**Daily Java Learning**

Date : 31-12-2022 Time : 7:30 pm

**What is Java ?**

* Java is a high-level, object-oriented, secured programming language developed by sun networks in the year 1995.
* Java is also a PLATFORM

Any hardware or software environment in which a program runs, is known as a platform. Since Java has a runtime environment (JRE) and API, it is called a platform.

* James Gosling is known as father of java

**Applications of JAVA**

* According to sun , more than 3 billion devices run **JAVA**
* Desktop Applications ( e.g. acrobat reader, media player and etc. )
* Web Applications (e.g. irctc.co.in and etc. )
* Enterprise Applications (e.g. banking applications)
* Mobile Applications(e.g. Spotify , twitter and etc. )

**Types of JAVA Applications**

There are 4 platforms or editions of **JAVA**

**JAVA SE (** java standard Edition **):**

It is java programming platform , it includes java programming API’s such a java.lang , java.io, java.net ,java.util, java.sql ,java.math and etc. it include the core topics like object oriented Programming language ,string ,regex, Exception, inner class, Multi threading ,I/O streams , Networking , AWT(abstract window toolkit),collection .

**JAVA EE (**java enterprise Edition**):**

It is enterprise platform ,mainly used to develop web and enterprise applications . it is built on the top of **JAVA SE** platform . it includes the topics like Servlets , JSP (**Jakarta Server Pages** ) ,web services , Enterprise java Bean (EJB), Java persistence API etc.

**JAVA ME (**java micro Edition**) :**

The Java ME stands for **Java Micro Edition**. It is a development and deployment platform of portable code for embedded and mobile devices (sensors, gateways, mobile phones, printers, TV set-top boxes). It is based on **object-oriented Java**. The Java ME has a robust user interface, great security, built-in network protocols, and support for applications that can be downloaded dynamically. Applications which are developed on Java ME are portable and can run across various devices and can also leverage the native capabilities of the device.

* It is a micro platform that is dedicated to mobile applications.

**JAVA FX (**special effects in JAVA language**):**

It is used to develop rich internet applications. It uses a lightweight user interface API

**Simple HelloWorld program in JAVA**

**public** **class** HelloWorld {

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

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

System.***out***.println("Hello World");

}

}

**Public static void main (String[] args)**

**Public**

It is an access specifier that means main() method is accessible globally available . This is necessary because this method is being called by the Java Runtime Environment (JRE) which is not located in your current class. It is important to note that if you make main() method non-public then it's not allowed to be executed by any program, there are some access restrictions applied.

**static**

The main() method in Java must be static, because they can then be invoked by the runtime engine without having to instantiate any objects then the code in the body of main() method will do the rest. The main() method should be static because otherwise there would be ambiguity : which constructor should be called? If the main() is allowed to be non-static, then while calling the main() method JVM has to instantiate its class. While instantiating it has to call the constructor of that class, There will be ambiguity if the constructor of that class takes an argument.

**void**

Java is a platform independent programming language and if it will return some value then the value may mean different things to different platforms. The "void" is a return type i.e it does not return any value. When the main() method terminates, the java program terminates too. Hence, it doesn't make any sense to return from main() method. If the main() method spawns new threads, then these threads can keep program running. The return type of main doesn't make much sense at this point.

**Main()**

It's just the name of method or a function name. This name is fixed and as it's called by the JVM as entry point for an application. It's not a keyword.

**String args[]**

These are the arguments of type String that your Java application accepts when you run it. Java main() method accepts only string type of argument and stores it in a string array. It is a collection of Strings , separated by a space, which can be typed into the program on the terminal.

As the Java language allows the brackets [] to be positioned after the type or the variable (the first is generally preferred)

* **System.out.println()** is used to print statement. Here, System is a class, out is an object of the PrintStream class, println() is a method of the PrintStream class. We will discuss the internal working of [System.out.println()](https://www.javatpoint.com/system-out-println-in-java) statement in the coming section.

