

Assignment 12

Bollepalli Sai Sreekanth

1A.

```
package sre;
interface Shape{
    void draw();
}
class Rectangle implements Shape{

    @Override
    public void draw() {
        System.out.println("This is Rectangle class, draw method");
    }

}
class Square implements Shape{

    @Override
    public void draw() {
        System.out.println("This is Square class, draw method");
    }

}
class RoundedRectangle implements Shape{

    @Override
    public void draw() {
        System.out.println("This is RoundedRectangle class, draw
method");
    }

}
class RoundedSquare implements Shape{

    @Override
    public void draw() {
        System.out.println("This is RoundedSquare class, draw
method");
    }

}
abstract class AbstractFactory{
    abstract Shape shape(String type) ;
}
class ShapeFactory extends AbstractFactory{
```

```

@Override
Shape shape(String type) {
    if(type.equalsIgnoreCase("RECTANGLE")){
        return new Rectangle();
    }else if(type.equalsIgnoreCase("SQUARE")){
        return new Square();
    }
    return null;
}

}

class RoundedShapeFactory extends AbstractFactory{

    @Override
    Shape shape(String type) {
        if(type.equalsIgnoreCase("RECTANGLE")){
            return new RoundedRectangle();
        }else if(type.equalsIgnoreCase("SQUARE")){
            return new RoundedSquare();
        }
        return null;
    }

}

class FactoryProducer {
    public static AbstractFactory getFactory(boolean rounded){
        if(rounded){
            return new RoundedShapeFactory();
        }else{
            return new ShapeFactory();
        }
    }
}

public class AbstractFactoryPatternDemo {

    public static void main(String[] args) {
        AbstractFactory shapeFactory = FactoryProducer.getFactory(false);
        Shape shape1 = shapeFactory.shape("RECTANGLE");
        shape1.draw();
        Shape shape2 = shapeFactory.shape("SQUARE");
        shape2.draw();
        AbstractFactory shapeFactory1 = FactoryProducer.getFactory(true);
        Shape shape3 = shapeFactory1.shape("RECTANGLE");
        shape3.draw();
        Shape shape4 = shapeFactory1.shape("SQUARE");
        shape4.draw();
    }

}

```

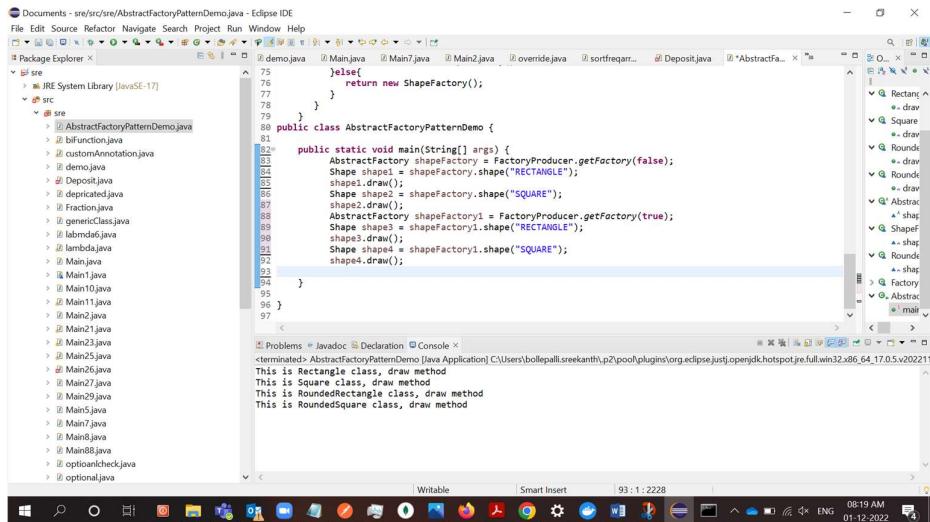
Output:

This is Rectangle class, draw method

This is Square class, draw method

This is RoundedRectangle class, draw method

This is RoundedSquare class, draw method



The screenshot shows the Eclipse IDE with the file `AbstractFactoryPatternDemo.java` open. The code defines an `AbstractFactory` interface, a `FactoryProducer` class, and a `AbstractFactoryPatternDemo` class. The `main` method in `AbstractFactoryPatternDemo` creates a `ShapeFactory` and uses it to create and draw four shapes: `Rectangle`, `Square`, `RoundedRectangle`, and `RoundedSquare`. The console output at the bottom shows the execution results, confirming that each shape's `draw` method is called.

```
75 }else{
76     return new ShapeFactory();
77 }
78 }
79 }
80
81 public class AbstractFactoryPatternDemo {
82
83     public static void main(String[] args) {
84         AbstractFactory shapeFactory = FactoryProducer.getFactory(false);
85         Shape shape1 = shapeFactory.shape("RECTANGLE");
86         shape1.draw();
87         Shape shape2 = shapeFactory.shape("SQUARE");
88         shape2.draw();
89         AbstractFactory shapeFactory1 = FactoryProducer.getFactory(true);
90         Shape shape3 = shapeFactory1.shape("RECTANGLE");
91         shape3.draw();
92         Shape shape4 = shapeFactory1.shape("SQUARE");
93         shape4.draw();
94     }
95 }
96 }
97 }
```

Console Output:

```
<terminated> AbstractFactoryPatternDemo [Java Application] C:\Users\bollepalli.sreekanth\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.17.0.5.v202211
This is Rectangle class, draw method
This is Square class, draw method
This is RoundedRectangle class, draw method
This is RoundedSquare class, draw method
```