**JAVA Introduction :-**

1. JAVA was developed by Sun Microsystems Inc in 1991, later acquired by Oracle Corporation. It was developed by James Gosling(Father of Java) along with others (Patrick Naughton and Mike Sheridon).There teams name was Green Team.
2. Initially, name of the programming language was Green talk with extension (.gt) in 1991. Then later on they changed to OAK but unfortunately there was another trade mark company with the same name so they dropped ‘Oak’ this name.
3. Around in early 1995 they gave ‘JAVA’ as their name for the programming language.
4. Java is a popular programming language, created in 1995.It is owned by Oracle, and more than 3 billion devices run Java.
5. It is used for:

* Mobile applications (specially Android apps)
* Desktop applications
* Web applications
* Web servers and application servers
* Games
* Database connection
* And much, much more!

**JDK :-**

* 1. The Java Development Kit (JDK) is a software development environment used for developing Java applications and applets. It includes the Java Runtime Environment (JRE), an interpreter/loader (Java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc) and other tools needed in Java development.
  2. **Java Development Kit** (in short JDK) is Kit which provides the environment to **develop and execute(run)** the Java program. JDK is a kit(or package) which includes two things
     + Development Tools(to provide an environment to develop your java programs)
     + JRE (to execute your java program).

**Note :**JDK is only used by Java Developers.(any person who wants to write/create the java code)

--------------------------------------------------------------------------------------------

**JRE:-**

1. **JRE stands for “Java Runtime Environment”**and may also be written as “Java RTE.” The Java Runtime Environment provides the minimum requirements for executing a Java application; it consists of the *Java Virtual Machine (JVM), core classes*, and *supporting files*.
2. **Java Runtime Environment** (to say JRE) is an installation package which provides environment to **only run(not develop)** the java program(or application)onto your machine. JRE is only used by them who only wants to run the Java Programs i.e. end users of your system.

**-------------------------------------------------------------------------**

**JVM:-**

1. **Java Virtual machine**(JVM) is a very important part of both JDK and JRE because it is contained or inbuilt in both. Whatever Java program you run using JRE or JDK goes into JVM and JVM is responsible for **executing the java program line by line** hence it is also known as interpreter.

**---------------------------------------------------------------------------------------**

**How is Java platform independent?**

1. The meaning of platform-independent is that the java compiled code(byte code) can run on all operating systems.
2. A program is written in a language that is a human-readable language. It may contain words, phrases, etc which the machine does not understand. For the source code to be understood by the machine, it needs to be in a language understood by machines, typically a machine-level language. So, here comes the role of a compiler. The compiler converts the high-level language (human language) into a format understood by the machines. Therefore, a compiler is a program that translates the source code for another program from a programming language into executable code.
3. This executable code may be a sequence of machine instructions that can be executed by the CPU directly, or it may be an intermediate representation that is interpreted by a virtual machine. This intermediate representation in Java is the **Java Byte Code.**

**Step by step Execution of Java Program:**

* Whenever, a program is written in JAVA, the javac compiles it.
* The result of the JAVA compiler is the **.class file or the bytecode** and not the machine native code (unlike C compiler).
* The bytecode generated is a non-executable code and needs an interpreter to execute on a machine. This interpreter is the JVM and thus the Bytecode is executed by the JVM.
* And finally program runs to give the desired output.

1. In case of C or C++ (language that are not platform independent), the compiler generates an .exe file which is OS dependent. When we try to run this .exe file on another OS it does not run, since it is OS dependent and hence is not compatible with the other OS.
2. In Java, the main point here is that the JVM depends on the operating system – so if you are running Mac OS X you will have a different JVM than if you are running Windows or some other operating system.
3. **Important Points:**
   1. In the case of Java, **it is the magic of Bytecode that makes it platform independent**.
   2. This adds to an important feature in the JAVA language termed as **portability**. Every system has its own JVM which gets installed automatically when the jdk software is installed. For every operating system separate JVM is available which is capable to read the .class file or byte code.
   3. An important point to be noted is that while **JAVA is platform-independent language, the JVM is platform-dependent.** Different JVM is designed for different OS and byte code is able to run on different OS.

**For-each loop in Java**

For-each is another array traversing technique like for loop, while loop, do-while loop introduced in Java5.

* It starts with the keyword **for** like a normal for-loop.
* Instead of declaring and initializing a loop counter variable, you declare a variable that is the same type as the base type of the array, followed by a colon, which is then followed by the array name.
* In the loop body, you can use the loop variable you created rather than using an indexed array element.
* It’s commonly used to iterate over an array or a Collections class (eg, ArrayList)

**Syntax:**

for (type var : array)

{

statements using var;

}

**is equivalent to:**

for (int i=0; i<arr.length; i++)

{

type var = arr[i];

statements using var;

}

**Limitations of for-each loop**

* 1. For-each loops are not appropriate when you want to modify the array:

for (int num : marks)

{

// only changes num, not the array element

num = num\*2;

}

* 1. For-each loops do not keep track of index. So we can not obtain array index using For-Each loop

for (int num : numbers)

{

if (num == target)

{

return ???; // do not know the index of num

}

}

* 1. For-each only iterates forward over the array in single steps

// cannot be converted to a for-each loop

for (int i=numbers.length-1; i>0; i--)

{

System.out.println(numbers[i]);

}

* 1. For-each cannot process two decision making statements at once

// cannot be easily converted to a for-each loop

for (int i=0; i<numbers.length; i++)

{

if (numbers[i] == arr[i])

{ ...

}

}

**public static void main statement**

Explanation:

class Demo {

public static void main(String[] args)

{

System.out.println("Hello world");

}

}

Every word in the public static void main statement has got a meaning to the JVM.

1. Public: It is an Access modifier, which specifies from where and who can access the method. Making the main() method public makes it globally available. It is made public so that JVM can invoke it from outside the class as it is not present in the current class.
2. Static: It is a keyword which is when associated with a method, makes it a class related method. The main() method is static so that JVM can invoke it without instantiating the class. This also saves the unnecessary wastage of memory which would have been used by the object declared only for calling the main() method by the JVM.
3. Void: It is a keyword and used to specify that a method doesn’t return anything. As main() method doesn’t return anything, its return type is void. As soon as the main() method terminates, the java program terminates too. Hence, it doesn’t make any sense to return from main() method as JVM can’t do anything with the return value of it.
4. main: It is the name of Java main method. It is the identifier that the JVM looks for as the starting point of the java program. It’s not a keyword.
5. String[] args: It stores Java command line arguments and is an array of type java.lang.String class. Here, the name of the String array is args but it is not fixed and user can use any name in place of it.

# System.out.println in Java

Java System.out.println() is used to print an argument that is passed to it. The statement can be broken into 3 parts which can be understood separately as:

1. System: It is a final class defined in the java.lang package.
2. out: This is an instance of PrintStreamtype, which is a public and static member field of the System class.
3. println(): As all instances of PrintStream class have a public method println(), hence we can invoke the same on out as well. This is an upgraded version of print(). It prints any argument passed to it and adds a new line to the output. We can assume that System.out represents the Standard Output Stream.

Syntax:

System.out.println(parameter)

Parameters: The parameter might be anything that the user wishes to print on the output screen.

**JAVA DATA TYPES:-**

Data types are divided into two groups:

* Primitive data types - includes byte, short, int, long, float, double, boolean and char
* Non-primitive data types - such as [String](https://www.w3schools.com/java/java_strings.asp), [Arrays](https://www.w3schools.com/java/java_arrays.asp) and [Classes](https://www.w3schools.com/java/java_classes.asp)

## Primitive Data Types

A primitive data type specifies the size and type of variable values, and it has no additional methods.