Date : 01-01-2023 Time : 1:00 pm

* Java is a Both complied and interpreted language

1st the java code complied by the complier and converted to the byte code

.java file -------------------> .class file

The byte code runs on java virtual machine (**JVM**) which is an interpreter

**Features of JAVA**

Features of java is also known as buzzwords.

1. **Simple** :

* Java is a simple programming language ,it is easy to learn , clean and easy to understand the code
* It dose not have a concept of pointers
* There is an Automatic garbage collection in Java

1. **Object Oriented Programming** :

* Java follows the oops as it include classes , objects and instances

1. **Platform independent** :

* Java is a write once and run anywhere language
* The java code is compiled by the compiler and converted to the byte code .
* This byte code can run on any platform

1. **Secured**:

* Java is best known for the security, with java we can develop free virus systems. Java is secured because
* No explicit pointers
* Java program run inside the virtual machine sandbox

1. **Robust**:

* It uses strong memory management.
* No pointers
* Java provides Automatic garbage collection
* There is an Exception handling & type checking mechanism in Java

1. **Architecture-neutral**:

* Java is architecture neutral because there is no implementation dependent features , for Examples the size of the primitive datatypes is fixed
* In C programming, int data type occupies 2 bytes of memory for 32-bit architecture and 4 bytes of memory for 64-bit architecture. However, it occupies 4 bytes of memory for both 32 and 64-bit architectures in Java.

1. Portable

* Java is portable because it facilitates you to carry the Java bytecode to any platform. It doesn't require any implementation.

1. Refer this for more <https://www.javatpoint.com/features-of-java>

To run and Compile the Java Code in command promt

**To compile**

Javac <filename.java> e.g. javac Simple.java

**To Execute**

Java <filename> e.g. java Simple

Date : 02-01-2023 Time : 11:00 am

Differences between JDK JRE and JVM

**Java development Kit (JDK) :**

This kit provide the environment to build and run the java code or (deploy and execute the java code ) . JDK includes

* Development tools(to develop JAVA code )
* JRE(java runtime environment) to run the java code

**Java Runtime Environment(JRE):**

Its an installation package , it is used to only run the java programs (**no development of java code** )

It is used by the end-users ,who only want to run the java code .

**Java virtual Machine(JVM):**

JVM is included in both JRE and JDK , what ever the java code run using the JRE or JDK goes in to JVM . JVM is responsible for executing the code line by line , hence it is called the interpreter.

JVM Loads the code 🡪 verifies the code 🡪 execute the code 🡪

Provide the runtime environment

**Variables In JAVA**

There are three types of variables in java

* Static variables : variable declared as static is called static variable , memory allocated only once.
* Local variables: this variables are declared inside the method of the class and accessible only method of the class
* Instance variable :this variable are declared outside the method and inside the class .

Example :

Class variabledemo {

Public static void main(String [] args){

Static int m =100; //this is static variable;

