# Design PRINCIPLES AND PATTERN

# 1.Implement SingleTon PATTERN (5 TYPES)

class EagerSingleton {

private static final EagerSingleton instance = new EagerSingleton(); private EagerSingleton() {}

public static EagerSingleton getInstance() { return instance;

}

}

class LazySingleton {

private static LazySingleton? instance; private LazySingleton() {}

public static LazySingleton getInstance() { if (instance == null) {

instance = new LazySingleton();

}

return instance;

}

}

class ThreadSafeSingleton {

private static ThreadSafeSingleton? instance; private ThreadSafeSingleton() {}

public static synchronized ThreadSafeSingleton getInstance() { if (instance == null) {

instance = new ThreadSafeSingleton();

}

return instance;

}

}

class DoubleCheckedLockingSingleton {

private static volatile DoubleCheckedLockingSingleton? instance; private DoubleCheckedLockingSingleton() {}

public static DoubleCheckedLockingSingleton getInstance() { if (instance == null) {

synchronized (DoubleCheckedLockingSingleton.class) { if (instance == null) {

instance = new DoubleCheckedLockingSingleton();

}

}

}

return instance;

}

}

class BillPughSingleton { private BillPughSingleton() {} private static class Helper {

private static final BillPughSingleton INSTANCE = new BillPughSingleton();

}

public static BillPughSingleton getInstance() { return Helper.INSTANCE;

}

}

public class Singleton {

public static void main(String[] args) {

System.out.println("Eager Singleton:"); EagerSingleton eager1 = EagerSingleton.getInstance(); EagerSingleton eager2 = EagerSingleton.getInstance();

System.out.println("Same instance? " + (eager1 == eager2));

System.out.println("\nLazy Singleton:"); LazySingleton lazy1 = LazySingleton.getInstance(); LazySingleton lazy2 = LazySingleton.getInstance();

System.out.println("Same instance? " + (lazy1 == lazy2));

System.out.println("\nThread-Safe Singleton:");

ThreadSafeSingleton threadSafe1 = ThreadSafeSingleton.getInstance(); ThreadSafeSingleton threadSafe2 = ThreadSafeSingleton.getInstance(); System.out.println("Same instance? " + (threadSafe1 == threadSafe2));

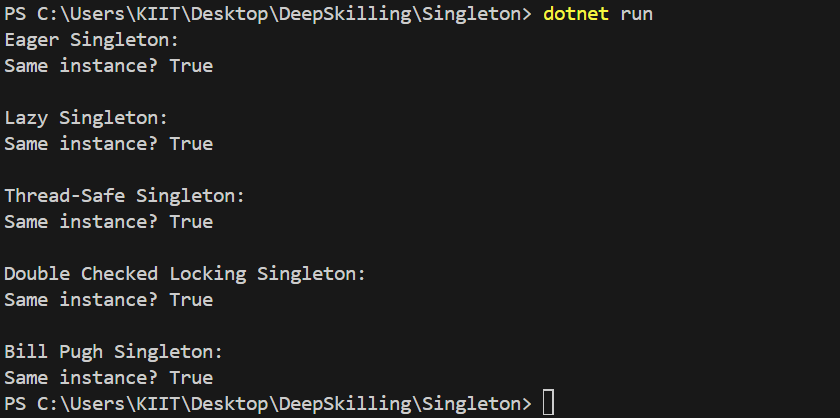
System.out.println("\nDouble Checked Locking Singleton:"); DoubleCheckedLockingSingleton double1 = DoubleCheckedLockingSingleton.getInstance(); DoubleCheckedLockingSingleton double2 = DoubleCheckedLockingSingleton.getInstance(); System.out.println("Same instance? " + (double1 == double2));

System.out.println("\nBill Pugh Singleton:"); BillPughSingleton pugh1 = BillPughSingleton.getInstance(); BillPughSingleton pugh2 = BillPughSingleton.getInstance(); System.out.println("Same instance? " + (pugh1 == pugh2));

}

}

**OUTPUT**



# 2.IMPLEMENT FACTORY METHOD PATTERN

**using System;**

abstract class Shape

{

public abstract void Draw();

}

class Circle : Shape

{

public override void Draw()

{

Console.WriteLine("Drawing a Circle");

}

}

class Rectangle : Shape

{

public override void Draw()

{

Console.WriteLine("Drawing a Rectangle");

}

}

class Square : Shape

{

public override void Draw()

{

Console.WriteLine("Drawing a Square");

}

}

interface IShapeFactory

{

Shape CreateShape();

}

class CircleFactory : IShapeFactory

{

public Shape CreateShape()

{

return new Circle();

}

}

class RectangleFactory : IShapeFactory

{

public Shape CreateShape()

{

return new Rectangle();

}

}

class SquareFactory : IShapeFactory

{

public Shape CreateShape()

{

return new Square();

}

}

class Client

{

private Shape shape;

public Client(IShapeFactory factory)

{

shape = factory.CreateShape();

}

public Shape GetShape()

{

return shape;

}

}

class FactoryDesign

{

public static void Main(string[] args)

{

IShapeFactory circleFactory = new CircleFactory();

Client circleClient = new Client(circleFactory);

circleClient.GetShape().Draw();

IShapeFactory rectangleFactory = new RectangleFactory();

Client rectangleClient = new Client(rectangleFactory);

rectangleClient.GetShape().Draw();

IShapeFactory squareFactory = new SquareFactory();

Client squareClient = new Client(squareFactory);

squareClient.GetShape().Draw();

}

}

**OUTPUT**

