UNIT-1

Object Oriented Programming(OOPS): Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which can contain data and code: data in the form of fields (known as attributes or properties), and code, in the form of behaviour (known as methods).

Characteristic or Features of Object Oriented Programming:

Objects:

- Objects are the basic run-time entities in an object oriented system.They may represent a person,a place or any item that the program has to handle.
- An object is an instance of a class.
- Example: Student, table, teacher, car, fan, chair etc..

Object has three properties:

1. Identity
2. Attribute
3. Behaviour

Ex- In student object

Roll no is identity

Name is attribute

getResult() is behaviour

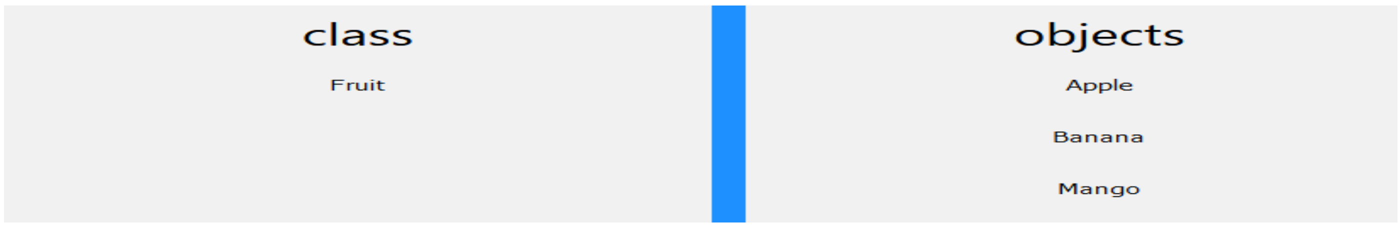
Creating Objects:

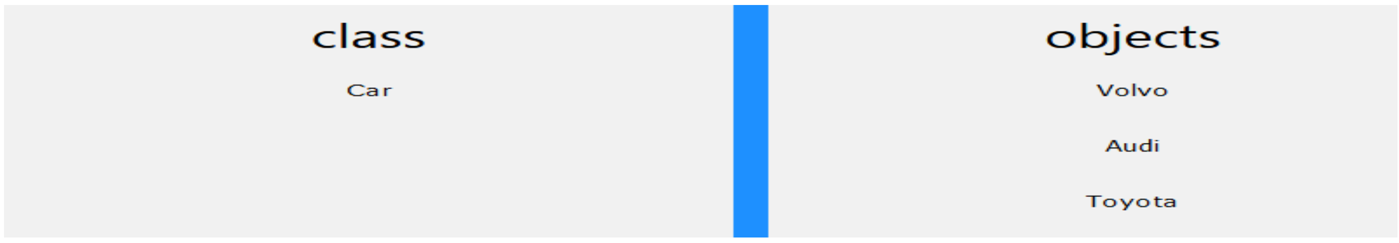
Example:

```
Student S1= new Student();
```

Class:-

- Class is the blue-print of class. Or it is the template of object.
- The entire set of data and code of an object can be made a user-defined data type with the help of a class.
- The objects are variables of the type class.
- Once a class has been defined, we can create any number of objects belonging to that class.





So, a class is a template for objects, and an object is an instance of a class.

When the individual objects are created, they inherit all the variables and functions from the class.

__Declaring Classes:__

We declare classes using the following syntax:-

```
class < Class Name >
{
data members…
member functions…
}
```

Here class is a keyword.

__Data Hiding:__

Data hiding is hiding the details of internal data members of an object.

- Data hiding is also known as Information hiding.
- Sometimes Data Hiding includes Encapsulation. Thus Data Hiding is heavily related to Abstraction and Encapsulation.
- Data Hiding is the one most important OOP mechanism. Which is hide the details of the class from outside of the class.
- The Class used by only a limited set of variables and functions, others are hidden by the class.
- We use private keyword to hide data.
- Data hiding provide security to the data.

