Niyati B Mehta

SAU Batch-1

DESIGN PRINCIPLE AND PATTERN

1. You have a Smartphone class and will have derived classes like iPhone, Android Phone, Windows Mobile Phone can be even phone names with brand, how would you design this system of Classes.

Ans:

-> The Design Pattern that can be used is the Factory Pattern in the above scenario.

-> For the above example there will be a parent abstract class named Smart Phone having member attributes Model Number, Model Name and Year of Release.

-> It will have member functions of finding the Name of the Phone, Calling, and Check battery status.

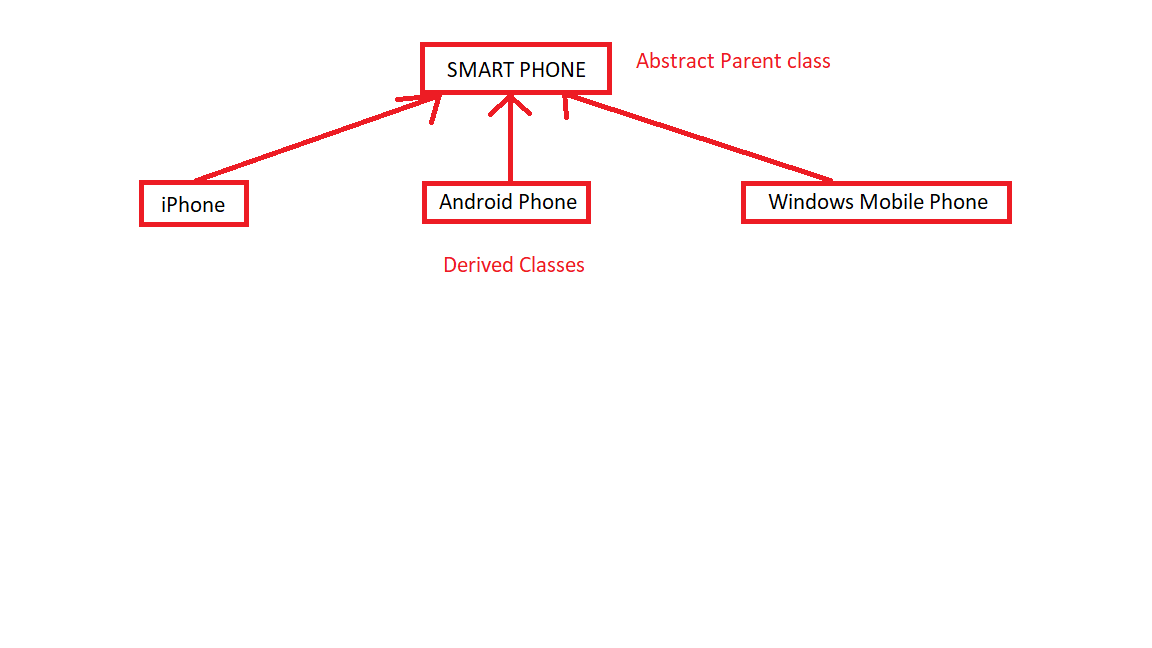
-> iPhone, Android and Windows as child classes derived from the Smart Phone class.

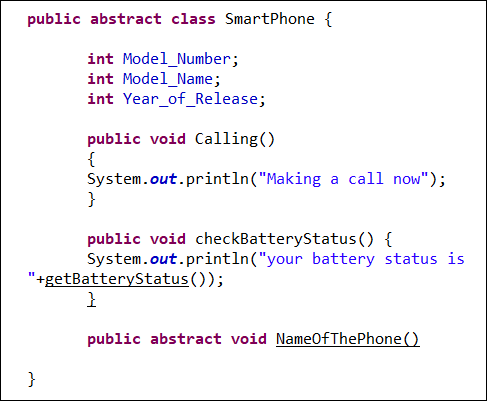
-> The advantage of this design pattern is that any new model can be added without any major changes to the Program.

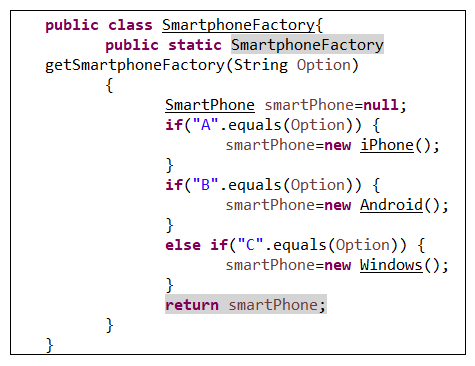
-> We have a Smartphone factory that gives the appropriate phone based on the choice selected.

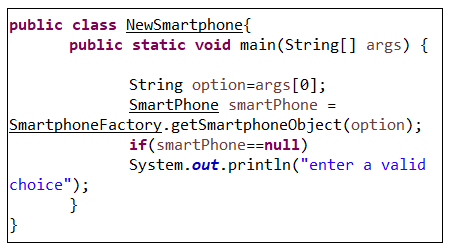
-> Option A gives an Iphone, Option B gives an Android Phone and Option C gives a Windows Phone.

-> The abstract class has one abstract method of find the name based on the phone selected.



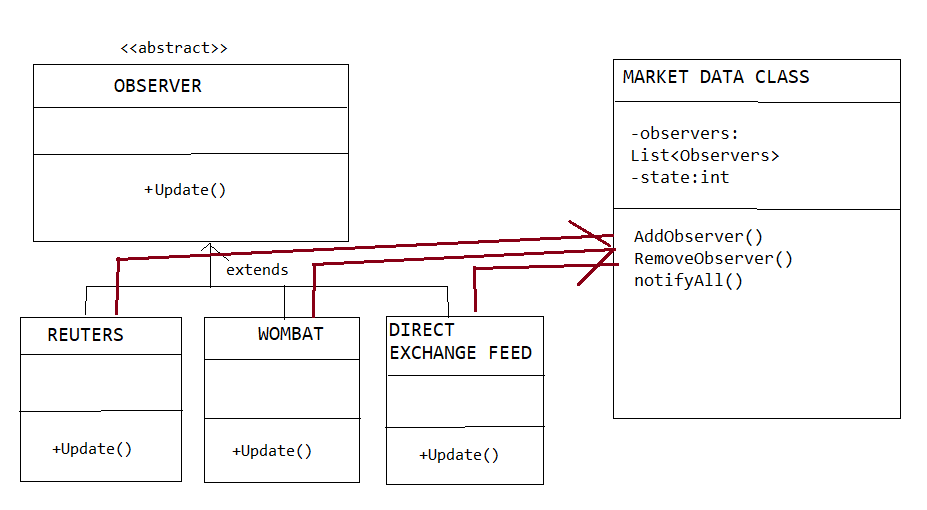




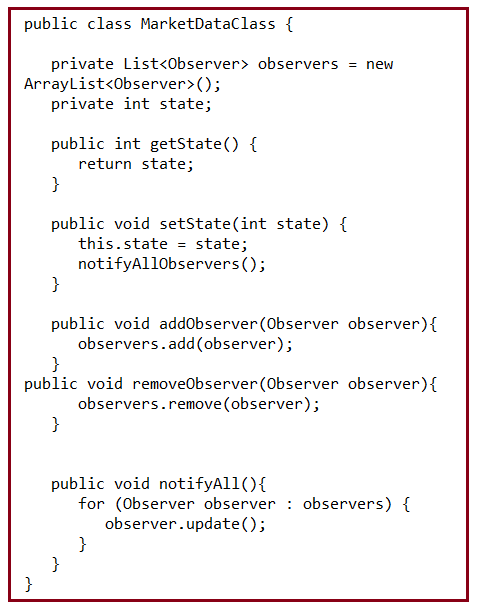


1. Write classes to provide Market Data and you know that you can switch to different vendors overtime like Reuters, wombat and may be even to direct exchange feed , how do you design your Market Data system.

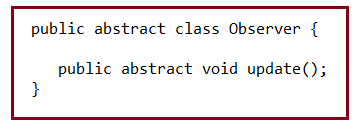
Ans: We can use the Observer Design Pattern for the above scenario because one-to-many relationship between objects can be done.



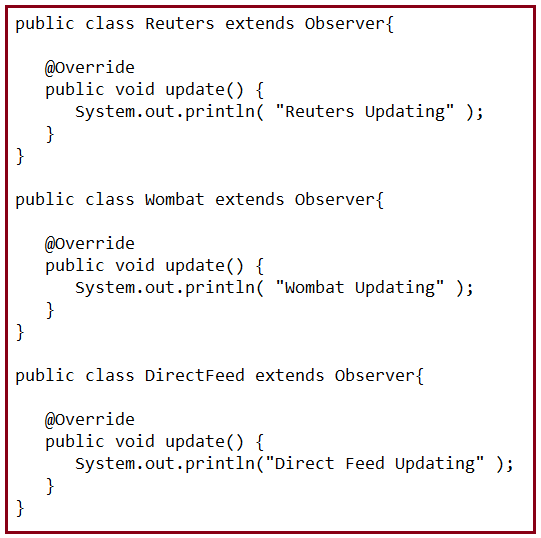
Market Data Class



Abstract Observer Class



Reuters, Wombat and Direct Exchange Feed Class



1. What is Singleton design pattern in Java ? write code for thread-safe singleton in Java and handle Multiple Singleton cases shown in slide as well.

Ans:

This pattern involves a single class which is responsible to create an object while making sure that only single object gets created. This class provides a way to access its only object which can be accessed directly without need to instantiate the object of the class.

Eager initialization: This is the simplest method of creating a singleton class. In this, object of class is created when it is loaded to the memory by JVM. It is done by assigning the reference an instance directly.

// Eager Initialization

public class singleton

{

// public instance initialized when loading the class

private static final GFG instance = new singleton ();

private singleton ()

{

// private constructor

}

public static singleton getInstance(){

return instance;

}

}

Lazy initialization: In this method, object is created only if it is needed. This may prevent resource wastage.

// With Lazy initialization

public class singleton

{

// private instance, so that it can be

// accessed by only by getInstance() method

private static singleton instance;

private singleton ()

{

// private constructor

}

//method to return instance of class

public static singleton getInstance()

{

if (instance == null)

{

// if instance is null, initialize

instance = new singleton ();

}

return instance;

}

}

Thread Safe Singleton: A thread safe singleton in created so that singleton property is maintained even in multithreaded environment. To make a singleton class thread-safe, getInstance() method is made synchronized so that multiple threads can’t access it simultaneously.

// Java program to create Thread Safe

// Singleton class

public class Java

{

// private instance, so that it can be

// accessed by only by getInstance() method

private static Java instance;

private Java()

{

// private constructor

}

//synchronized method to control simultaneous access

synchronized public static Java getInstance()

{

if (instance == null)

{

// if instance is null, initialize

instance = new Java();

}

return instance;

}

}

1. Design classes for Builder Pattern.

Builder pattern aims to “Separate the construction of a complex object from its representation so that the same construction process can create different representations.” It is used to construct a complex object step by step and the final step will return the object. The process of constructing an object should be generic so that it can be used to create different representations of the same object.

