JAVA {Short Notes}

Q. What is OOP? Name some OOP Languages.

Object-Oriented Programming is a **modular approach**. Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects. It simplifies the software development and maintenance by providing some concepts:

- Object
- Class
- Inheritance
- Polymorphism
- Abstraction
- Encapsulation
- Dynamic Binding

OOP Languages- **C++, Java, Small talk, Visual Basic .NET, Ruby, C#** etc.

Q. Basic Features of Java

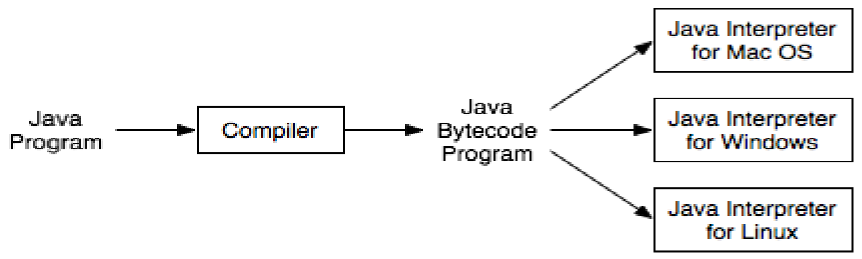
- Java is an **Object oriented** programming language.
- Java program is both **compiled and interpreted**
- Java programs are **platform independent**
- Java is a **multithreaded language**.
- Java is a **case sensitive** language

Q. Difference between Compiler and Interpreter

S No.	Compiler	Interpreter
1	Compiler Takes **Entire** program as input.	Interpreter Takes **Single** instruction as input.
2	**Errors** are displayed after **entire program** is checked	**Errors** are displayed for **every instruction** interpreted (if any)
3	Conditional Control Statements are Executes **faster**	Conditional Control Statements are Executes **slower**
4	**Memory Requirement** is **More**	**Memory Requirement** is **Less**
5	Intermediate Object Code is **Generated**	**No** Intermediate Object Code is **Generated**
6	**Example** : C Compiler	**Example** : BASIC

Q. Byte code and JVM

Java is **a high level language.**
The program written in Java is compiled for conversion to an intermediate code called **Byte Code.** This code is **independent** of the machine on which the program is to run. This makes a Java program highly portable as its Bytes codes can easily be transferred from one system to another.
When this byte code is to be run on any system, an **interpreter,** known as **Java Virtual Machine(JVM**) is needed which translates the **byte code to machine code.**


Java Program
Compiler
Java Bytecode Program
Java Interpreter for Mac OS
Java Interpreter for Windows
Java Interpreter for Linux

Q. Object

- Objects are the basic run-time entities in an object-oriented system.
- When an object is brought into existence , it is called **Instantiation.**
- Programming problem is analyzed in terms of objects and nature of communication between them.
- When a program is executed, objects interact with each other by sending messages.
- Different objects can also interact with each other without knowing the details of their data or code.
- Real-world objects share two characteristics: They all have ***state*** and ***behavior*.**
  e.g. Bicycles have state (current gear, current speed, etc) and behavior (changing gear, applying brakes, etc)

Q. Class

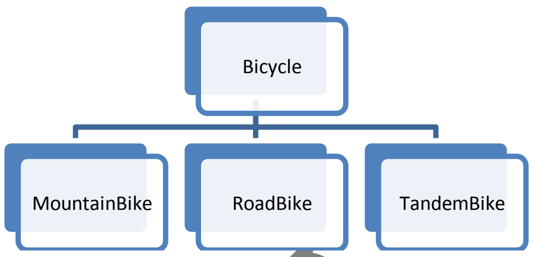
- A class is a collection of objects of similar type. Once a class is defined, any number of objects can be created which belong to that class.
- Class is a **blueprint or factory** that describes the nature of an object.
- Class is a **user defined or composite** date type.
- Each object of a class possesses same attributes and common behavior within the same class.

Note: Data members and methods (functions) contained within a class are termed as attributes of a class

Q. Inheritance

- Inheritance is the process by which objects can acquire the properties of objects of other class.

- In OOP, inheritance provides reusability, like, adding additional features to an existing class without modifying it.
- This is achieved by deriving a new class from the existing one. The new class will have combined features of both the classes.
- **Reusability**: Object-oriented programming allows classes to *inherit* commonly used state and behavior from other classes.
- Example, Bicycle is the *superclass* of MountainBike, RoadBike, and TandemBike.



- In the Java programming language, each class is allowed to have one direct **superclass** and each superclass has the potential for an unlimited number of ***subclasses*.**

Q. Polymorphism

- Polymorphism means the ability to take more than one form.
- Polymorphism is derived from 2 greek words: **poly and morphs**. The word "poly" means many and "morphs" means forms. So polymorphism means many forms.
- An operation may exhibit different behaviors in different instances. The behavior depends on the data types used in the operation.

We can perform polymorphism in java by method overloading and method overriding.

Method Overloading:

- In Java, it is possible to define two or more methods of same name in a class, provided that there argument list or parameters are different.
- When Java encounters a call to an overloaded method, it simply executes the version of the method whose parameters match the arguments used in the call.
- It is known as compile time polymorphism.