Ex:

```
 class Student{
private int rollno;//data hiding using private access specifier
String name;
}
```

Abstraction:

Abstraction is the concept of object-oriented programming that "shows" only essential attributes and "hides" unnecessary information. The main purpose of abstraction is hiding the unnecessary details from the users. It helps in reducing programming complexity and efforts. It is one of the most important concepts of OOPs.

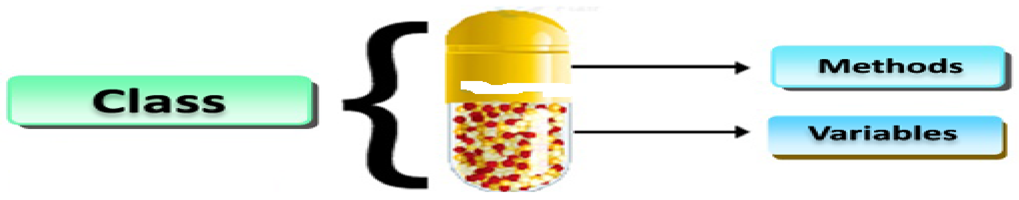
Consider a real-life example of a man driving a car. The man only knows that pressing the accelerators will increase the speed of car or applying brakes will stop the car, but he does not know about how on pressing the accelerator the speed is actually increasing, he does not know about the inner mechanism of the car or the implementation of the accelerator, brakes etc in the car. This is what abstraction is.

In java, abstraction is achieved by <u>interfaces</u> and <u>abstract classes</u>. We can achieve 100% abstraction using interfaces.

Encapsulation :

Encapsulation represents the information of variables (attributes) in terms of data and, methods (functions) and its operations in terms of purpose.

- Encapsulation is the process of combining data and function into a single unit called class.
- Encapsulation is a powerful feature that leads to information hiding and abstract data type.
- They encapsulate all the essential properties of the object that are to be created.
- Using the method of encapsulation the programmer cannot access the class directly.



Ex: class Student{

```
int rollno;
String name;
void getName(){
}
}
```

Advantages of Encapsulation:

- The main advantage of using of encapsulation is to secure the data from other methods, when we make a data private then these data only use within the class, but these data not accessible outside the class.
- The major benefit of data encapsulation is the security of the data. It protects the data from unauthorized users that we inferred from the above stated real-real problem.
- We can apply the concept of data encapsulation in the marketing and finance sector where there is a high demand for security and restricted access of data to various departments.
- Encapsulation helps us in binding the data(instance variables) and the member functions(that work on the instance variables) of a class.
- Encapsulation is also useful in hiding the data(instance variables) of a class from an illegal direct access.
- Encapsulation also helps us to make a flexible code which is easy to change and maintain.

Dis-Advantage of Encapsulation

You can't access private date outside the class.

Inheritance :

- When the properties and the methods of the parent class are accessed by the child class, Then it called inheritance. The child class can inherit the parent variables and methods and can also give own method implementation.

In the inheritance the class which is give data members and methods is known as base or super or parent class.

The class which is taking the data members and methods is known as sub or derived or child class.

For code Re-usability is provided by inhritance

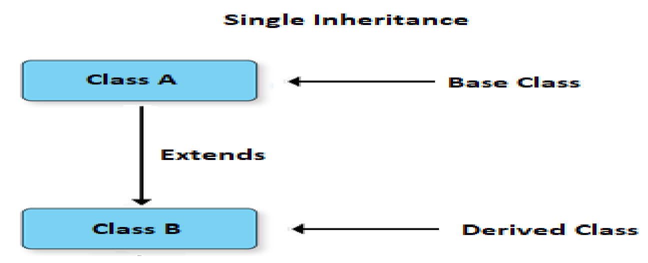
Types of Inheritance

Based on number of ways inheriting the feature of base class into derived class we have five types of inheritance; they are:

- Single inheritance
- Multiple inheritance
- Hierarchical inheritance
- Multilevel inheritance
- Hybrid inheritance

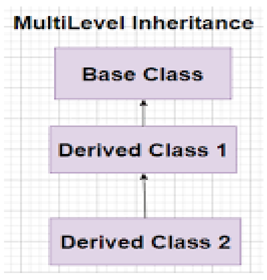
Single inheritance

In single inheritance there exists single base class and single derived class.



