier("value 1");
51         SingleObject obj2=SingleObject.getInstance();
52         assertEquals("value 1", obj2.getIdIdentifier());
53     }
54
55     /**
56      * Test of setIdIdentifier method, of class SingleObject.
57      */
58     @Test
59     public void testSetIdentifier() {
59 }
```

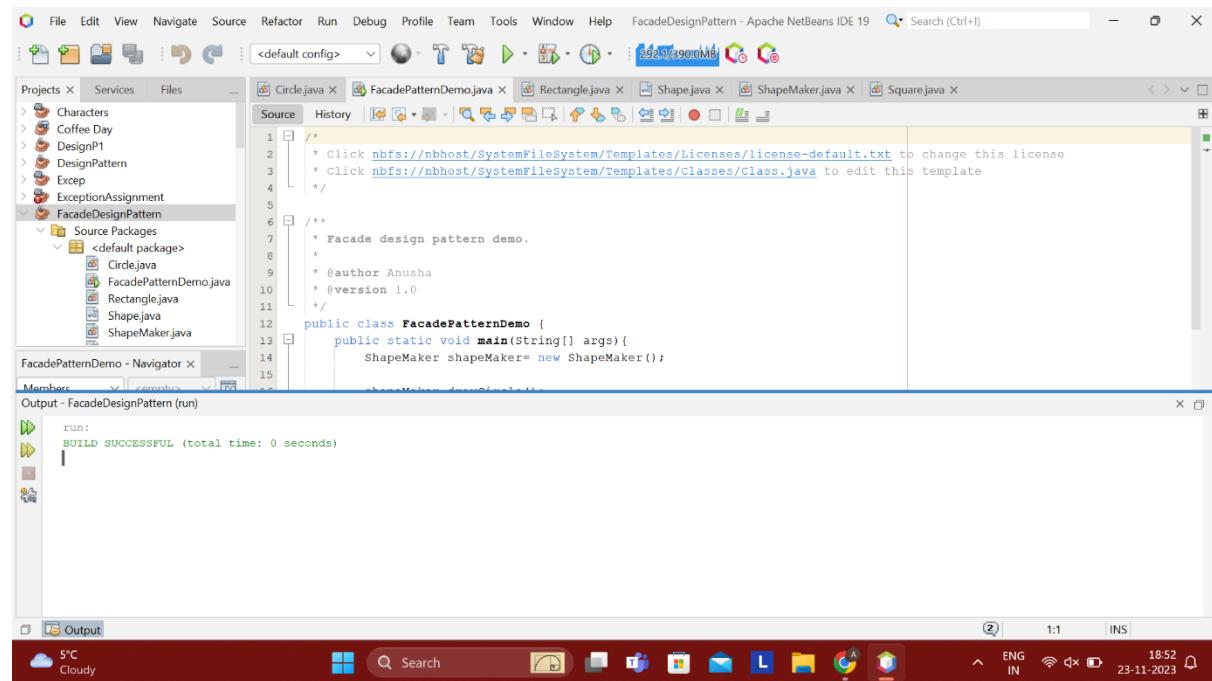
This screenshot is nearly identical to the one above, showing the same Apache NetBeans IDE interface. The main difference is that the code editor now highlights the testGetIdentifier() method in the SingleObjectTest.java file. The Test Results panel shows "Tests passed: 100.00%" and "The test passed. (0.353 s)". The rest of the interface, including the menu bar, toolbars, and taskbar, remains the same.

```
43     /**
44      * Test of getInstance method, of class SingleObject.
45      */
46     @Test
47     public void testGetInstance() {
48         SingleObject obj1=SingleObject.getInstance();
49         obj1.setIdIdentifier("value 1");
50         SingleObject obj2=SingleObject.getInstance();
51         assertEquals("value 1", obj2.getIdIdentifier());
52     }
53
54     /**
55      * Test method for getting the identifier value of the Singleton object.
56      * This method sets the identifier value of the Singleton object using the setIdIdentifier() method,
57      * then it gets the identifier value of the Singleton object using the getIdentifier() method.
58     */
59     @Test
59 }
```



2) FAÇADE PATTERN

SAMPLE RUN:

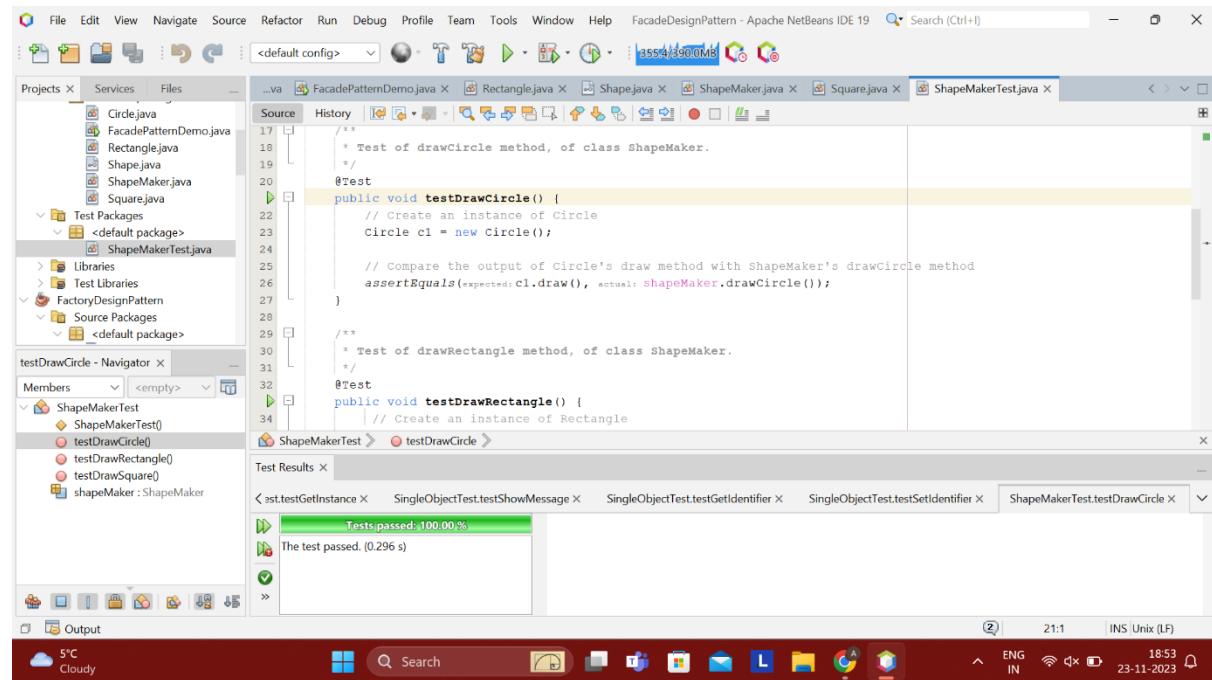


