**Java**

**Java is an interpreted and complied language.**

**Data Types:** There are two types of data types.

* Primitive is also will use the name is primary data type.
  + Numeric
    - Integer
      * Byte : 1 byte
      * Short: 2 byte
      * Int : 4 byte
      * Long : 8 byte
    - Floating point
      * Float : 4 byte
      * Double: 8 byte
  + Non – numeric
    - Character – In memory 2 byte.
    - Boolean only in java introduced. (true/false) – 1 bit.
* Non primitive: - User defined data types.
  + Classes
  + Arrays
  + Interface

**JVM:** Java virtual machine. It’s like software will interpret in class. It will different from different machine to machine. Ex- Linux or window machine.

* Java file Compile byte code is not depended on machine.
* Byte Code Interpreter machine code.

**Public static void main (String args[]) in java.**

Public static void main(String args[])

Public – Access specified – Main method JVM will call main method that’s why main method should be public.

Static – If you define static method you need not to be define object in that class.

**Objects**

* Mapping real time objects / entities into
* Real time Objects

Objects

Data Behavior

Variable Method

* Abstract method means declaration.
* Abstraction is the process of hiding the implementation and showing the functionality to end users.

Entities | Variable

Ex: Car | Maruti Car

**Interface**

* Only declaration of method
* Signature is same but implementation is different

Remote Philips  
 on LG  
 off VI  
Set timer

* Known by one object and implemented by others.

**Encapsulation**

* Binding variable (data) and methods (behavior) in a single unit.
* Encapsulation in java is a process of wrapping code and data together into a single unit.

Advantage :- By providing only setter or getter method.

Ex: Accounts  
 private pin\_number;  
 private account\_type;  
 String name;  
 Number account\_namber;

withdrawal();  
 deposit();  
 fundTransfer();

**Data hiding -** Access modifier

* Private: - We can access within a class only.
* Public: - We can access from anywhere.
* Default: - If we can’t define anything from value that should be default.

**Class**

**Note: -** You cannot define private, protected in outer class. You can define only public or only class. If cannot define anything that should be default.

**Getting input from user**

First you need to import from classes.

* import java.util.Scanner; - The Scanner is the class to get input from user.

**import** java.util.Scanner;

**public** **class** TestScanner {

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

**int** no, i = 2;

Scanner obj = **new** Scanner(System.***in***);

System.***out***.println("Enter ur Number");

no = obj.nextInt();

System.***out***.println(no\*no);

}

}

Ex:

**Class**

A class will define the objects and state behavior will define. Without class you can’t define the objects and field declaration and methods as well.

* Class name should start with capital word.
* If you want to define an objects it’s define on a class in that scenario to create an object using with new keyword and if you want to instantiate to that object to use new keyword. (obj = new obj())
* Object will store in a memory.
* When you define a new keyword in a class that should be default constructor.

**Constructor**

* Constructor is a special type of method is same like a class name.
* Class name and constructor name should be the same and there is no any return type.

Ex: Class Student(){

Student(){}

}

* The main purpose of constructor to initialize the objects means to initialize the instance variable. When you instantiate the object at that time constructor will invoke.
* If you define or may not be define once you define a class at the time complier will give you default constractor.

Parameterize contractor

Default Constructor

Constructor

**Static keywords and Static Methods**

* If you define a static keyword before to ur variable that field should be class variable.
* If you define static keyword before to ur method that function should be class methods.
* If any changes in the static variable you ll get changed value in obj1 or obj2.
* If you define static keyword there is no need to define an object.
* Ex: classname.variableName = value;

Ex:

**class** Test{

**int** a; // instance variable;

**static** **int** *b*; // class variables;

}

**Method overloading**:

* In a class when you define multiple methods on the same class but arguments is different that is method overloading like void add(int x, int y), void add(int x, int y, int z).

**Method overriding**:

* Method overriding is only use in heritance like super class and child class method and arguments and return type should be match but implementation is different. Super class like should say overridden and subclass should be say overriding. Like void move(), move()
* If method is private will not override because of inherit will not possible.
* Constructor will not override.
* Static method will not override.
* Final will not override.

**Abstract:**

* Abstract is a keyword, you will use class or in method.
* You will create an object for subclass but not create an object for an Abstract class.
* It will indicate that method is not complete. It will complete by subclass.
* It will not compulsory there is in abstract method.
* Header will be there but body will not define. Means definition will not there.
* If you will define abstract method is compulsory to define abstract class.
* You can abstract in subclass as well.

**Abstract.java**

**abstract** **class** x{

**abstract** **void** disc();

}

**class** y **extends** x{

**void** disc(){

System.***out***.println("Abtstract Class");

}

}

**AbstractMain.class**

**class** AbstractMain {

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

// **TODO** Auto-generated method stub

y obj = **new** y();

obj.disc();

}

}

**Interface:**

An interface in java is a blue print of a class. It has static constants and abstract method. It is used to achieve abstraction. By interface we can support the functionality of multiple inheritances.

* We cannot define object for an interface
* In interface methods will use like abstract class
* All variables are constants.
* Only public keyword will use in interface. Field and method both should be public.
* Interface method will use in class using with implements.
* One method to another method will use only extends.

**Packages**: It’s just like a grouping ex: classes, interfaces etc. Packages will work like a container. If your project is big better to use packages.

Types of packages :

1. Built in packages : Java is predefined classes, methods in java. Java.io.classname, java.lang.classname, java.util.sacanner… java is package and util is subpackage. Scanner is class.
   1. Java is a packages – AWT is a subpackage – color is a class.
2. User defined packages : -
   1. Package name should be unique.
   2. LowerCase.
3. Reusability
4. Access control –We can use only package but can’t use outside class.
5. Accessing Packages – Fully qualified class name and import statement. You can access packages as a obj.