Multilevel inheritances

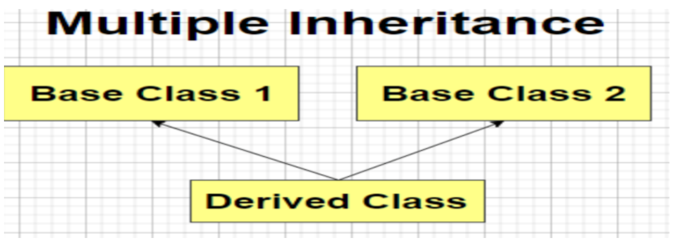
In Multilevel inheritances there exists single base class, single derived class and multiple intermediate base classes.



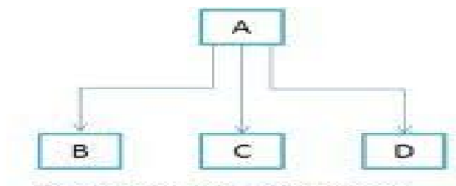
Multiple inheritance

In multiple inheritance there exist multiple classes and singel derived class.

The concept of multiple inheritance is not supported in java through concept of classes but it can be supported through the concept of interface.

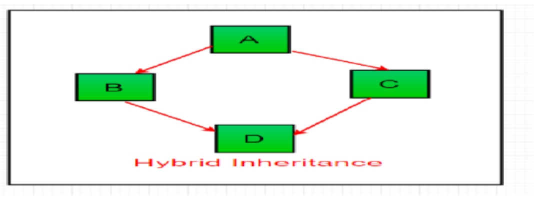


Hierarchical Inheritance: When more than one classes inherit a same class then this is called *hierarchical inheritance*. For example class B, C and D inherit a same class A.
L



(c) Hierarchical Inheritance

Hybrid inheritance: In the combination if one of the combination is multiple inheritance then the inherited combination is not supported by java through the classes concept but it can be supported through the concept of interface.



Advantage of inheritance

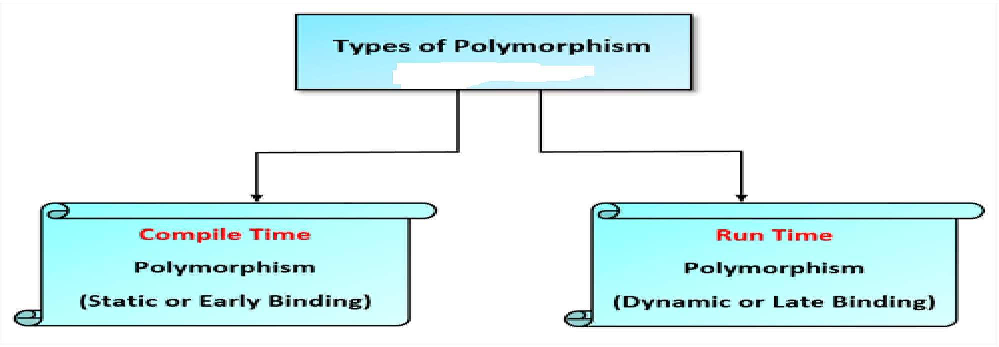
If we develop any application using concept of Inheritance than that application have following advantages,

- Application development time is less.
- Application take less memory.
- Application execution time is less.
- Application performance is enhance (improved).
- Redundancy (repetition) of the code is reduced or minimized so that we get consistence results and less storage cost.

Polymorphism

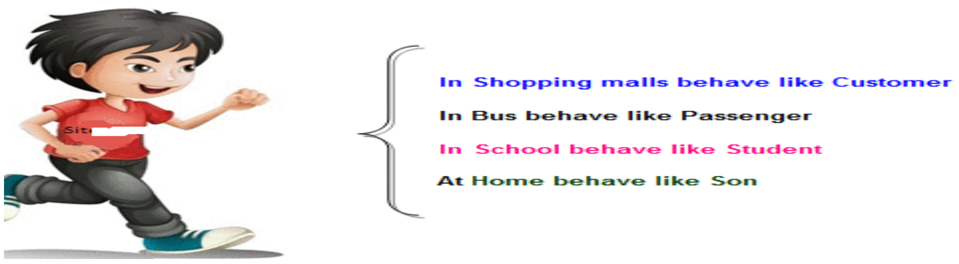
The process of representing one form in multiple forms is known as **Polymorphism**.