```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
/*
 * Facade design pattern demo.
 *
 * @author Anusha
 * @version 1.0
 */
public class FacadePatternDemo {
    public static void main(String[] args){
        ShapeMaker shapeMaker= new ShapeMaker();
        shapeMaker.drawCircle();
        shapeMaker.drawRect();
        shapeMaker.drawSquare();
    }
}
```

Output - FacadeDesignPattern (run)

```
run:
BUILD SUCCESSFUL (total time: 0 seconds)
```

TEST CASES RUN:



```
/*
 * Test of drawCircle method, of class ShapeMaker.
 */
@Test
public void testDrawCircle() {
    // Create an instance of Circle
    Circle c1 = new Circle();

    // Compare the output of Circle's draw method with ShapeMaker's drawCircle method
    assertEquals(expected:c1.draw(), actual: shapeMaker.drawCircle());
}

/*
 * Test of drawRectangle method, of class ShapeMaker.
 */
@Test
public void testDrawRectangle() {
    // Create an instance of Rectangle
}
```

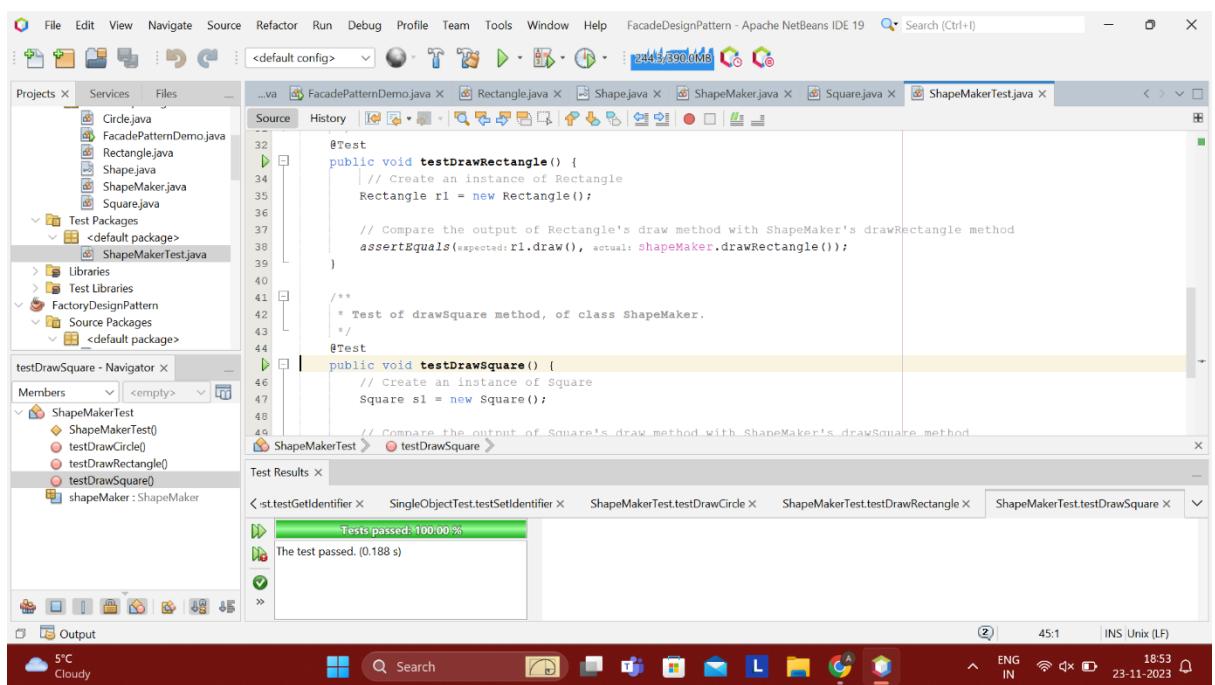
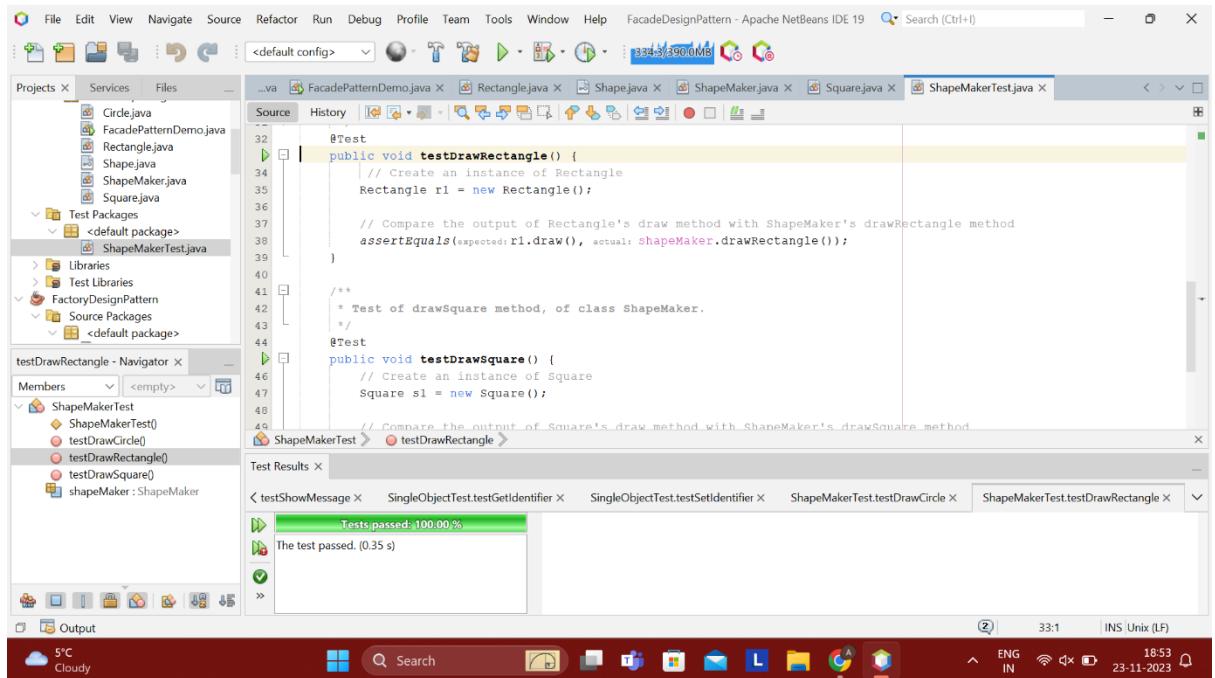
testDrawCircle - Navigator

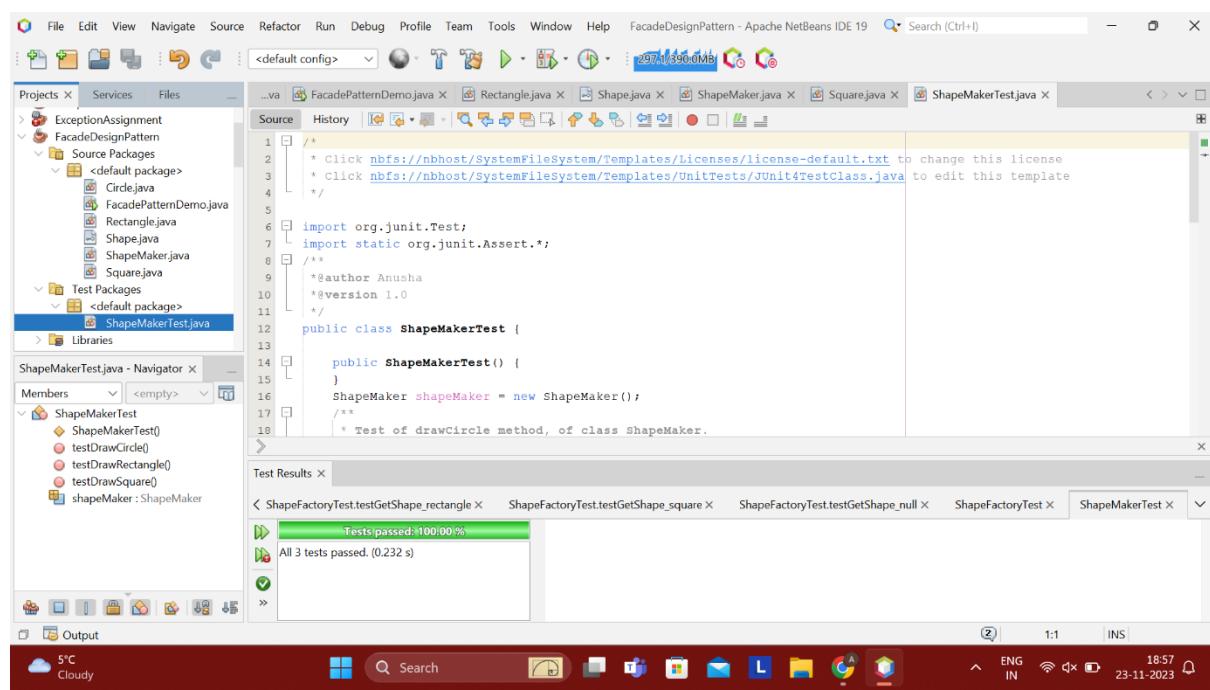
Members

- ShapeMakerTest
- ShapeMakerTest()
- testDrawCircle()
- testDrawRectangle()
- testDrawSquare()
- shapeMaker : ShapeMaker

Test Results

Test	Status	Details
testDrawCircle	Passed	Tests passed: 100.00%
testDrawRectangle	Passed	The test passed. (0.296 s)





3) FACTORY PATTERN

SAMPLE RUN:

The screenshot shows the Apache NetBeans IDE interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and Search (Ctrl+I). The title bar says "FactoryDesignPattern - Apache NetBeans IDE 19". The Projects tab shows a project named "FactoryDesignPattern" with several source files: Circle.java, FactoryPatternDemo.java, Rectangle.java, Shape.java, Shapefactory.java, and Square.java. The Source tab displays the code for FactoryPatternDemo.java, which demonstrates the Factory design pattern. The Output tab shows the build results, indicating a successful build with no errors or warnings.

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
/**
 * Factory design pattern demo. It uses the ShapeFactory class to get instances of shapes without knowing their classes.
 * @author Anusha
 * @version 1.0
 */
public class FactoryPatternDemo {
    public static void main(String[] args){
        ShapeFactory shapeFactory=new ShapeFactory();
        Shape shape1=shapeFactory.getShape(shapeType: "CIRCLE");
        shape1.draw();
    }
}
```

