**Week-4 Lab**

**Design Pattern Assignment**

**RocketPoweredDuck class**

**package** lab4;

/\*\*

\* **@author** Prashanthi sudha kosgi

\* date :06/23/2017.

\*

\*/

**public** **class** RocketPoweredDuck **extends** Duck {

**public** RocketPoweredDuck() {

setFlyBehavior(**new** FlyRocketPowered());

setQuackBehavior(**new** WhooshQuack());

}

**public** **void** display() {

System.***out***.println("I'm a Rocket Powered duck");

}

}

**MiniDuckSimulator class**

**package** lab4;

/\*\*

\* **@author** Prashanthi sudha kosgi

\* date :06/23/2017.

\*

\*/

**public** **class** MiniDuckSimulator {

**public** **static** **void** main(String[] args) {

MallardDuck mallard = **new** MallardDuck();

RubberDuck rubberDuckie = **new** RubberDuck();

DecoyDuck decoy = **new** DecoyDuck();

ModelDuck model = **new** ModelDuck();

mallard.performQuack();

rubberDuckie.performQuack();

decoy.performQuack();

model.performFly();

model.setFlyBehavior(**new** FlyRocketPowered());

model.performFly();

RocketPoweredDuck rocketPoweredDuck = **new** RocketPoweredDuck();

rocketPoweredDuck.display();

rocketPoweredDuck.performFly();

rocketPoweredDuck.performQuack();

}

}

**WhooshQuack class**

**package** lab4;

/\*\*

\* **@author** Prashanthi sudha kosgi

\* date :06/23/2017.

\*

\*/

**public** **class** WhooshQuack **implements** QuackBehavior {

**public** **void** quack() {

System.***out***.println("Whhooshh");

}

}

**FlyRocketPowered class**

**package** lab4;

/\*\*

\* **@author** Prashanthi sudha kosgi

\* date :06/23/2017.