Polymorphism is derived from 2 greek words: **poly** and **morphs**. The word "poly" means many and "morphs" means forms. So polymorphism means many forms.



Real life example of polymorphism

Suppose if you are in class room that time you behave like a student, when you are in market at that time you behave like a customer, when you at your home at that time you behave like a son or daughter, Here one person present in different-different behaviors.


In Shopping malls behave like Customer
In Bus behave like Passenger
In School behave like Student
At Home behave like Son

How to achieve Polymorphism in Java ?

In java programming the Polymorphism principal is implemented with method overriding concept of java.

Polymorphism principal is divided into two sub principal they are:

- Static Binding or Compile time polymorphism
- Dynamic Binding or Runtime polymorphism

Static polymorphism

- The process of binding the overloaded method within object at compile time is known as **Static polymorphism** due to static polymorphism utilization of resources (main memory space) is poor because for each and every overloaded method a memory space is created at compile time when it binds with an object.
- Compile time polymorphism or static polymorphism relates to method overloading.

Dynamic polymorphism

- In dynamic polymorphism method of the program binds with an object at runtime the advantage of dynamic polymorphism is allocating the memory space for the method (either for overloaded method or for override method) at run time.
- Run time polymorphism or dynamic polymorphism or dynamic binding relates to method

What is Java

Java is a high level, robust, secured and object-oriented programming language.

Characteristics or Features of Java

There is given many features of java. They are also known as java buzzwords. The Java Features given below are simple and easy to understand.

Simple
Object-Oriented
Platform independent
Secured
Robust
Portable
Dynamic
Compiled and Interpreted
High Performance
Multithreaded
Distributed

Simple

Java language is simple because:

syntax is based on C++ (so easier for programmers to learn it after C++).

removed many confusing and/or rarely-used features e.g., explicit pointers, operator overloading etc.

No need to remove objects because there is Automatic Garbage Collection in java.

Object-oriented

Object-oriented means we organize our software as a combination of different types of objects that incorporates both data and behaviour.

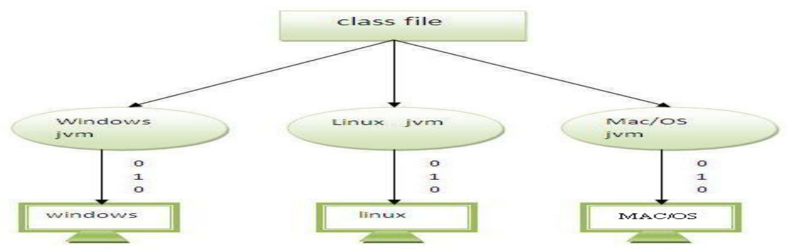
Basic concepts of OOPs are:

Object
Class
Data Hiding
Abstraction
Encapsulation
Inheritance
Polymorphism

Platform Independent

A platform is the hardware or software environment in which a program runs. There are two types of platforms software-based and hardware-based. Java provides software-based platform. The Java platform differs from most other platforms in the sense that it's a software-based platform that runs on top of other hardware-based platforms. It has two components:

1. Runtime Environment
2. API(Application Programming Interface)

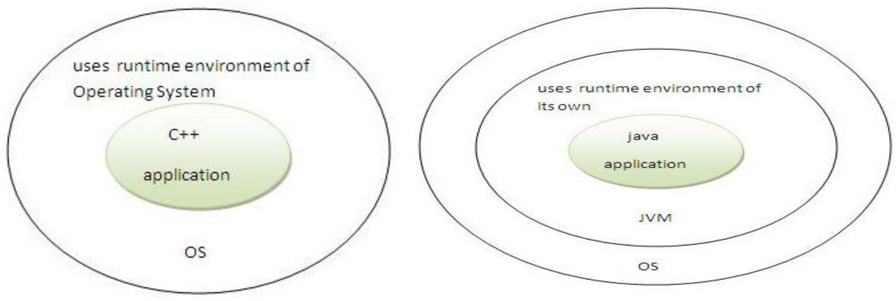


Java code can be run on multiple platforms e.g. Windows, Linux, Sun Solaris, Mac/OS etc. Java code is compiled by the compiler and converted into bytecode. This bytecode is a platform independent code because it can be run on multiple platforms i.e. Write Once and Run Anywhere(WORA).

<u>Secured</u>

Java is secured because:

No explicit pointer
Programs run inside virtual machine
sandbox.


uses runtime environment of Operating System
C++
application
OS
uses runtime environment of its own
java
application
JVM
OS

<u>Classloader</u>- adds security by separating the package for the classes of the local file system from those that are imported from network sources.
<u>Bytecode Verifier- </u>checks the code fragments for illegal code that can violate access right to objects.
Security Manager- determines what resources a class can access such as reading and writing to the local disk.

<u>Robust</u>

Robust simply means strong. Java uses strong memory management. There are lack of pointers that avoids security problem. There is automatic garbage collection in java. There is exception handling and type checking mechanism in java. All these points makes java robust.

<u>Portable</u>

We may carry the java bytecode to any platform. i.e it can be run in windows,linux,unix ,macOS etc. operating system.

<u>High-performance</u>

Java is faster than traditional interpretation since byte code is "close" to native code still somewhat slower than a compiled language (e.g., C++)

<u>Distributed</u>

We can create distributed applications in java. RMI and EJB are used for creating distributed applications. We may access files by calling the methods from any machine on the internet.

Multi-threaded

A thread is like a separate program, executing concurrently. We can write Java programs that deal with many tasks at once by defining multiple threads. The main advantage of multi-threading is that it shares the same memory. Threads are important for multi-media, Web applications etc.

Example of Creating hello java program:

```
class Simple{
   public static void main(String args[]){
    System.out.println("Hello Java");
   }
```

save this file as Simple.java

To compile:	javac Simple.java
To execute:	java Simple

Output: Hello Java

Understanding first java program:

class :keyword is used to declare a class in java.
public :keyword is an access modifier which represents visibility, it means it is visible to all.
static :is a keyword, if we declare any method as static, it is known as static method. The core advantage of static method is that there is no need to create object to invoke the static method. The main method is executed by the JVM, so it doesn't require to create object to invoke the main method. So it saves memory.
void :is the return type of the method, it means it doesn't return any value.
main :represents startup of the program.
String[] args :is used for command line argument.
System.out.println() :is used print statement. Where System is class and out is a static variable defined in System class and print() is a method defined inside the PrintStream class.

JVM (Java Virtual Machine) Architecture

JVM (Java Virtual Machine) is an abstract machine. It is a specification that provides runtime environment in which java bytecode can be executed.

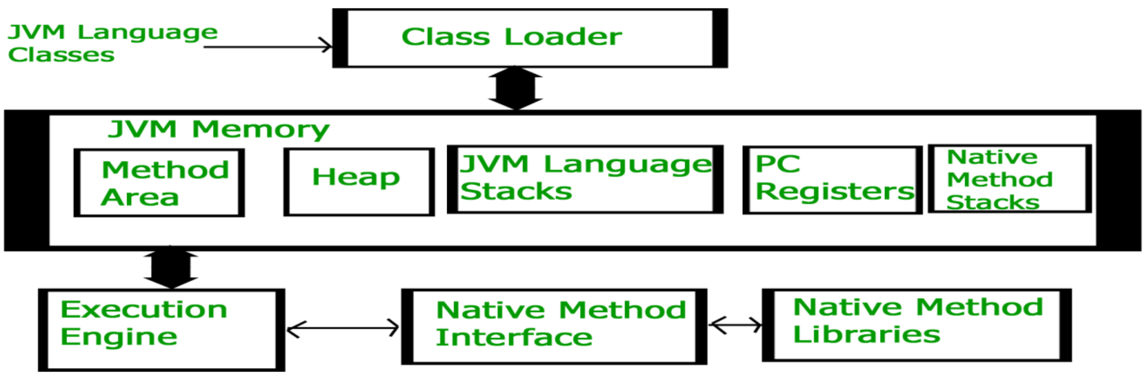
The JVM performs following operation:

- Loads code
- Verifies code
- Executes code
- Provides runtime environment

JVM provides definitions for the:

- Memory area
- Class file format
- Register set
- Garbage-collected heap
- Fatal error reporting etc.

JVM Architecture



1) Classloader

Classloader is a subsystem of JVM which is used to load class files. Whenever we run the java program, it is loaded first by the classloader. There are three built-in classloaders in Java.

1. Bootstrap ClassLoader: This is the first classloader which is the super class of Extension classloader. It loads the *rt.jar* file which contains all class files of Java Standard Edition like java.lang package classes, java.net package classes, java.util package classes, java.io package classes, java.sql package classes etc.
2. Extension ClassLoader: This is the child classloader of Bootstrap and parent classloader of System classloader. It loades the jar files located inside *$JAVA_HOME/jre/lib/ext* directory.
3. System/Application ClassLoader: This is the child classloader of Extension classloader. It loads the classfiles from classpath. By default, classpath is set to current directory. You can change the classpath using "-cp" or "-classpath" switch. It is also known as Application classloader.

2) Class(Method) Area

Class(Method) Area stores per-class structures such as the runtime constant pool, field and method data, the code for methods.

3) Heap

It is the runtime data area in which objects are allocated.

4) Stack

Java Stack stores frames. It holds local variables and partial results, and plays a part in method invocation and return.

Each thread has a private JVM stack, created at the same time as thread.

A new frame is created each time a method is invoked. A frame is destroyed when its method invocation completes.

5) Program Counter Register

PC (program counter) register contains the address of the Java virtual machine instruction currently being executed.

6) Native Method Stack

It contains all the native methods used in the application.

7) Execution Engine

It contains:

1. A virtual processor
2. Interpreter: Read bytecode stream then execute the instructions.
3. Just-In-Time(JIT) compiler: It is used to improve the performance. JIT compiles parts of the byte code that have similar functionality at the same time, and hence reduces the amount of time needed for compilation. Here, the term "compiler" refers to a translator from the instruction set of a Java virtual machine (JVM) to the instruction set of a specific CPU.

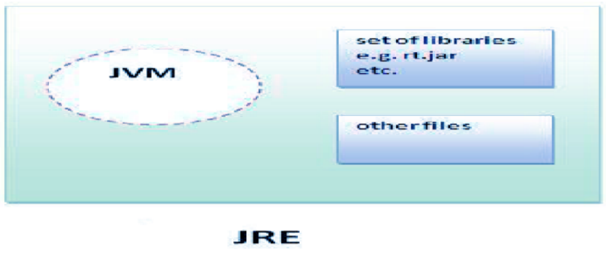
8) Java Native Interface

Java Native Interface (JNI) is a framework which provides an interface to communicate with another application written in another language like C, C++, Assembly etc. Java uses JNI framework to send output to the Console or interact with OS libraries.

JRE:

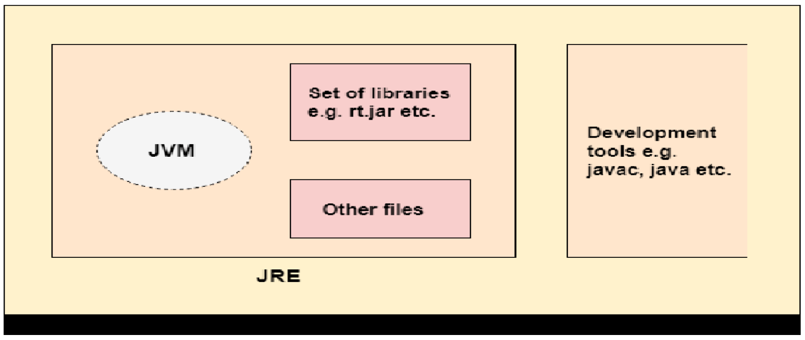
JRE is an acronym for Java Runtime Environment. It is also written as Java RTE. The Java Runtime Environment is a set of software tools which are used for developing Java applications. It is used to provide the runtime environment. It is the implementation of JVM. It physically exists. It contains a set of libraries + other files that JVM uses at runtime.

