**Day 23 Core\_Java Assignments**

[**digvijaythakare2017@gmail.com**](mailto:digvijaythakare2017@gmail.com)

**Task 1: Singleton Implement a Singleton class that manages database connections. Ensure the class adheres strictly to the singleton pattern principles.**

**Code-**

package com.epwipro.day23;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class DatabaseConnectionManager {

// Private static instance variable (Lazy Initialization)

private static DatabaseConnectionManager *instance*;

// Private connection variable

private Connection connection;

// Database URL, username and password

private final String url = "jdbc:mysql://localhost:3306/testdb";

private final String username = "root";

private final String password = "root";

// Private constructor to prevent instantiation

private DatabaseConnectionManager() throws SQLException {

try {

// Load MySQL JDBC driver

Class.*forName*("com.mysql.cj.jdbc.Driver");

this.connection = DriverManager.*getConnection*(url, username, password);

} catch (ClassNotFoundException ex) {

throw new SQLException(ex);

}

}

// Public static method to get the instance of the class

public static DatabaseConnectionManager getInstance() throws SQLException {

if (*instance* == null) {

synchronized (DatabaseConnectionManager.class) {

if (*instance* == null) {

*instance* = new DatabaseConnectionManager();

}

}

}

return *instance*;

}

// Public method to get the connection

public Connection getConnection() {

return connection;

}

// Method to close the connection

public void closeConnection() {

if (connection != null) {

try {

connection.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

// Main method for testing the singleton

public static void main(String[] args) {

try {

DatabaseConnectionManager manager = DatabaseConnectionManager.*getInstance*();

Connection conn = manager.getConnection();

if (conn != null && !conn.isClosed()) {

System.***out***.println("Successfully connected to the database.");

// Perform database operations if needed

} else {

System.***out***.println("Failed to connect to the database.");

}

// Close the connection

manager.closeConnection();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

**Output-**

Successfully connected to the database.

**Task 2: Factory Method Create a ShapeFactory class that encapsulates the object creation logic of different Shape objects like Circle, Square, and Rectangle."**

**Code 1- Shape interface**

package com.epwipro.day23;

public interface Shape {

void draw();

}

**Code 2-Circle class**

package com.epwipro.day23;

public class Circle implements Shape {

*@Override*

public void draw() {

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

}

}

**Code 3 Square class**

package com.epwipro.day23;

public class Square implements Shape {

*@Override*

public void draw() {

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

}

}

**Code 4- Rectangle class**

package com.epwipro.day23;

public class Rectangle implements Shape {

*@Override*

public void draw() {

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

}

}

**Code 5-Shape Factory class-**

package com.epwipro.day23;

public class ShapeFactory {

// Method to get an instance of Shape based on the given shape type

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();

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

return new Rectangle();

}

return null;

}

}

**Code 6 Factory Pattern Demo**

package com.epwipro.day23;

public class FactoryPatternDemo {

public static void main(String[] args) {

ShapeFactory shapeFactory = new ShapeFactory();

// Get an object of Circle and call its draw method

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

shape1.draw();

// Get an object of Square and call its draw method

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

shape2.draw();

// Get an object of Rectangle and call its draw method

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

shape3.draw();

}

}

**Output-**

Drawing a Circle

Drawing a Square

Drawing a Rectangle