\*

\*/

**public** **class** FlyRocketPowered **implements** FlyBehavior {

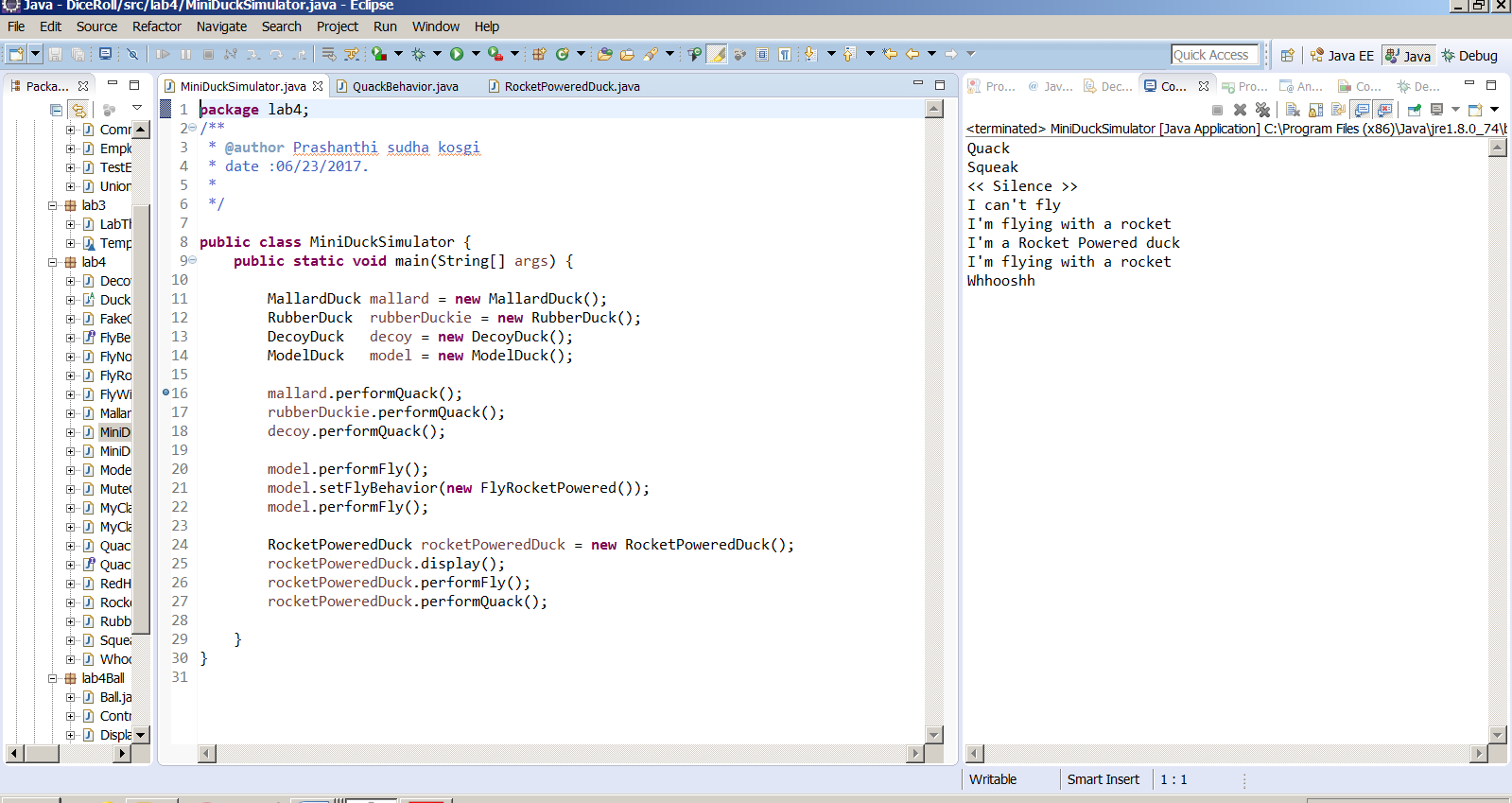
**public** **void** fly() {

System.***out***.println("I'm flying with a rocket");

}

}

**Output:**



**Junit Assignment**

**MyClass :**

**package** lab4;

/\*\*

\* **@author** Prashanthi sudha kosgi

\* date :06/23/2017.

\*

\*/

**public** **class** MyClass {

**public** **int** multiply(**int** x, **int** y) {

**return** x \* y;

}

}

**MyClassTest:**

package lab4;

/\*\*

\* @author Prashanthi sudha kosgi

\* date :06/23/2017.

\*/

import static org.junit.Assert.\*;

import org.junit.Test;

public class MyClassTest {

/\*\*

\* Test method for {@link junit.test.MyClass#multiply(int, int)}

\*/

@Test

public void testMultiply() {

MyClass tester = new MyClass();

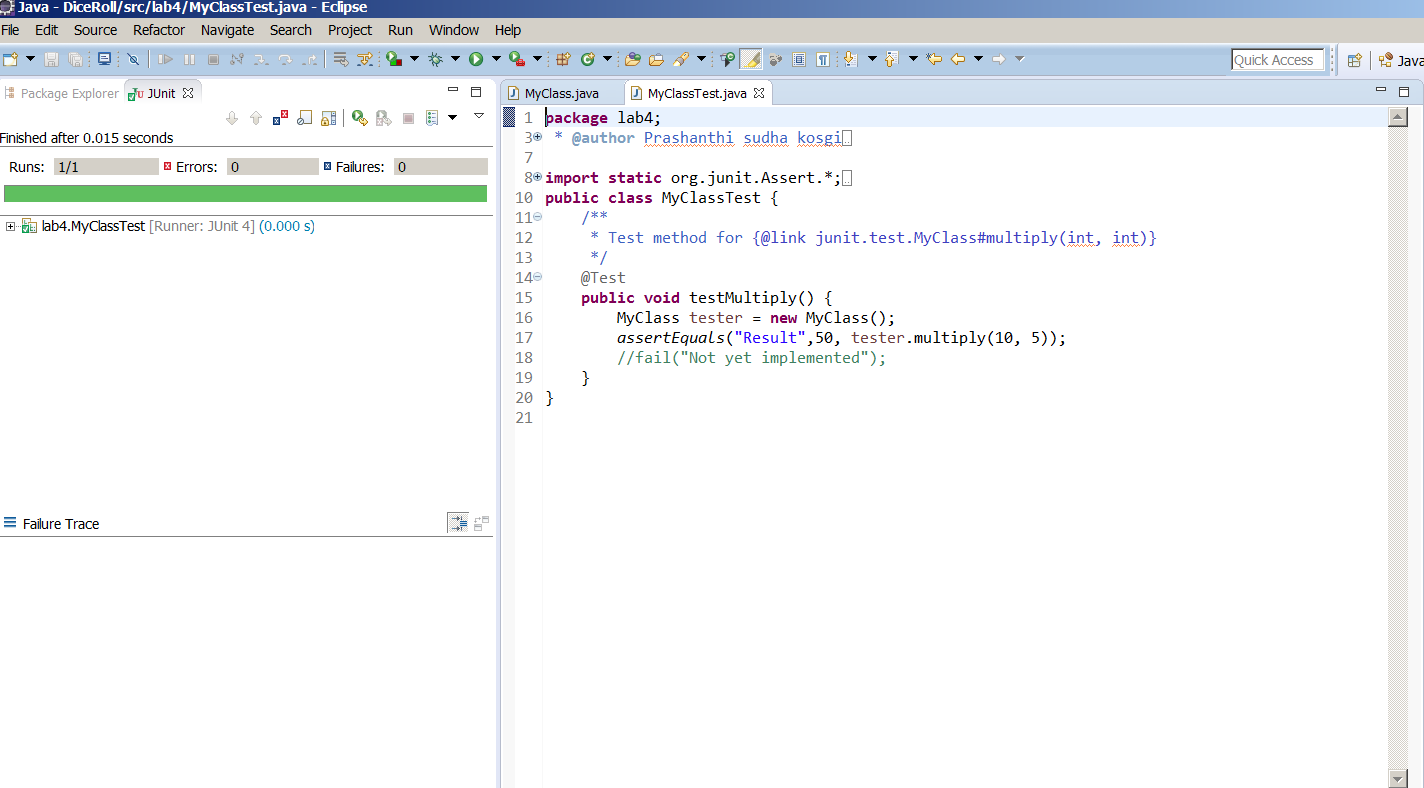
assertEquals("Result",50, tester.multiply(10, 5));

