**Exercise 1: Implementing the Singleton Pattern**

Code:

package DesignPatternsAndPrinciples.Singleton;

public class Singleton {

  // Step 1: Create a private static instance variable

  private static Singleton instance;

  // Step 2: Make the constructor private to prevent instantiation from outside

  private Singleton() {

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

  }

  // Step 3: Provide a public static method to get the single instance (Lazy

  // Initialization)

  public static Singleton getInstance() {

    if (instance == null) {

      instance = new Singleton(); // Create the instance only if it doesn't exist

    }

    return instance;

  }

  // Step 4: Example method

  public void showMessage() {

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

  }

}

package DesignPatternsAndPrinciples.Singleton;

// Main.java

public class Main {

  public static void main(String[] args) {

    // Get the only object from Singleton class

    Singleton obj1 = Singleton.getInstance();

    obj1.showMessage();

    // Try getting another reference to check if it's the same object

    Singleton obj2 = Singleton.getInstance();

    obj2.showMessage();

    // Compare both references

    if (obj1 == obj2) {

      System.out.println("Both references point to the same instance.");

    } else {

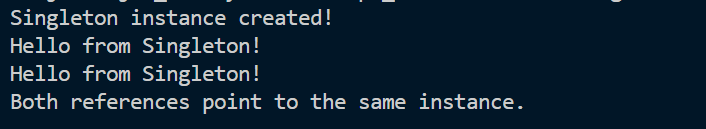
      System.out.println("Different instances (this shouldn't happen).");

    }

  }

}

**Output:**



**Exercise 2: Implementing the Factory Method Pattern**

**Code:**

package DesignPatternsAndPrinciples.FactoryMethod;

public interface Shape {

  void draw();

}

package DesignPatternsAndPrinciples.FactoryMethod;

public class Circle implements Shape {

  public void draw() {

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

  }

}

package DesignPatternsAndPrinciples.FactoryMethod;

public class Square implements Shape {

  public void draw() {

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

  }

}

package DesignPatternsAndPrinciples.FactoryMethod;

public class ShapeFactory {

    // Factory method

    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;

    }

}

package DesignPatternsAndPrinciples.FactoryMethod;

public class Main {

  public static void main(String[] args) {

    ShapeFactory shapeFactory = new ShapeFactory();

    // Get Circle

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

    shape1.draw(); // Output: Drawing a Circle

    // Get Square

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

    shape2.draw(); // Output: Drawing a Square

    // Invalid shape

    Shape shape3 = shapeFactory.getShape("TRIANGLE");

    if (shape3 == null) {

      System.out.println("Invalid shape type.");

    }

  }

}

**Output:A blue background with white dots

AI-generated content may be incorrect.**