Output - FactoryDesignPattern (run)

```
run:
Circle:draw()
Rectangle:draw()
Square:draw()
BUILD SUCCESSFUL (total time: 0 seconds)
```

TEST CASES RUN:

The screenshot shows the Apache NetBeans IDE interface with the same project structure as the previous screenshot. The Source tab now displays the code for ShapeFactoryTest.java, which contains test methods for the ShapeFactory class. The Test Results tab shows the outcome of the tests: "Tests passed: 100.00%" and "The test passed. (0.199 s)". The status bar at the bottom indicates the date and time as 23-11-2023 18:56.

```
public void testGetShape() {
    // Validate if null is returned when the input is null
    assertTrue(sh.getShape(shapeType: null) == null);
}

/** 
 * Test of getShape method, of class ShapeFactory.
 */
@Test
public void testGetShape_circle() {
    // Validate if the instance of Circle is returned when input is "circle"
    assertTrue(sh.getShape(shapeType: "circle") instanceof Circle);
}

/** 
 * Test of getShape method, of class ShapeFactory.
 */

```

Test Results

Tests passed: 100.00%

The test passed. (0.199 s)

The screenshot shows the Apache NetBeans IDE interface with the following details:

- Toolbar:** File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help.
- Project Explorer:** FactoryDesignPattern (Source Packages: Circle.java, FactoryPatternDemo.java, Rectangle.java, Shape.java, ShapeFactory.java, Square.java); Test Packages: ShapeFactoryTest.java.
- Code Editor:** ShapeFactoryTest.java (containing testGetShape() and testGetShape_circle() methods).
- Navigator:** testGetShape_circle - Navigator (Members: <empty>).
- Test Results:** Test Results tab showing "Tests passed: 100.00%" and "The test passed. (0.184 s)".
- System Tray:** Shows weather (5°C Cloudy), system status (ENG IN), network (Wi-Fi), battery (23-11-2023), and system time (18:56).

FactoryDesignPattern - Apache NetBeans IDE 19

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help Search (Ctrl+I)

Projects X Services Files

Source History <default config> 296.9/390.0MB ShapeFactoryTest.java

FactoryDesignPattern

Source Packages

<default package>

FactoryPatternDemo.java Circle.java Rectangle.java Shape.java ShapeFactory.java Square.java

Test Packages

<default package>

ShapeFactoryTest.java

Libraries Test Libraries

testGetShape_square - Navigator X

Members <empty>

ShapeFactoryTest

testGetShape()

testGetShape_square()

testGetShape_circle()

testGetShape_rectangle()

testGetShape_null()

sh : ShapeFactory

Source History <default config> 296.9/390.0MB ShapeFactoryTest.java

46 assertTrue(sh.getShape(shapetype: "rectangle") instanceof Rectangle);

47 }

48 */**

49 * Test of getShape method, of class ShapeFactory.

50 */

51 @Test

52 public void testGetShape_square() {

53 // Validate if the instance of Square is returned when input is "square"

54 assertTrue(sh.getShape(shapetype: "square") instanceof Square);

55 }

56 */**

57 * Test of getShape method, of class ShapeFactory.

58 */

59 @Test

60 public void testGetShape_null() {

61 // Validate if null is returned for an unknown shape

62 }

63

ShapeFactoryTest > testGetShape_square >

Test Results X

ShapeFactoryTest.testGetShape X ShapeFactoryTest.testGetShape_circle X ShapeFactoryTest.testGetShape_rectangle X ShapeFactoryTest.testGetShape_square X

Tests passed: 100.00% The test passed. (0.198 s)

Output

5°C Cloudy Search 53:1 ENG IN 18:56 23-11-2023

FactoryDesignPattern - Apache NetBeans IDE 19

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help Search (Ctrl+I)

Projects X Services Files

Source History <default config> 222.2/390.0MB ShapeFactoryTest.java

FactoryDesignPattern

Source Packages

<default package>

FactoryPatternDemo.java Circle.java Rectangle.java Shape.java ShapeFactory.java Square.java

Test Packages

<default package>

ShapeFactoryTest.java

Libraries Test Libraries

testGetShape_null - Navigator X

Members <empty>

ShapeFactoryTest

testGetShape()

testGetShape_square()

testGetShape_circle()

testGetShape_rectangle()

testGetShape_null()

sh : ShapeFactory

Source History <default config> 222.2/390.0MB ShapeFactoryTest.java

55 assertTrue(sh.getShape(shapetype: "square") instanceof Square);

56 }

57 */**

58 * Test of getShape method, of class ShapeFactory.

59 */

60 @Test

61 public void testGetShape_null() {

62 // Validate if null is returned for an unknown shape

63 assertEquals(sh.getShape(shapetype: "unknown"), null);

