

Last Edit: 24-Jul-09

JAVA**Further
Inside***History of Java***At This
Level**

Sun Microsystems set up a team in 1990 to develop software for consumers electronics devices. **James Gosling** headed the team first C++ was chosen for the development. But for a number of technical reasons, C++ was dropped and a new language called **Oak** was developed. **Oak** was based on C++ but use of pointers, multiple inheritance, programmatic memory management and automatic type conversion were eliminated automatic type conversion were eliminated.

In 1994 internet was becoming very popular and they found this language suitable for the internet. They named it **JAVA**. They also developed a browser named HotJava, which was the first browser to support Java. Java was introduced by Sun Microsystems in 1995. Java has released **JDK1.2** with a number of facilities added. Sun describes Java as a simple, object oriented, distributed, interpreted, robust, secure, architecture neutral, portable, high-performance, multithreaded and dynamic language.

Features of Java

- *Java is simple* : Java uses many of the language constructs of C & C++. So it is easy for C and C++ programmers to migrate to Java.
- *Java is Object Oriented* : Java is a truly object oriented language.
- *Java is distributed* : The aim of Java team was to design a language that supports network applications.
- *Java is interpreted* : Java is also unusual in that each Java program is both compiled and interpreted. With a compiler, you translate a Java program into an intermediate language called Java byte codes- the platform independent codes interpreted by the Java interpreter. With an interpreter, each Java byte codes instruction is passed and run on the computer. Compilation happens just once, interpretation occurs each time the program is executed. These byte codes can be run on any platform that has an implementation for the Java interpreter and the run time system collectively implement a virtual machine called the **Java Virtual Machine**
- *Java is robust* : Java, has been designed for developing robust software.
- *Java is secure* : Java is a language concentrating on the network. Java takes care of the security.
- *Java is portable* : Java environment can be ported to an new hardware platform and operating system,
- *Java is platform independent* : Platform independence is another way of saying that Java is architectural neutral. Platform independence is the capability of the same program to work on different operating system without any modification in its code. Thus Java is said to be platform independent.

The Java Platform

A platform is the H/W or S/w environment in which a program runs. **The Java platform has two components :**

- Java Virtual Machine (JVM)
- The Java Application Programming Interface (Java API)

How Java Works :

Java uses a compiler to convert human readable source code into executable programs Java compilers generates architecture independent byte codes. The byte codes can be executed by only a **Java Virtual Machine (JVM)**, Which is an idealized Java Processor chip usually implemented in software rather than hardware.

To execute Java byte codes, the VM used a class loaders to fetch the byte codes from a disk or from the network each class file is fed to a byte code verifier that ensures the class is formatted correctly and that the class will not corrupt memory when it is executed.

The execution unit of the VM carries out the instructions specified in the byte codes the execution unit is an interpreter, which is a program that reads the byte codes. interprets their meaning and performs the associated functions.

Interpreters are much slower than native code compilers because they continuously need to look up the meaning of each byte codes during execution. There is an elegant alternative to interpreting code, called **Just - in - time (JIT) compilation**.

The JIT compiler converts the byte code to native code instructions on the user's machine immediately before execution.

Java Enabled Browsers :

Java Enabled Web Browsers contains its own VM.

JDK J.2 (Java Development Kit)

JDK is a bundle of a Java compiler & a Java Runtime environment. The easiest way we can get a Java

compiler and a Java Runtime Environment is by downloading it from the Sun's site.

For writing a Java program we must use the text editor supplied by our platform. In case of Windows platform we can use Notepad for writing the program.

A First Java Program :

Let us start by compiling and running the short sample and simple programs shown below.

Class Simple

```
{
public static void main(String args[] )
{
System.out.println ( " This is my first Java Program " );
}
}
```

For most computer languages, the name of the file that holds the source code to a program is arbitrary. However, this is not the case in Java. For the above program the name of the source file should be simple Java. Let's see why.

The Java compiler requires that a source file use the Java filename extension. By looking at the program the name of the class defined by the program is also simple. This is not a coincidence. In Java, all code must reside inside a class. By convention, the name of that class should match exactly with the name of the file that holds the program.

Compiling the Program

To compile the simple program, execute the compiler, Javac, specifying the source file on the command line, as shown below :

```
C:\> javac Simple.java
```

The javac compiler creates a file called simple.class that contains the byte code version of program. The java byte code is the intermediate representation of your program that contains instructions the Java interpreter will execute. Thus the output of the javac is not code that can be directly executed.

To run the program, you must use the Java interpreter; called java. To do so pass the class name simple as a command line argument; as shown below.

```
C:\> java Simple
```

when the program is run, you can see the following output on your monitor.

This is my first Java program.

When Java source code is compiled, each individual class is put into its own output file named after the class and using the .class extension. This is why it is a good idea to give your Java Source files the same name as the class they contain the name of the source file will match the name of the .class file. When you execute the java interpreter as shown above you are actually specifying the name of the class that you want the interpreters to execute.

To fully understand what is happening let's examine the program; by line.