Int a =5; // this is instance variable

Int b=10; // this is instance variable

}

}

Class varaiabledemo{

Void method(){

Int a =10 //this is local variable

}

}

**DATA TYPES IN JAVA**



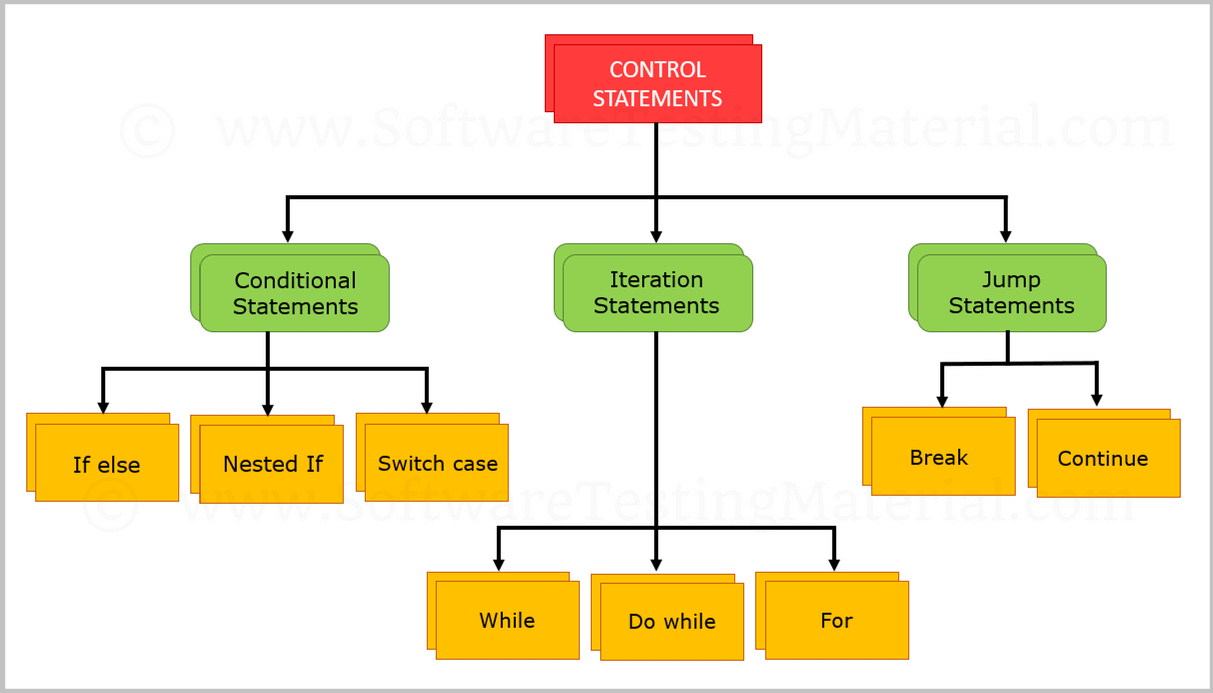
**Operators in JAVA**

****

**Keywords in JAVA**

1. [**abstract**](https://www.javatpoint.com/abstract-keyword-in-java)**:** Java abstract keyword is used to declare an abstract class. An abstract class can provide the implementation of the interface. It can have abstract and non-abstract methods.
2. [**boolean:**](https://www.javatpoint.com/boolean-keyword-in-java) Java boolean keyword is used to declare a variable as a boolean type. It can hold True and False values only.
3. [**break**](https://www.javatpoint.com/java-break)**:** Java break keyword is used to break the loop or switch statement. It breaks the current flow of the program at specified conditions.
4. [**byte**](https://www.javatpoint.com/byte-keyword-in-java)**:** Java byte keyword is used to declare a variable that can hold 8-bit data values.
5. [**case**](https://www.javatpoint.com/case-keyword-in-java)**:** Java case keyword is used with the switch statements to mark blocks of text.
6. [**catch**](https://www.javatpoint.com/try-catch-block)**:** Java catch keyword is used to catch the exceptions generated by try statements. It must be used after the try block only.
7. [**char**](https://www.javatpoint.com/char-keyword-in-java)**:** Java char keyword is used to declare a variable that can hold unsigned 16-bit Unicode characters
8. [**class**](https://www.javatpoint.com/class-keyword-in-java)**:** Java class keyword is used to declare a class.
9. [**continue**](https://www.javatpoint.com/java-continue)**:** Java continue keyword is used to continue the loop. It continues the current flow of the program and skips the remaining code at the specified condition.
10. [**default**](https://www.javatpoint.com/default-keyword-in-java)**:** Java default keyword is used to specify the default block of code in a switch statement.
11. [**do**](https://www.javatpoint.com/java-do-while-loop)**:** Java do keyword is used in the control statement to declare a loop. It can iterate a part of the program several times.
12. [**double**](https://www.javatpoint.com/double-keyword-in-java)**:** Java double keyword is used to declare a variable that can hold 64-bit floating-point number.
13. [**else**](https://www.javatpoint.com/java-if-else)**:** Java else keyword is used to indicate the alternative branches in an if statement.
14. [**enum**](https://www.javatpoint.com/enum-in-java)**:** Java enum keyword is used to define a fixed set of constants. Enum constructors are always private or default.
15. [**extends**](https://www.javatpoint.com/inheritance-in-java)**:** Java extends keyword is used to indicate that a class is derived from another class or interface.
16. [**final**](https://www.javatpoint.com/final-keyword)**:** Java final keyword is used to indicate that a variable holds a constant value. It is used with a variable. It is used to restrict the user from updating the value of the variable.
17. [**finally**](https://www.javatpoint.com/finally-block-in-exception-handling)**:** Java finally keyword indicates a block of code in a try-catch structure. This block is always executed whether an exception is handled or not.
18. [**float**](https://www.javatpoint.com/float-keyword-in-java)**:** Java float keyword is used to declare a variable that can hold a 32-bit floating-point number.
19. [**for**](https://www.javatpoint.com/java-for-loop)**:** Java for keyword is used to start a for loop. It is used to execute a set of instructions/functions repeatedly when some condition becomes true. If the number of iteration is fixed, it is recommended to use for loop.
20. [**if**](https://www.javatpoint.com/java-if-else)**:** Java if keyword tests the condition. It executes the if block if the condition is true.
21. [**implements**](https://www.javatpoint.com/interface-in-java)**:** Java implements keyword is used to implement an interface.
22. [**import**](https://www.javatpoint.com/package)**:** Java import keyword makes classes and interfaces available and accessible to the current source code.
23. [**instanceof**](https://www.javatpoint.com/downcasting-with-instanceof-operator)**:** Java instanceof keyword is used to test whether the object is an instance of the specified class or implements an interface.