64 }

65 }

66

67

68 }

69

ShapeFactoryTest > testGetShape_null >

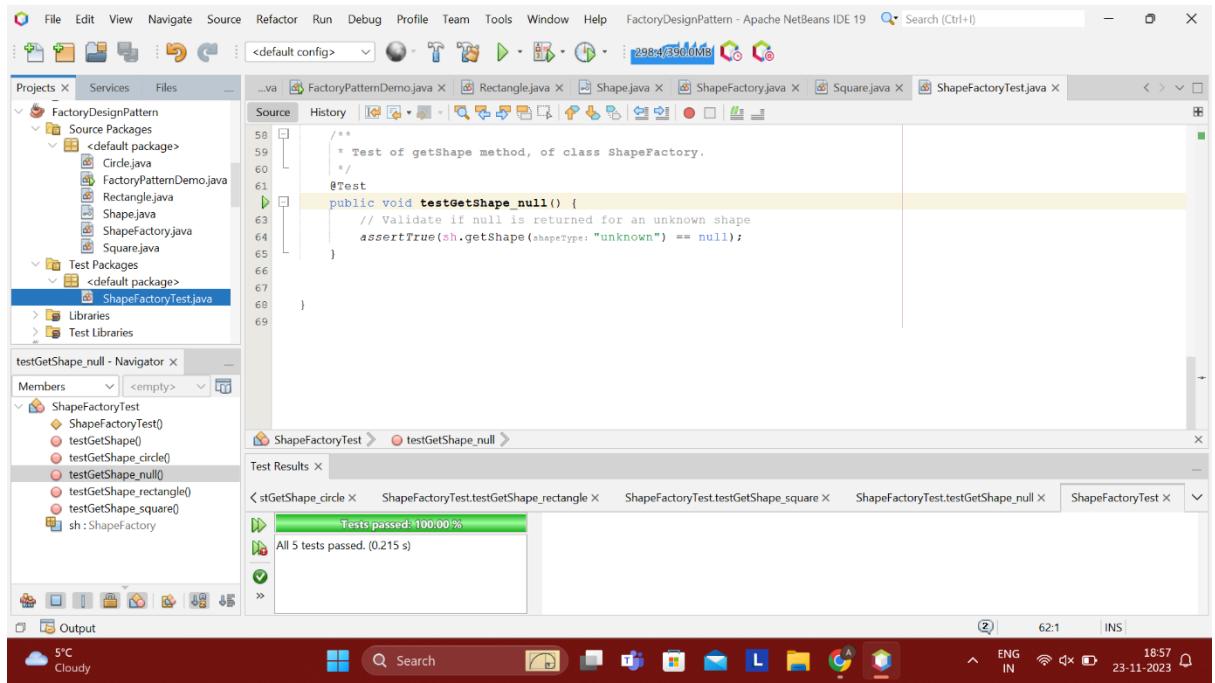
Test Results X

ShapeFactoryTest.testGetShape_circle X ShapeFactoryTest.testGetShape_rectangle X ShapeFactoryTest.testGetShape_square X ShapeFactoryTest.testGetShape_null X

Tests passed: 100.00% The test passed. (0.185 s)

Output

5°C Cloudy Search 62:1 ENG IN 18:56 23-11-2023



4) STATE DESIGN PATTERN: MOON ROVER

SAMPLE RUN:

The screenshot shows the Apache NetBeans IDE interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and Search (Ctrl+I). The title bar displays "StateDesignPattern_MoonRover - Apache NetBeans". The left sidebar shows the project structure under "Projects X": "Source Packages" containing "AtRest.java", "MoveBackward.java", "MoveForward.java", "Rover.java", and "State.java"; and "Test Packages" containing "RoverTest.java". The main editor window shows the "Rover.java" code. The output window at the bottom shows the run results:

```
Output: StateDesignPattern_MoonRover (run)
Error: Can only press Left Pedal thrice when in Decelerate State.
Unable to move.
Current State: Move Backward
Current SubState: Accelerate
Transitioning from Accelerate State to Decelerate State...
Current State: Move Backward
Current SubState: Decelerate
Transitioning from Decelerate State to At Rest State...
Current State: At Rest
Current SubState: None
BUILD SUCCESSFUL (total time: 0 seconds)
```

The status bar at the bottom right shows the date and time as 23-11-2023, 18:58.

TEST CASES RUN:

The screenshot shows the Apache NetBeans IDE interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help, and Search (Ctrl+I). The title bar displays "StateDesignPattern_MoonRover - Apache NetBeans". The left sidebar shows the project structure under "Projects X": "Source Packages" containing "AtRest.java", "MoveBackward.java", "MoveForward.java", "Rover.java", and "State.java"; and "Test Packages" containing "RoverTest.java". The main editor window shows the "RoverTest.java" code. The test results window at the bottom shows the outcome of the "testPressRightPedal" test:

```
Test Results:
Tests passed: 100.00%
The test passed. (0.212 s)
```

The status bar at the bottom right shows the date and time as 23-11-2023, 18:59.

Screenshot of the Eclipse IDE interface showing the code editor and test results for the RoverTest.java file.

Code Editor:

```

    @Test
    public void testPressRightPedalForTime() {
        //1. Can only press Right Pedal for Time when inside Move Forward State, it wont work when rover is At Rest at
        r.pressRightPedalForTime(numOfSecondsPressed:3);
        assertEquals(expected:"At Rest",actual: r.currentState.name);

        //2. Transitioning from At Rest State to Move Forward State to check for right peda functionality
        r.pressRightPedal (numOftimesPressed: 1);
        assertEquals(expected:"Move Forward",actual: r.currentState.name);

        //3. When pressed right pedal for 5 sec, Transitioning from Accelerate subState to Constant Speed subState
        r.pressrightPedalForTime(numOfsecondsPressed:5);
        assertEquals(expected:"Constant Speed",actual: r.currentState.subState);

        //4. When pressed right pedal for 3 sec, Transitioning from Constant Speed subState to Decelerate subState
        r.pressRightPedalForTime(numOfSecondsPressed:3);
    }

```

Test Results:

Test Case	Status	Message
testPressRightPedalForTime	Tests passed: 100.00%	Error: Can only press Right Pedal for Time when inside Move Forward State. Unable to move. Transitioning from At Rest State to Move Forward State... Transitioning from Accelerate State to Constant Speed State... Transitioning from Constant Speed State to Decelerate State...

Screenshot of the Eclipse IDE interface showing the code editor and test results for the RoverTest.java file.