Class Simple

```
{
The first line uses the keyword class to declare that a new class is being defined Simple is an identifier that is the name of the class. The entire class definition including all of its members will be between the opening curly brace ( { ) and the closing ( } ) brace.
```

The next two lines are :

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

This is the line where the program will begin executing. All java applications begin execution by calling main()

The public keyword is an access specifier, which allows the programmer to control the visibility of class members. When a class member is preceded by public, then that member may be accessed by code outside the class in which it is declared.

The keyword static allows main () to be called without having to instantiate a particular instance of the class. This is necessary since main () is called by the Java interpreter before any objects are made.

The keyword void simply tells the compiler that main() does not return any value.

Any information that you need to pass to a method is received by variables specified within the set of parenthesis that follow the name of the method. These variables are called parameters required for a given method, you still need to include the empty parenthesis.

In main (), there is only one parameter string args[]. String args [] declares a parameter named args which is an array of instances of the class string.

The next line character is { . This signals the start of main ()'s body. All of the code that comprises a method will occur between the method's opening curly brace and a closing brace.

The next line in the Program reads as **System.out.println** (" This is my first java program ");

This line is inside the main (). This line outputs the string " This is my first java program".

Output is actually accomplished by the built in println () method. In this case println () displays the string which is passed to it. System is a predefined class that provides access to the system and out is the

output stream that is connected to the console.

All the statement in java end with semicolon. The reason that the other lines in the program do not end in a Semicolon is that they are not technically statements.

Tokens and Identifiers

When you submit a Java program to the Java computer, the computer parses the text and extracts individual tokens. A token is the smallest element of a program that is meaningful to the computer. These tokens define the structure of Java language. All the tokens that comprise the Java are known as the Java Token Set. Java tokens can be divided into 5 categories. Identifiers, Keywords, Literals, Operators and Separators.

Identifiers: -Identifiers are tokens that represent names. There are some limitations in naming identifiers in Java. All Java identifiers are case sensitive and must begin with a letter, an underscore (_) or a dollar sign (\$).

Letters includes both uppercase and lowercase letters. Subsequent identifier characters can be include the number 0 to 9. The only limitations to the identifier names is that the Java keywords such as long, short, int, while, do can't be used.

Keywords: -Keywords are the predefined identifiers reserved by Java for a specific purpose and are used only in a limited, specified manner.

Variables And Data Types: -

Variables:- Variable is the basic unit of storage in a java program. A variable is defined by the combination of an identifier, a type and an optional initializer.

In java all variables must be declared before they can be used. The basic form of a variable declaration is:

type identifier;

The declaration statement tells the compiler to set aside memory variable of type *type* with the name *identifier*.

Data Types: -Java defines eight simple types of data, which are also called as literals. These are the program elements used in an invariant manner. The eight data types or literals are as follows : byte, short, int, long, char, float, double, Boolean. These can be put into four groups as:

- 1) *Integer literals*
- 2) *Floating point literals*
- 3) *Boolean literals*
- 4) *Character and string literals*

Integer Literals:- Integers are probably the most commonly used type in the typical program. These are the primary literals used in Java Programming. Java defines 4 types of integer literals: *byte, short, int, long* which all have signed positive and negative value.

Width and range of these integer types vary widely as shown:

<u>Name</u>	<u>Width in Bits</u>	<u>Range</u>
byte	8	-126 to 127
short	16	-32,768 to 32,767
int	32	-2^{31} to $2^{31}-1$
long	64	-2^{63} to $2^{63}-1$

byte: -*byte* is the smallest integer type. Variables of type byte are especially useful when you are working with a stream of data from network or file. byte variables are declared by using *byte* keyword.

byte b;

int: - The int type is the most versatile and efficient type and it should be used most of the time when you want to create a number for counting or indexing arrays or doing integer math. int variable are declared by *int* keyword.

int x; int y;

long: -long datatype is used when an int type is not large enough to hold the desired value. The range of long is quite large. This makes it useful when big, whole numbers are needed. long variables are declared by using keyword *long*.

long a; long b;

Floating Point Literals: -These are also known as real numbers, which are used when evaluating expression that requires fractional precision such 3.142. Floating point literals in Java default to the double type which is a 64-bit value.

float and double variable are declared by using *float* and *double* keyword resp.

float x; float y;

The width and range of floating point literal is as shown below:

<u>Floating point literals</u>	<u>Width in bits</u>	<u>Range</u>
Double	64	1.7e-308 to 1.7e+308
float	32	3.4e-038 to 3.4e+038

Boolean Literals: -Java has a simple type called boolean,for logical values.It has only one of the 2 possible values, true or false. Boolean is the type required by the conditional expressions that govern the control statements such as if and for.The important thing is that the values of true and false do not convert into numerical representation.boolean variables use the keyword *boolean* for declaring them.

boolean a; boolean b;

Type Conversion And Casting: -

Java's Automatic Conversion:

When one type of data is to be assigned to another type of a variable,an automatic type conversion will take place,if the following 2 conditions are met:

- 1) The 2 types are compatible.
- 2) The destination type is larger than the source type.

When these 2 conditions are met ,a widening conversion takes place. The in type is always large enough to hold all valid byte values and thus no explicit cast statement is required.