The implementation of JVM is also actively released by other companies besides Sun Micro Systems.



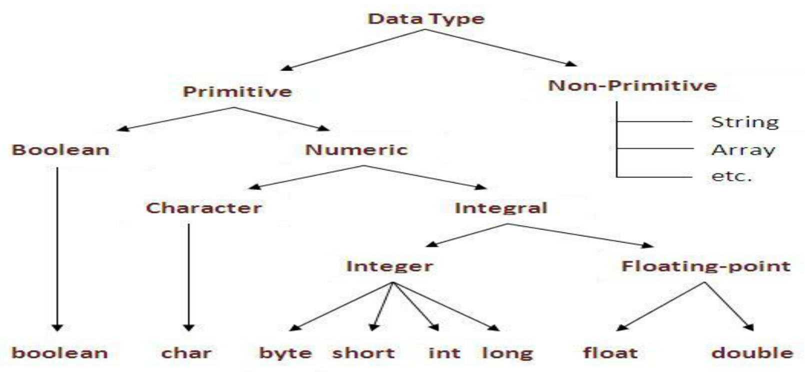
JDK

JDK is an acronym for Java Development Kit. The Java Development Kit (JDK) is a software development environment which is used to develop Java applications . It physically exists. It contains JRE + development tools.


Set of libraries e.g. rt.jar etc.
JVM
Other files
JRE
Development tools e.g. javac, java etc.

Data Types in Java:

Data types specify the different sizes and values that can be stored in the variable. There are two types of data types in Java:



Data Type	Default Value	Size
boolean	False	1 bit
Char	'\u0000'	2 byte
Byte	0	1 byte
Short	0	2 byte
Int	0	4 byte
Long	0L	8 byte
Float	0.0f	4 byte
Double	0.0d	8 byte

Variables in java:

Variable is an identifier which holds data or another one variable is an identifier whose value can be changed at the execution time of program. Variable is an identifier which can be used to identify input data in a program.

Rules to declare a Variable

- Every variable name should start with either alphabets or underscore (_) or dollar ($) symbol.
- No space are allowed in the variable declarations.
- Except underscore (_) no special symbol are allowed in the middle of variable declaration
- Variable name always should exist in the left hand side of assignment operators.
- Maximum length of variable is 64 characters.
- No keywords should access variable name

Types of Variables

There are three types of variables in Java:

- local variable
- instance variable
- static variable

1) Local Variable

A variable declared inside the body of the method is called local variable. You can use this variable only within that method and the other methods in the class aren't even aware that the variable exists.

A local variable cannot be defined with "static" keyword.

2) Instance Variable

A variable declared inside the class but outside the body of the method, is called instance variable. It is not declared as static.

It is called instance variable because its value is instance specific and is not shared among instances.

3) Static variable

A variable which is declared as static is called static variable. It cannot be local. You can create a single copy of static variable and share among all the instances of the class. Memory allocation for static variable happens only once when the class is loaded in the memory.

```
class A{
int data=50;//instance variable
static int m=100;//static variable
void method(){
int n=90;//local variable
}
}
```

Java Naming conventions

Java **naming convention** is a rule to follow as you decide what to name your identifiers such as class, package, variable, constant, method etc.
But, it is not forced to follow. So, it is known as convention not rule.
All the classes, interfaces, packages, methods and fields of java programming language are given according to java naming convention.

Advantage of naming conventions in java

By using standard Java naming conventions, you make your code easier to read for yourself and for other programmers. Readability of Java program is very important. It indicates that **less time** is spent to figure out what the code does.