There are eight primitive data types in Java:

|  |  |  |
| --- | --- | --- |
| Data Type | Size | Description |
| byte | 1 byte | Stores whole numbers from -128 to 127 |
| short | 2 bytes | Stores whole numbers from -32,768 to 32,767 |
| int | 4 bytes | Stores whole numbers from -2,147,483,648 to 2,147,483,647 |
| long | 8 bytes | Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 |
| float | 4 bytes | Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits |
| double | 8 bytes | Stores fractional numbers. Sufficient for storing 15 decimal digits |
| boolean | 1 bit | Stores true or false values |
| char | 2 bytes | Stores a single character/letter or ASCII values |

# Java Variables

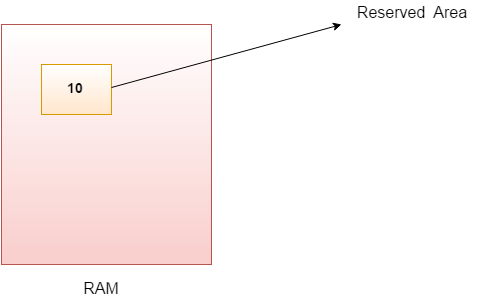
A variable is a container which holds the value while the [Java program](https://www.javatpoint.com/simple-program-of-java) is executed. A variable is assigned with a data type.

Variable is a name of memory location. There are three types of variables in java: local, instance and static.

There are two types of [data types in Java](https://www.javatpoint.com/java-data-types): primitive and non-primitive.

## Variable

**Variable** is name of reserved area allocated in memory. In other words, it is a name of memory location. It is a combination of "vary + able" that means its value can be changed.



**synatx:- datatype variable\_name = value;**

**for eg:- int** d=50;//Here d is the variable

**Programs:-**

**Q1) WAP to print addition of 2 numbers**

**Q2) WAP to print the area of a circle**

**Q3) WAP to print the perimeter of the triangle**

**Q4), Q5), Q6)(repeat the above same 3 questions with scanner)**

**Q7) WAP to swap the Numbers without using 3rd Variable**

**Q8) WAP to swap the Numbers with using 3rd Variable**

**Q9) WAP to check whether the person is Adult or minor**

**Q10) WAP to check whether the person is eligible for license or not if that person is adult then he/she is eligible, minor & senior citizen is not eligible (hint:- use nested if)**

**Q11) WAP to find the largest value from the 2 digits using nested if**

**Q12) WAP to find the largest value from 3 digits**

**Q13) WAP to find the Grade of a student as per percentage (hint:-Ladder if elseif)**

**percentage(100 till 95) - grade O(outstanding)**

**percentage(94 till 80) - grade A**

**percentage(79 till 70) - grade B**

**percentage(69 till 60) - grade C**

**percentage(59 till 50) - grade D**

**percentage(49 till 40) - grade E**

**percentage(below 40) - grade F(FAIL)**

**Q14) WAP for Calculator which has Addition, Subtraction, Multiplication, Division.**

**Q15) WAP to display the name of the day of a week**

**Q16) WAP to print table of any number.**

**(format -**

**2x1=2**

**2x2=4**

**2x3=6 ....etc)**

**Q17) WAP to print all the numbers entered by user one by one untill the user enters zero (hint:- use While loop)**

**Q18) WAP to print the sum of all the numbers entered by user untill the user enters zero (hint:- use doWhile loop)**

**Q19) WAP to print the reverse of a number and to check whether the number is palindrome or not.**

**Q20) WAP to check whether the number positive or negative**

**Q21) WAP to find the factorial of a number**

**Q22)WAP to print the array elements (using both for loop and for each loop)**

**Q23) WAP to insert the array elements dynamically through scanner**

**Q24) WAP to find the Maximum and Minimum value from an array**

**Q25) WAP to reverse a string**

**Q26) WAP to find a word within a String**

**Q27) WAP to sort the Array elements in Ascending and Descending order**

**Q28) WAP to find the count of vowels and consonant from a String**

**Q29) WAP to increase the salary of an employee according to their designation/Post/Role. Increase 1000Rs for office boy, 5000 for clerk and 10000 for manager.**

**Q30) WAP to Store Even & Odd Elements of an Array into Separate Arrays.**

**Q31) WAP to Print the Triangle Pyramid of astride.**

**Q32) WAP to Print the Right angle Pyramid of astride.**

**Q33) WAP to print the Armstrong numbers from between 0 to 999.**

**Q34) WAP to print the Fibonacci series numbers for a given count.**

**Q35) WAP to check whether the given year is a Leap Year or not.**

**Q36) WAP to Swapping Characters of a String.**

**Q37)**