Java also performs an automatic type conversion when storing a literal integer constant into variables of type byte,short or long.

Casting Incompatible Type: -

While assigning an int value to a byte variable,the conversion is not performed automatically,because a *byte* is smaller than an *int*.This kind of conversion is called narrowing conversions.

To create a conversion between 2 compatible types, you must use a cast.A cast is simply explicit type conversion.It has the general form as:

(target type) value

Operators: -Operators enable you to perform an evaluation or computation on a data objects.

Integer Operators:-There are 3 types of operators that can be performed on integers:*unary,binary and relational*.Unary operators act only on single integer numbers where as binary operator acts on pair of integers .Relational operators acts on 2 integer numbers but return a Boolean result rather than an integer.

Binary Integer Operators:

Binary integer operators act on pairs of integers.Following are the binary integer operators which are also called basic arithmetic operator.

Description Operator

Addition	+
subtraction	-
Multiplication	*
Division	/
Modulus	%

Unary Integer Operator:- Unary integer operator acts on a single integer.

Description Operator

Increment	++
Decrement	--

The increment operator increases its operand by 1.

The decrement operator decreases its operand by 1.

Modulus operator(%):It returns the remainder of division operation.

Relational Operator: -

Description Operator

Equal to	==
Not Equal to	!=
Greater than	>
Less than	<
Greater than or Equal to	>=
Less than or Equal to	<=

Boolean Logical Operators:-

Description Operator

Logical And	&&
Logical OR	
Equal to	==
Not equal to	!=

Control Flow Statements:-

There are 2 types of control flow statements:

- 1) Conditional Flow Statements
- 2) Repeatative Flow Statements

In Conditional Flow statements the flow of the program is goes as per the conditions specified in the program. The types of conditional statements are **if-else and switch**

The syntax of if statement is:

```

if(expression)
  {
    block of statements;
  }
else
  {
    block of statements;
  }

```

If condition is satisfied then *if* block gets executed otherwise *else* block gets executed.

E.g.

```

class trial
{
  public static void main(String args[])
  {
    int marks;
    marks=50;
    if(marks>=35)
    {
      System.out.println("PASS");
    }
    else
    {
      System.out.println("PASS");
    }
  }
}

```

The switch statement:-

We provide a list of possible values for a variable and code to execute when that value matches the variable in the switch statement. The variable we compare in a switch statement must be of type int,long,char or byte and must be enclosed in parenthesis

E.g.

```

class trial
{
  public static void main(String args[])
  {
    char color="b";
    switch(color)
    {
      case 'r':
        System.out.println("RED");
        break;
      case 'b':
        System.out.println("BLUE");
        break;
      case 'o':
        System.out.println("ORANGE");
        break;
      case 'v':
        System.out.println("VIOLET");
        break;
      default:
        System.out.println("Colour lesss");
    }
  }
}

```

Repeative Statements Or Iterative Statements:-

These are also called as Loops, enables you to execute code repeatedly. It can be defined as the one which repeatedly executes the same set of instructions until the termination condition is met.

Java's iteration statements includes : **while,do-while and for.**

while:-

The *while* loop is Java's most fundamental looping statement. It repeats a statement or block while its controlling expression is true. Its syntax is:

```
while( condition )
{
    body of loop
}
```

The condition can be any boolean expression. The body of the loop will be executed as long as the conditional expression is true. When the condition becomes false, control passes to the next line of the code immediately following the loop. The curly braces are unnecessary if only single statement is being repeated..

E.g.**class WhileLoop**

```
{
    public static void main(String args[])
    {
        int n=0;
        while(n<=10)
        {
            System.out.println("Good Morning :"+n);
            n++;
        }
    }
}
```

Since the while loop evaluates the conditional expression at the top of the loop, the body of the loop will not execute even once if the condition is false at the beginning.

do-while:-

Sometimes it is desirable to execute the body of *while* loop at once, even if the conditional expression is false to begin with i.e. there are sometimes when you would like to test the termination expression at the end of the loop rather than at the beginning. The do-while loop always executes its body at least once, because its conditional expression is at the bottom of the loop. The syntax is,

```
do
{
    body of the loop
}
while(condition);
```

Each iteration of the do while loop first executes the body of the loop and then evaluates the conditional expression. If this expression is true, the loop will repeat else the loop terminates. As with all of the Java's loop, condition must be a boolean expression.

The same "Good Morning" program can be written with do-while loop as follows:

E.g.**class doWhile**

```
{
    public static void main(String args[])
    {
        int n=0;
        do
        {
            System.out.println("Good Morning :"+n);
            n++;
        }while(n<=10)
    }
}
```

The for loop:-

The for statement is the most useful looping construct available in Java. Java allows multiple comma-separated expressions to appear in the initialization and increment sections, but not the test section of the loop.

We can declare local loop variables in the initialization section of the loop.

```
E.g.  
class For  
{  
    public static void main(String args[])  
    {  
        for(int i=0;i<10;i++)  
        {  
            System.out.println("Good Morning :"+n);  
        }  
    }  
}
```