24. [**int**](https://www.javatpoint.com/int-keyword-in-java)**:** Java int keyword is used to declare a variable that can hold a 32-bit signed integer.
25. [**interface**](https://www.javatpoint.com/interface-in-java)**:** Java interface keyword is used to declare an interface. It can have only abstract methods.
26. [**long**](https://www.javatpoint.com/long-keyword-in-java)**:** Java long keyword is used to declare a variable that can hold a 64-bit integer.
27. **native:** Java native keyword is used to specify that a method is implemented in native code using JNI (Java Native Interface).
28. [**new**](https://www.javatpoint.com/new-keyword-in-java)**:** Java new keyword is used to create new objects.
29. [**null**](https://www.javatpoint.com/null-keyword-in-java)**:** Java null keyword is used to indicate that a reference does not refer to anything. It removes the garbage value.
30. [**package**](https://www.javatpoint.com/package)**:** Java package keyword is used to declare a Java package that includes the classes.
31. [**private**](https://www.javatpoint.com/private-keyword-in-java)**:** Java private keyword is an access modifier. It is used to indicate that a method or variable may be accessed only in the class in which it is declared.
32. [**protected**](https://www.javatpoint.com/protected-keyword-in-java)**:** Java protected keyword is an access modifier. It can be accessible within the package and outside the package but through inheritance only. It can't be applied with the class.
33. [**public**](https://www.javatpoint.com/public-keyword-in-java)**:** Java public keyword is an access modifier. It is used to indicate that an item is accessible anywhere. It has the widest scope among all other modifiers.
34. [**return**](https://www.javatpoint.com/return-keyword-in-java)**:** Java return keyword is used to return from a method when its execution is complete.
35. [**short**](https://www.javatpoint.com/short-keyword-in-java)**:** Java short keyword is used to declare a variable that can hold a 16-bit integer.
36. [**static**](https://www.javatpoint.com/static-keyword-in-java)**:** Java static keyword is used to indicate that a variable or method is a class method. The static keyword in Java is mainly used for memory management.
37. [**strictfp**](https://www.javatpoint.com/strictfp-keyword)**:** Java strictfp is used to restrict the floating-point calculations to ensure portability.
38. [**super**](https://www.javatpoint.com/super-keyword)**:** Java super keyword is a reference variable that is used to refer to parent class objects. It can be used to invoke the immediate parent class method.
39. [**switch**](https://www.javatpoint.com/java-switch)**:** The Java switch keyword contains a switch statement that executes code based on test value. The switch statement tests the equality of a variable against multiple values.
40. [**synchronized**](https://www.javatpoint.com/synchronization-in-java)**:** Java synchronized keyword is used to specify the critical sections or methods in multithreaded code.
41. [**this**](https://www.javatpoint.com/this-keyword)**:** Java this keyword can be used to refer the current object in a method or constructor.
42. [**throw**](https://www.javatpoint.com/throw-keyword)**:** The Java throw keyword is used to explicitly throw an exception. The throw keyword is mainly used to throw custom exceptions. It is followed by an instance.
43. [**throws**](https://www.javatpoint.com/throws-keyword-and-difference-between-throw-and-throws)**:** The Java throws keyword is used to declare an exception. Checked exceptions can be propagated with throws.
44. [**transient**](https://www.javatpoint.com/transient-keyword)**:** Java transient keyword is used in serialization. If you define any data member as transient, it will not be serialized.
45. [**try**](https://www.javatpoint.com/try-catch-block)**:** Java try keyword is used to start a block of code that will be tested for exceptions. The try block must be followed by either catch or finally block.
46. **void:** Java void keyword is used to specify that a method does not have a return value.
47. [**volatile**](https://www.javatpoint.com/volatile-keyword-in-java)**:** Java volatile keyword is used to indicate that a variable may change asynchronously.
48. [**while**](https://www.javatpoint.com/java-while-loop)**:** Java while keyword is used to start a while loop. This loop iterates a part of the program several times. If the number of iteration is not fixed, it is recommended to use the while loop.