//fail("Not yet implemented");

}

}

**Output:**



**Does a green bar now show in the Viewer?**

Yes Junit test is passed and Green bar is shown (above picture).

**Ball class:**

**package** lab4Ball;

/\*\*

\* **@author** Prashanthi sudha kosgi

\* date :06/23/2017.

\*

\*/

**import** java.awt.Color;

**import** java.awt.Graphics;

**public** **class** Ball {

//declaring parameters.

**public** **int** size;

**public** **int** xIncrement;

**public** **int** yIncrement;

**public** **int** xPosition;

**public** **int** yPosition;

**public** Ball( **int** size, **int** increment ){

**this**.size = size;

**this**.xIncrement = increment;

**this**.yIncrement = increment;

xPosition = 100;

yPosition = 100;

}

//getters

**public** **int** getxIncrement() {

**return** xIncrement;

}

**public** **int** getyIncrement() {

**return** yIncrement;

}

//position getters

**public** **int** getTop() {

**return** yPosition;

}

**public** **int** getBottom() {

**return** yPosition + size;

}

**public** **int** getLeft() {

**return** xPosition;

}

**public** **int** getRight() {

**return** xPosition + size;

}

//Reversing horizontal direction

**public** **void** reverseDirectionX() {

xIncrement \*= -1;

}

//Reversing vertical direction

**public** **void** reverseDirectionY() {

yIncrement \*= -1;

}

//Returns horizontal center

**public** **int** getHorizontalCenter() {

**return** getLeft() + (size / 2);

}

//Returns vertical center

**public** **int** getVerticalCenter() {

**return** getTop() + (size / 2);

}

**public** **void** draw(Graphics graphics) {

graphics.setColor(Color.***RED***); //Color of the ball

graphics.fillOval(xPosition, yPosition, size, size); //Drawing the ball

}

**public** **void** changePosition() {

changeDirection(Display.***length***, Display.***height***);

//Changing the position

xPosition += xIncrement;

yPosition += yIncrement;

}

//Detecting the contact with window walls

**public** **void** changeDirection(**int** length, **int** height) {

**if** (getLeft() <= 0 || getRight() >= length)

reverseDirectionX(); //Changing horizontal direction

**else** **if** (getTop() <= 0 || getBottom() >= height)

reverseDirectionY(); //Changing vertical direction

}

}

**Paddle class**

**package** lab4Ball;

/\*\*

\* **@author** Prashanthi sudha kosgi

\* date :06/23/2017.