Code Editor:

```

    @Test
    public void testPressLeftPedal() {
        //1. Left pedal can only be used when in Move Backward state
        r.pressLeftPedal (numOftimesPressed: 1);
        assertEquals(expected:"At Rest",actual: r.currentState.name);

        //2. Transition to Move Backward state to access left pedal by pressing it for 5 sec
        r.pressLeftPedalForTime(numOfSecondsPressed:5);
        assertEquals(expected:"MOVE Backward", actual: r.currentState.name);

        //3. Left pedal when pressed in Move Backward state, rover transitions from Accelerate substate to Decelerate
        r.pressLeftPedal (numOftimesPressed: 2);
        assertEquals(expected:"Decelerate",actual: r.currentState.subState);

        //4. Left pedal when pressed in Decelerate subState, rover transitions from Decelerate subState to Constant Sp
        r.pressLeftPedal (numOftimesPressed: 1);
        assertEquals(expected:"Constant Speed",actual: r.currentState.subState);
    }

```

Test Results:

Test Case	Status	Message
testPressLeftPedal	Tests passed: 100.00%	Error: Can only press Left Pedal when inside Move Backward State. Unable to move. Transitioning from At Rest State to Move Backward State... Transitioning from Accelerate State to Decelerate State... Transitioning from Decelerate State to Constant Speed State...

Screenshot of the Eclipse IDE interface showing the State Design Pattern implementation for a Moon Rover.

Projects View:

- StateDesignPattern_MoonRover
- Source Packages:
 - <default package>
 - AtRest.java
 - MoveBackward.java
 - MoveForward.java
 - Rover.java
 - State.java
 - Test Packages:
 - <default package>
 - RoverTest.java
- Libraries
- Test Libraries
- StateMachine

Members View:

- RoverTest
 - RoverTest()
 - testMain()
 - testPressLeftPedal()
 - testPressLeftPedalForTime()
 - testPressRightPedal()
 - testPressRightPedalForTime()
 - testPrintStateAndSubState()

Source Editor:

```

91     /**
92      * Test of printStateAndSubState method, of class Rover.
93
94      */
95      @Test
96      public void testPressLeftPedalForTime() {
97
98          //1. Rover should always start in "At Rest" state
99          assertEquals(expected:"At Rest",actual:r.currentState.name);
100
101         //2. When at rest, pressing left pedal for 5 sec
102         r.pressLeftPedalForTime(numOfSecondsPressed:5);
103         assertEquals(expected:"Move Backward",actual:r.currentState.name);
104
105         //3. When at Move Backward state, pressing left pedal for 3 sec to move rover in constant backward speed subs
106         r.pressLeftPedalForTime(numOfSecondsPressed:3);
107         assertEquals(expected:"Constant Speed",actual:r.currentState.subState);
108
109     }

```

Test Results View:

- Tests passed: 100.00%
- The test passed. (0.343 s)
- Transitioning from At Rest State to Move Backward State...
- Transitioning from Accelerate State to Constant Speed State...
- Transitioning from Constant Speed State to Accelerate State...

System Bar:

- 5°C Cloudy
- Search
- File Explorer
- Task View
- Properties
- Run View
- Console
- Output
- Help
- 92:1
- INS Unix (LF)
- ENG IN
- 18:59
- 23-11-2023

Screenshot of the Eclipse IDE interface showing the State Design Pattern implementation for a Moon Rover.

Projects View:

- StateDesignPattern_MoonRover
- Source Packages:
 - <default package>
 - AtRest.java
 - MoveBackward.java
 - MoveForward.java
 - Rover.java
 - State.java
 - Test Packages:
 - <default package>
 - RoverTest.java
- Libraries
- Test Libraries
- StateMachine

Members View:

- RoverTest
 - RoverTest()
 - testMain()
 - testPressLeftPedal()
 - testPressLeftPedalForTime()
 - testPressRightPedal()
 - testPressRightPedalForTime()
 - testPrintStateAndSubState()

