**Two use cases to demonstrate two creational design pattern.**

1. **Database Connection**

public class DatabaseConnection {

// Example property for the database connection

private String connectionString;

// Static instance pointer

private static DatabaseConnection instance = null;

// Private constructor to prevent instantiation

private DatabaseConnection() {

this.connectionString = "Database=MyDB;User=admin;Password=admin;";

System.out.println("Database Connection created with connection string: " + this.connectionString);

}

// Function to get the singleton instance

public static DatabaseConnection getDatabaseConnection() {

if (instance == null) {

instance = new DatabaseConnection();

}

return instance;

}

// Function to clean up the singleton instance

public static void cleanupDatabaseConnection() {

if (instance != null) {

instance = null;

System.out.println("Database Connection cleaned up.");

}

}

public String getConnectionString() {

return connectionString;

}

public static void main(String[] args) {

// Get the singleton instance of the DatabaseConnection

DatabaseConnection dbConn1 = DatabaseConnection.getDatabaseConnection();

System.out.println("Accessing connection string: " + dbConn1.getConnectionString());

// Attempt to get another instance

DatabaseConnection dbConn2 = DatabaseConnection.getDatabaseConnection();

System.out.println("Accessing connection string: " + dbConn2.getConnectionString());

// Clean up

DatabaseConnection.cleanupDatabaseConnection();

}

}

1. **Shape Creation**

import java.util.Objects;

// Abstract Shape class

abstract class Shape {

abstract void draw();

}

// Concrete implementations

class Circle extends Shape {

@Override

void draw() {

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

}

}

class Rectangle extends Shape {

@Override

void draw() {

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

}

}

// Factory class

class ShapeFactory {

public static Shape createShape(String type) {

if (Objects.equals(type, "circle")) {

return new Circle();

} else if (Objects.equals(type, "rectangle")) {

return new Rectangle();

} else {

return null;

}

}

}

public class Main {

public static void main(String[] args) {

Shape shape1 = ShapeFactory.createShape("circle");

if (shape1 != null) {

shape1.draw();

}

Shape shape2 = ShapeFactory.createShape("rectangle");

if (shape2 != null) {

shape2.draw();

}

}

}