Name	Convention
class name	should start with uppercase letter and be a noun e.g. String, Color, Button, System, Thread etc.
interface name	should start with uppercase letter and be an adjective e.g. Runnable, Remote, ActionListener etc.
method name	should start with lowercase letter and be a verb e.g. actionPerformed(), main(), print(), println() etc.
variable name	should start with lowercase letter e.g. firstName, orderNumber etc.
package	should be in lowercase letter e.g. java, lang, sql, util etc.

name	
constants name	should be in uppercase letter. e.g. RED, YELLOW, MAX_PRIORITY etc.

<u>CamelCase in java naming conventions:</u>
Java follows camelcase syntax for naming the class, interface, method and variable. If name is combined with two words, second word will start with uppercase letter always e.g. method names : actionPerformed(), firstName
Class names: ActionEvent, ActionListener

<u>Java Tokens:</u>
The smallest units of Java language are called Java tokens. There are five tokens available in java

1. Keywords 2.Identifiers 3.Literals 4. Separators 5.Operators

1.Keywords:
Java Keywords also called a reserved word. Keywords are Java reserves words for its own use. These have built-in meanings that cannot change. There are 49 reserved keywords currently defined in the Java language and they are shown in the below table.

abstract	Double	Int	Switch
assert	Else	interface	Synchronized
boolean	Extends	long	This
break	False	native	Throw
byte	Final	new	Transient
case	Finally	package	True
catch	Float	private	Try
char	For	protected	Void
class	Goto	public	Volatile
const	If	return	While
continue	implements	short	
default	Import	static	
Do	instanceof	super	

The keywords const and goto are reserved but not used.
<u>Identifiers:</u>
Identifiers are defined by programmer.
Java developers follow naming conventions for writing identifiers.
Ex:-Names of all methods and variables, classes, interfaces, packages, constant

Example: average
sum

<u>Literals:--</u>
=>Literals in Java are a sequence of characters (digits, letters, and other characters)

that represent values to be stored in variables.
=>Java language specifies five major types of literals. Literals can be any number, text, or other information that represents a value. They are:
Integer literals
Ex:10,20
Floating literals
Ex:12.33
Character literals
Ex: ‘c’
String literals
Ex: “Hello”
Boolean literals
Ex: true and false
<u>Separators:---</u>
Separators helps to define the structure of a program. The separators are parentheses, ( ), braces, { }, the period, ., and the semicolon, ;.

<u>Operators:---</u>
Arithmetic , relational, bitwise, logic , Assignment and conditional operator.

<u>JAVA TYPE CASTING or TYPE CONVERSION:</u>
Conversion of one data type to another data type is called type casting.
There are two types of type casting:
1.implicit type casting or widening or upcasting or automatic
2. Explicit type casting or narrowing or downcasting.

<u>1. Implicit type casting:--</u> Widening conversion takes place when two data types are automatically converted. This happens when:

- The two data types are compatible.
- When we assign value of a smaller data type to a bigger data type.

```
class Test
{
   public static void main(String[]
args)
   {
      int i = 100;

      //automatic type conversion
      long l = i;

      //automatic type conversion
      float f = l;
      System.out.println("Int value "+i);
      System.out.println("Long value
"+l);
      System.out.println("Float value
"+f);
   }
}
```
Output:

```
Int value 100
Long value 100
Float value 100.0
```

2.Narrowing or Explicit Type Castinng / Conversion:

If we want to assign a value of larger data type to a smaller data type we perform explicit type casting or narrowing.

- This is useful for incompatible data types where automatic conversion cannot be done.
- Here, target-type specifies the desired type to convert the specified value to.

```
class Test
{
   public static void main(String[] args)
   {
      double d = 100.04;
      //explicit type casting
      long l = (long)d;
     //explicit type casting
      int i = (int)l;
      System.out.println("Double value "+d);
     //fractional part lost
      System.out.println("Long value "+l);
      //fractional part lost
      System.out.println("Int value "+i);
   }
}
```

Output:

```
Double value 100.04
Long value 100
Int value 100
```