Example:

```
class Overload{
   void demo (int a){
     System.out.println ("a: " + a);
   }
   void demo (int a, int b) {
     System.out.println ("a and b: " + a + "," + b);
}
    double demo(double a) {
       System.out.println("double a: " + a);
       return a*a;
    }
}
class MethodOverloading{
    public static void main (String args []){
        Overload Obj = new Overload();
        double result;
        Obj .demo(10);
        Obj .demo(10, 20);
        result = Obj .demo(5.5);
        System.out.println("O/P : " + result);
    }
}
```

Method Overriding

In method overriding, **Child class (sub class)** has the same method as of **base class (super class)**. In such cases child class overrides the parent class method without even touching the source code of the base class.

Example

```
public class BaseClass{
   public void methodToOverride()                  //Base class method
   {
       System.out.println ("I'm the method of BaseClass");
   }
}
public class DerivedClass extends BaseClass
{
   public void methodToOverride()                //Derived Class method
   {
       System.out.println ("I'm the method of DerivedClass");
   }
}

public class TestMethod
{
```

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

     BaseClass obj1 = new BaseClass();       // BaseClass reference and object
     BaseClass obj2 = new DerivedClass();  // BaseClass reference but DerivedClass object
     obj1.methodToOverride();              // Calls the method from BaseClass class
     obj2.methodToOverride();            //Calls the method from DerivedClass class

   }
}
```

Important Points about Method Overriding

- applies only to inherited methods
- Abstract methods must be overridden
- Static and final methods cannot be overridden
- Constructors cannot be overridden
- It is also known as Runtime polymorphism.

Q. Encapsulation

Encapsulation is also known as "**data Hiding**".

- The whole idea behind encapsulation is to hide the implementation details from users.
- If a data member is private it means it can only be accessed within the same class.
- No outside class can access private data member (variable) of other class.

Q. Static Binding

- The binding which can be resolved at compile time by compiler is known as static or early binding.
- All the static, private and final methods have always been bonded at **compile-time** .

Q. Dynamic Binding

Dynamic binding is the process to link the function call with function signature **at run-time** (during execution of the program).
Overriding is a perfect example of dynamic binding as in overriding both parent and child classes have same method. Thus while calling the overridden method, the compiler gets confused between parent and child class method (since both the methods have same name).

Q. Difference between Procedure Oriented and Object Oriented Approach

	Procedure Oriented Programming	Object Oriented Programming
Divided Into	In POP, program is divided into small parts called **functions.**	In OOP, program is divided into parts called **objects.**
Importance	In POP, Importance is **not given to data** but to functions as well as **sequence** of actions to be done.	In OOP, Importance is **given to the data** rather than procedures or functions because it works as a real world.
Data Moving	In POP data can **move freely** from function to function in the system.	In OOP, **objects can move** and communicate with each other through member functions.
Approach	POP follows Top Down approach.	OOP follows Bottom Up approach.
Data Access	In POP, most functions use global data for sharing that can be accessed freely from function to function in the system.	In OOP, data can not move easily from function to function, it can be kept public or private so we can control the access of data.
Examples	C, VB, FORTRAN, Pascal.	C++, JAVA, VB.NET, C#.NET.

Q. Difference between object and class

S. No	Object	Class
1	Object is an **instance** of a class.	Class is a **blueprint or template** from which objects are created. It is known as **object factory.**
2	Object is a **real world entity** such as pen, laptop, mobile, bed, chair etc.	Class is a group of **similar objects.**
3	Object is a **physical** entity.	Class is a logical **entity.**
4	Object is created through **new keyword** mainly e.g. Student s1=new Student();	Class is declared using **class keyword** e.g. class Student{}
5	**Many** Objects can be created as per requirement.	Class is declared **once**.
6	Object **allocates memory** when it is created**.**	Class **does not allocate** memory when it is created.

Q. Benefits of Object Oriented Programming.

- It implements real life scenario.
- **Reusability:** In OOP's programs functions and modules that are written by a user can be reused by other users without any modification.
- **Inheritance:** Through this we can eliminate redundant code and extend the use of existing classes.
- **Data Hiding:** The programmer can hide the data and functions in a class from other classes. It helps the programmer to build the secure programs.
- **Reduced complexity of a problem:** The given problem can be viewed as a collection of different objects. Each object is responsible for a specific task. The problem is solved by interfacing the objects. This technique reduces the complexity of the program design.
- **Easy to Maintain and Upgrade:** OOP makes it easy to maintain and modify existing code as new objects can be created with small differences to existing ones.
- **Modifiability:** it is easy to make minor changes in the data representation or the procedures in an OO program. Changes inside a class do not affect any other part of a Program.

Q. Disadvantages Of Object Oriented Programming.

Size: Object Oriented programs are much larger than other programs.

Effort: Object Oriented programs require a lot of work to create. Specifically, a great deal of planning goes into an object oriented program well before a single piece of code is ever written.

Speed: Object Oriented programs are slower than other programs, partially because of their size. Other aspects of Object Oriented Programs also demand more system resources, thus slowing the program down.

Important Points

- Real-world objects contain **state** and **behavior**.
- A software object's state is stored in **fields**.
- A software object's behavior is exposed through **methods**.
- Hiding internal data from the outside world, and accessing it only through publicly exposed methods is known as data **encapsulation**.
- A blueprint for a software object is called a **class**.
- Common behavior can be defined in a **superclass** and inherited into a **subclass** using the **extends** keyword.
- A collection of methods with no implementation is called an **interface**.
- A namespace that organizes classes and interfaces by functionality is called a **package**.
- The term API stands for **Application Programming Interface**.