**Conditional Statements in JAVA :**



**If else :**

Class abc{

Public static void main(String[] args){

Int a =5;

Int b=6;

If(a>b){

System.out.println(“A is greater than B”);

}

else{

System.out.println(“B is greater than A”);

}

}

}

#refer the workspace for conditional statements

Date : 6-01-2023 Time : 5:00 pm

**Object Oriented Programming(OPP’s)**

An object is a real-world entity such as pen , chair, bike etc. Object oriented programming is a methodology or paradigm to design program using classes and objects .

It simplifies the software development and maintenance by providing some concepts :

* Objects
* Class
* Inheritance
* Polymorphism
* Abstraction
* Encapsulation



**Class :** it is the blueprint from which an object is created . class doesn’t consume any space.

**Object :** it is the instance of class , it takes up some space in memory.

**Inheritance:** when one object acquires all the properties and behaviours of parent object, its known as inheritance .

**Polymorphism:** if one task is performed in different ways it is known as polymorphism.

**Abstraction :** hiding the internal details and showing the functionality is known as abstraction e.g. T.V remote, mobile etc. In Java we use abstract class and interface to achieve abstraction.

**Encapsulation:** binding data and code together into a single unit are known as encapsulation .

**Naming conventions**

**Class**:

It should start with the uppercase letter.  
It should be a noun such as Color, Button, System, Thread, etc.  
Use appropriate words, instead of acronyms.

e.g. : Employee

**Interface:**

It should start with the uppercase letter.  
It should be an adjective such as Runnable, Remote, ActionListener.  
Use appropriate words, instead of acronyms.

e.g. :Printable

**Method:**

It should start with lowercase letter.  
It should be a verb such as main(), print(), println().  
If the name contains multiple words, start it with a lowercase letter followed by an uppercase letter such as actionPerformed().

E.g. : draw()

**Variables:**

It should start with a lowercase letter such as id, name.  
It should not start with the special characters like & (ampersand), $ (dollar), \_ (underscore).  
If the name contains multiple words, start it with the lowercase letter followed by an uppercase letter such as firstName, lastName.  
Avoid using one-character variables such as x, y, z.

E.g. : int id;

**Package:**

It should be a lowercase letter such as java, lang.  
If the name contains multiple words, it should be separated by dots (.) such as java.util, java.lang.

e.g. : com.badriTutorials.java

**Constant:**

It should be in uppercase letters such as RED, YELLOW.  
If the name contains multiple words, it should be separated by an underscore(\_) such as MAX\_PRIORITY.  
It may contain digits but not as the first letter.

e.g. : MIN\_AGE =18;