SDA Lab 12

22k-5195 Laiba Fatima

**Q1.**

class Singleton {

private static Singleton obj;

// private constructor to force use of

// getInstance() to create Singleton object

private Singleton() {}

public static Singleton getInstance() {

if (obj==null)

obj = new Singleton();

return obj;

}

}

**Ans:** Singleton

**Q2.**

class Singleton {

private static Singleton obj;

private Singleton() {}

// Only one thread can execute this at a time

public static synchronized Singleton getInstance() {

if (obj==null)

obj = new Singleton();

return obj;

}

}

**Ans:** Singleton (Thread Synchronized Java implementation of singleton design pattern)

**Q3.**

class {

Singleton

private static Singleton obj = new Singleton();

private Singleton() {}

public {

static Singleton getInstance()

return obj;

}

}

**Ans:** : Singleton (Static initializer based Java implementation of singleton design pattern)

**Q4.**

public class SingletonExample {

// Static member holds only one instance of the

// SingletonExample class

private static SingletonExample singletonInstance;

// SingletonExample prevents any other class from instantiating

private SingletonExample() {

}

// Providing Global point of access

public static SingletonExample getSingletonInstance() {

if (null singletonInstance) {

singletonInstance new SingletonExample();

}

return singletonInstance;

}

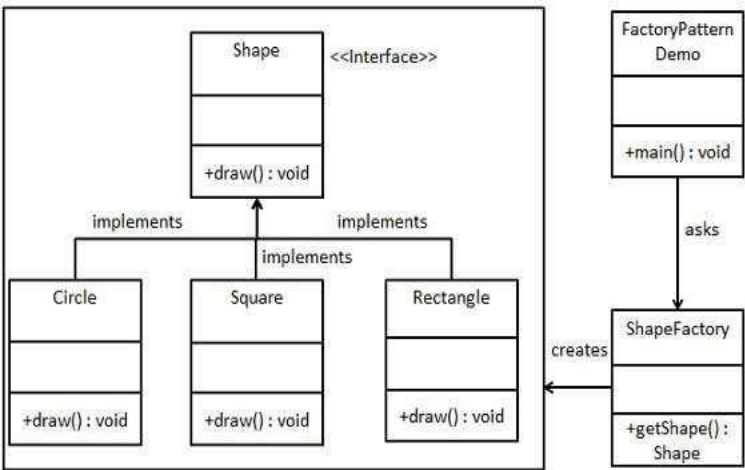
public void printSingleton(){

System.out.println("Inside print Singleton");

}

**Ans**: Singleton

**Q5.**

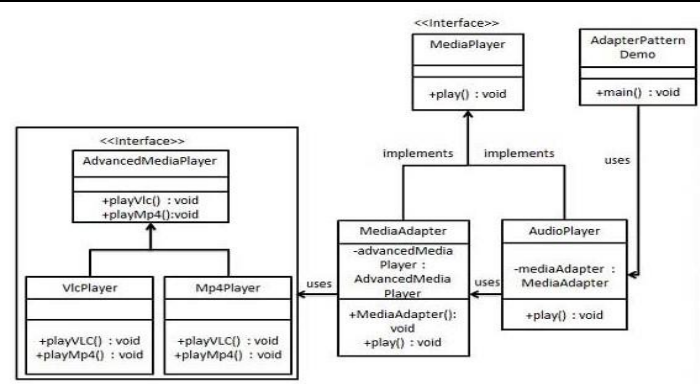
We're going to create a Shape interface and concrete interface. classes implementing the Shape

• A factory class ShapeFactory is defined as a next step.

• FactoryPatternDemo, our demo class will use ShapeFactory to get a Shape object. It will pass information (CIRCLE / RECTANGLE / SQUARE)to ShapeFactory to get the type of object it needs.

**Ans:** The described scenario outlines the Factory Method design pattern. This pattern encapsulates object creation by defining a separate method for creating objects. In this case, ShapeFactory serves as the factory class responsible for creating instances of various shapes based on the input provided by FactoryPatternDemo.

**Q6.**



**Ans:** Adapter