\*

\*/

**import** java.awt.Color;

**import** java.awt.Graphics;

**public** **class** Paddle {

// Declaring Properties

**private** **int** length;

**private** **int** height;

**private** **int** xPosition;

**private** **int** yPosition;

**public** Paddle(**int** length, **int** height, **int** xPosition, **int** yPosition) {

**this**.length = length;

**this**.height = height;

**this**.xPosition = xPosition;

**this**.yPosition = yPosition;

}

// Position getters

**public** **int** getTop() {

**return** yPosition - (height / 2);

}

**public** **int** getBottom() {

**return** yPosition + (height / 2);

}

**public** **int** getLeft() {

**return** xPosition - (length / 2);

}

**public** **int** getRight() {

**return** xPosition + (length / 2);

}

**public** **void** draw(Graphics graphics) {

graphics.setColor(Color.***BLUE***); // declaring color of the paddle

graphics.fillRect(getLeft(), getTop(), length, height); // Drawing paddle

}

**public** **void** changePosition(**int** x) {

// Detecting the edge of the window

**if** (x - (length / 2) >= 0 && x + (length / 2) <= Display.***length***)

xPosition = x; // Changing the position

}

}

**Display class**

**package** lab4Ball;

/\*\*

\* **@author** Prashanthi sudha kosgi

\* date :06/23/2017.

\*

\*/

**import** javax.swing.JPanel;

**import** java.awt.Color;

**import** java.awt.Graphics;

**import** javax.swing.JFrame;

**public** **class** Display **extends** JPanel {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**public** **static** **final** **int** ***length*** = 800; // length of frame

**public** **static** **final** **int** ***height*** = 400; // height of frame

**private** JFrame frame;

**private** Controller controller;

**public** Display(Controller controller) {

**this**.controller = controller;

frame = **new** JFrame("PADDLE BALL GAME");

setSize(***length***, ***height***);

frame.setSize(***length*** + 20, ***height*** + 40);

frame.add(**this**); // Adding game window to frame

frame.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

frame.setVisible(**true**);

}

@Override

**protected** **void** paintComponent(Graphics graphics) {

**super**.paintComponent(graphics);

graphics.setColor(Color.***BLACK***); // setting the color to the frame

graphics.fillRect(0, 0, getWidth(), getHeight()); // Drawing game window

controller.draw(graphics);

}

}

**Controller class:**

**package** lab4Ball;

/\*\*

\* **@author** Prashanthi sudha kosgi

\* date :06/23/2017.

\*

\*/

**import** java.awt.Graphics;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.event.MouseEvent;

**import** java.awt.event.MouseMotionListener;

**import** javax.swing.Timer;

**public** **class** Controller **implements** ActionListener, MouseMotionListener {

// creating references

**private** Ball ball;

**private** Paddle paddle;

**private** Display display;

**private** Timer timer;

**public** Controller() {

// Instantiating the components

ball = **new** Ball(40, 3);

paddle = **new** Paddle(100, 12, 300, 30);

display = **new** Display(**this**);

timer = **new** Timer(25, **this**);

timer.start(); // Starting the Timer for Ball

display.addMouseMotionListener(**this**); // Registering the mouse event for paddle

}

@Override

**public** **void** actionPerformed(ActionEvent actionEvent) {

// Detecting the contact between paddle and ball

**if** (ball.getTop() == paddle.getBottom()

|| (ball.getTop() < paddle.getBottom() && ball.getTop() >= paddle.getTop()))

**if** (ball.getHorizontalCenter() >= paddle.getLeft() && ball.getHorizontalCenter() <= paddle.getRight())

ball.reverseDirectionY(); // Reversing the ball after contact

ball.changePosition(); // updating the ball position

display.repaint(); // repainting the game window

}

@Override

**public** **void** mouseDragged(MouseEvent mouseDraggedEvent) {

paddle.changePosition(mouseDraggedEvent.getX()); // repainting the game window

display.repaint(); // repainting the game window

}

@Override

**public** **void** mouseMoved(MouseEvent mouseMovedEvent) {

paddle.changePosition(mouseMovedEvent.getX()); // Updating the paddle position

display.repaint(); // repainting the game window

}

**public** **void** draw(Graphics graphics) {

**if** (ball != **null**)

ball.draw(graphics); // Drawing the ball

**if** (paddle != **null**)

paddle.draw(graphics); // Drawing the paddle

}

}

**PaddleBallGame class**

**package** lab4Ball;

/\*\*

\* **@author** Prashanthi sudha kosgi

\* date :06/23/2017.

\*

\*/

**public** **class** PaddleBallGame {

**public** **static** **void** main(String[] args) {

//Starting the game

**new** Controller();

}

}

**Output:**