Source Editor:

```

111      /**
112       * Test of printStateAndSubState method, of class Rover.
113       */
114      @Test
115      public void testPrintStateAndSubState() {
116
117          /**
118           * Test of main method, of class Rover.
119           */
120          @Test
121          public void testMain() {
122
123      }

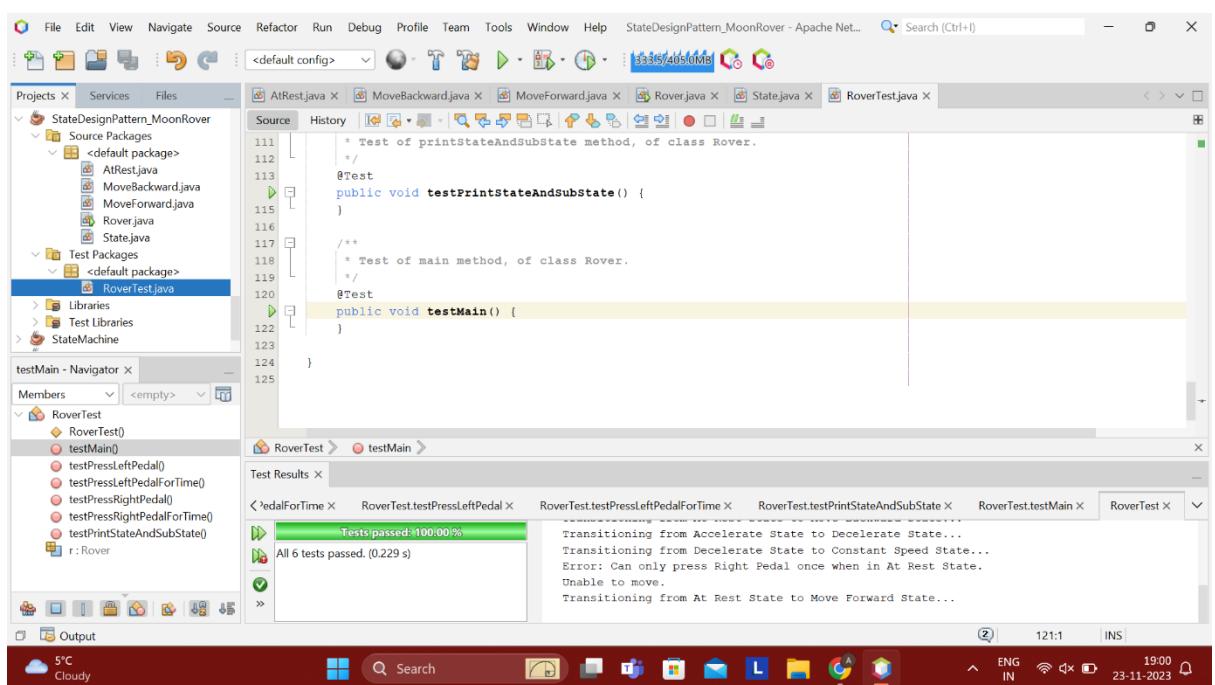
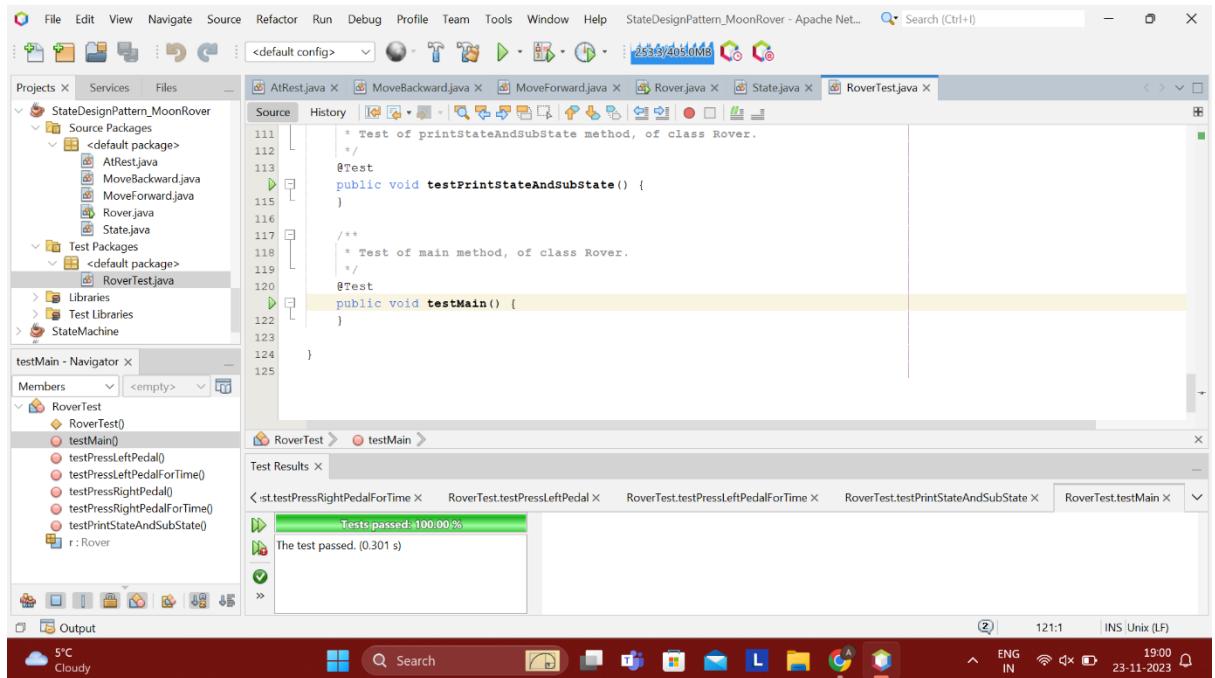
```

Test Results View:

- Tests passed: 100.00%
- The test passed. (0.227 s)

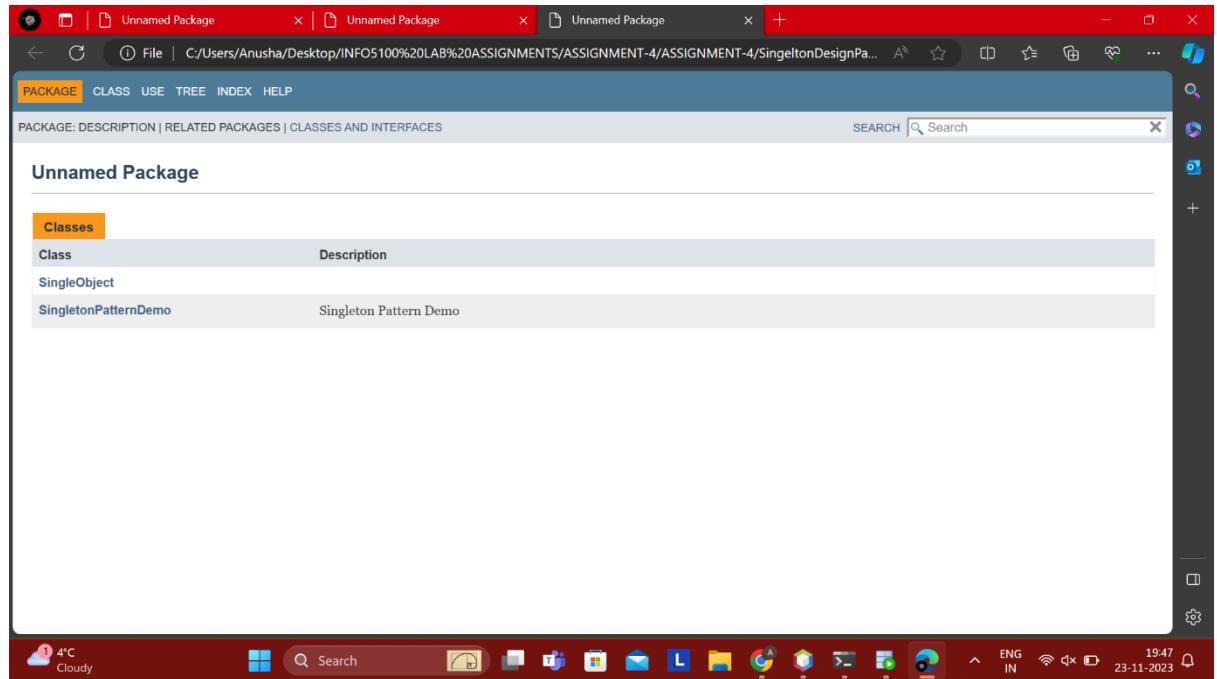
System Bar:

- 5°C Cloudy
- Search
- File Explorer
- Task View
- Properties
- Run View
- Console
- Output
- Help
- 114:1
- INS Unix (LF)
- ENG IN
- 18:59
- 23-11-2023

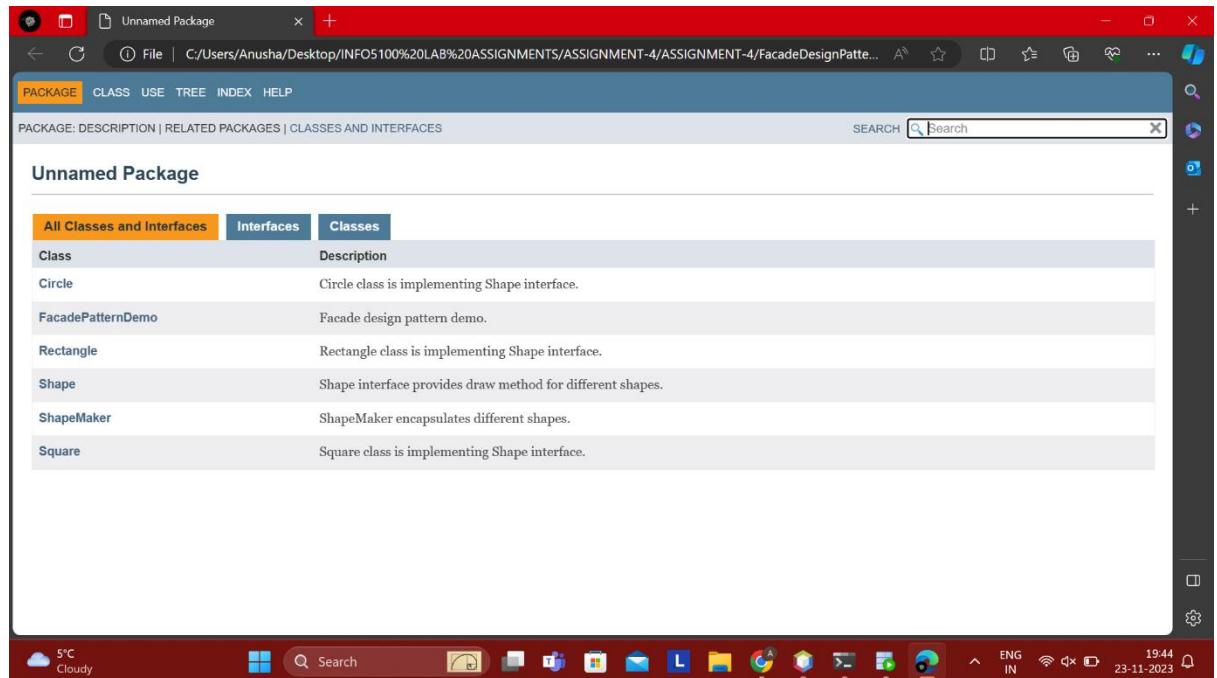


JAVADOC SCREENSHOTS:

1) SINGLETON PATTERN



2) FAÇADE PATTERN



3) FACTORY PATTERN

The screenshot shows a UML modeling interface with three tabs at the top: PACKAGE, CLASS, USE, TREE, INDEX, and HELP. The PACKAGE tab is selected. Below the tabs, there is a search bar labeled "SEARCH" with a magnifying glass icon and a "Search" button. The main area is titled "Unnamed Package". It contains three tabs: "All Classes and Interfaces", "Interfaces", and "Classes". The "Classes" tab is selected. A table lists the following classes and their descriptions:

Class	Description
Circle	Circle class is implementing Shape interface.
FactoryPatternDemo	Factory design pattern demo.
Rectangle	Rectangle class is implementing Shape interface.
Shape	Shape interface provides draw method for different shapes.
ShapeFactory	ShapeFactory provides methods for getting instances of shapes without knowing their classes.
Square	Square class is implementing Shape interface.

The bottom of the screen shows a Windows taskbar with various icons and system status indicators.

4) STATE DESIGN PATTERN: MOON ROVER

The screenshot shows a UML modeling interface with three tabs at the top: PACKAGE, CLASS, USE, TREE, INDEX, and HELP. The PACKAGE tab is selected. Below the tabs, there is a search bar labeled "SEARCH" with a magnifying glass icon and a "Search" button. The main area is titled "Unnamed Package". It contains three tabs: "All Classes and Interfaces", "Interfaces", and "Classes". The "Classes" tab is selected. A table lists the following classes and their descriptions:

Class	Description
AtRest	The rover is in At Rest state.
MoveBackward	The rover is in Move Backward state.
MoveForward	The rover is in Move Forward state.
Rover	This class encapsulates basic functionalities of a rover.
State	Rover class acts as an abstraction for different modes or conditions the Rover can be in.

The bottom of the screen shows a Windows taskbar with various icons and system status indicators.