Implementing Singleton Principle

package singletondemo;

public class Singleton {

    private static Singleton instance;

    private Singleton() {

        System.out.println("Singleton instance created.");

    }

    public static synchronized Singleton getInstance() {

        if (instance == null) {

            instance = new Singleton();

        }

        return instance;

    }

    public void showMessage() {

        System.out.println("Hello from Singleton!");

    }

}

package singletondemo;

public class Main {

    public static void main(String[] args) {

        Singleton s1 = Singleton.getInstance();

        Singleton s2 = Singleton.getInstance();

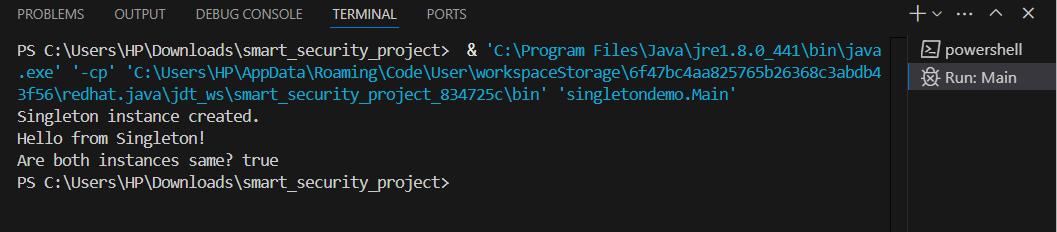
        s1.showMessage();

        System.out.println("Are both instances same? " + (s1 == s2));

    }

}

Output:



Implementing the Factory Method Pattern

Shape.java

package factorypattern;

public interface Shape {

    void draw();

}

Circle.java

package factorypattern;

public class Circle implements Shape {

    public void draw() {

        System.out.println("Drawing a Circle");

    }

}

Square.java

package factorypattern;

public class Square implements Shape {

    public void draw() {

        System.out.println("Drawing a Square");

    }

}

Shapefactory.java

package factorypattern;

public class ShapeFactory {

    public Shape getShape(String shapeType) {

        if (shapeType == null) return null;

        if (shapeType.equalsIgnoreCase("CIRCLE")) {

            return new Circle();

        } else if (shapeType.equalsIgnoreCase("SQUARE")) {

            return new Square();

        }

        return null;

    }

}

Main.java

package factorypattern;

public class Main {

    public static void main(String[] args) {

        ShapeFactory factory = new ShapeFactory();

        Shape shape1 = factory.getShape("CIRCLE");

        shape1.draw();

        Shape shape2 = factory.getShape("SQUARE");

        shape2.draw();

    }

}

Output:

