#### Q1. What are Design Patterns?

* Design Pattern provide us the template to create the object, to establish the communication between obect or if we want object of class A in class B(this is called composition)
* The design pattern was provided by 4 people in 1990s called as Gang of Four in their book
* Design patterns are based on below two principles
  + We should always prefer Composition over Inheritance
  + Code for Interface and not for implementation
* There are three type of design pattern
  + Creational Design Pattern: Whenever we want to create the object of class we can use Creational Design pattern. Some of creational design patterns are Singleton,Prototype etc
  + Structural Design Pattern: Wheneve we want to compose one object in another object we can use Structural design pattern eg Facade, Composite etc

#### Behavioral: Whenver we want to establish communication between two object we can use behavioral pattern eg Observer etc

#### Q2. What is Singleton Design Pattern?

* It is a creational design pattern which provide us the object of an class
* In Singleton design pattern only a single object is created for a class
* Other classes will share the same single instance of the class
* Singleton Pattern is used in Logging,Session, Driver
* To make the class as Singleton in Java we should follow below rules

###### Constructor should be private

###### **There should be one method which will return the object of class**

###### **The method should be declared as static and public**

###### There can be eager initialization where the object is created once the class is loaded in Java Program. We will create the object when declare it

* In lazy initailization the object is created whenever required. There will be public method which will return object of same class. In this method we will check the declared object is null or not
* There can be condition when two threads are trying to create the object, In that case we should make the method syncronized Which return the object

**class** Singleton{

**public** **static** Singleton *ss* ; //lazy ininitalizition

//public static Singleton ss = new Singleton(); //eager initialization

**private** Singleton() {

}

**public** **synchronized** **static** Singleton getInstance() {

**if**(*ss* == **null**) {

*ss*= **new** Singleton();

**return** *ss*;

}

**else**

**return** *ss*;

}

}

#### Q3. What is Factory Design Pattern?

* It is a creational design pattern which provide us the object of an class
* It is used when there are multiple subclass of one Parent Class eg there is Parent Class Vehicle and subchild classes like Bike,Car,Truck
* In factory design pattern we do not directly create the object of child class. We will provide the type of input and factory class will give the object of child class
* To implement the factory class below points are considered
  + Parent class can be normal class, interface or abstract class
  + Factory class have static method that will return object of child class depending on input parameter
  + The Factory class should have Factory mentiond in his name

#### Q4. What is Prototype Design Pattern?

* It is a creational design pattern which provide us the object of an class
* It is used when we want to avoid multiple object creation of same class. Instead of creating the new object we can copy or clone the existing object
* To implement the factory class below points are considered
  + Object or class which needs to be copied should implement Cloneable Interface
  + We can override clone method to implement as per our need

**import** java.util.\*;

**class** Vhicle **implements** Cloneable{

**public** List vehicleList = **new** ArrayList();

**public** Vhicle() {

}

**public** Vhicle(List a) {

**this**.vehicleList = a;

}

**public** **void** insertData() {

**this**.vehicleList.add("a");

**this**.vehicleList.add("b");

}

**public** Object clone() {

List tempList = **this**.vehicleList;

**return** **new** Vhicle(tempList);

}

}

**public** **class** ProtoTypeDesignPattern {

**public** **static** **void** main(String[] args) {

Vhicle v1 = **new** Vhicle();

v1.insertData();

System.***out***.println(v1.vehicleList);

Vhicle v2 = (Vhicle) v1.clone();

v2.vehicleList.remove(0);

System.***out***.println(v2.vehicleList);

}

}