**Introduction Of java:**

Java is an Object Oriented Programming Language, Developed by **Mr. James Gosling** at **Sun Micro-systems (Sun)** since **1991**.The first name of java was **OAK**. First **version jdk1.0**

1991 1995

-------After 4 year-------------

OAK JAVA

Why Java was named as "Oak"?



**Why Oak?** Oak is a symbol of strength and chosen as a national tree of many countries like the U.S.A., France, Germany, Romania, etc.

In 1995, Oak was renamed as **"Java"** because it was already a trademark by Oak Technologies.

Application

According to Sun, 3 billion devices run Java. There are many devices where Java is currently used. Some of them are as follows:

1. Desktop Applications such as acrobat reader, media player, antivirus, etc.
2. Web Applications such as irctc.co.in.
3. Enterprise Applications such as banking applications.
4. Mobile applications.
5. Embedded System
6. Smart Card
7. Robotics
8. Games, etc.

## Types of Java Applications

There are mainly 4 types of applications that can be created using Java programming:

#### **1) Standalone Application**

Standalone applications are also known as desktop applications or window-based applications. These are traditional software that we need to install on every machine. Examples of standalone application are Media player, antivirus, etc. AWT/Swing and Swing are used in Java for creating standalone applications.

#### **2) Web Application**

An application that runs on the server side and creates a dynamic page is called a web application. Currently, [Servlet](https://www.javatpoint.com/servlet-tutorial), [JSP](https://www.javatpoint.com/jsp-tutorial), [Spring](https://www.javatpoint.com/spring-tutorial), [Hibernate](https://www.javatpoint.com/hibernate-tutorial) etc. technologies are used for creating web applications in Java.

#### **3) Enterprise Application**

An application that is distributed in nature, such as banking applications, etc. is called an enterprise application. It has advantages like high-level security, load balancing, and clustering. In Java, spring mvc , spring boot,IOC, DI, JPA is used for creating enterprise applications.

#### **4) Mobile Application**

An application which is created for mobile devices is called a mobile application. Currently, Android and Java ME are used for creating mobile applications.

## Java Platforms / Editions

There are 4 platforms or editions of Java:

#### **1) Java SE (Java Standard Edition)**

It is a Java programming platform. It includes Java programming APIs such as java.lang, java.io, java.net, java.util, java.sql, java.math etc. It includes core topics like Oops[String](https://www.javatpoint.com/java-string), , Regex, Exception, Inner classes, Multi threading, I/O Stream, Networking, AWT, Swing, Reflection, Collection, etc.

#### **2) Java EE (Java Enterprise Edition)**

It is an enterprise platform that is mainly used to develop web and enterprise applications. It is built on top of the Java SE platform. It includes topics like Servlet, JSP, Web Services, EJB, [JPA](https://www.javatpoint.com/jpa-tutorial), hibernat Spring,Spring Boot,MVC etc.

#### **3) Java ME (Java Micro Edition)**

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

#### **4) JavaFX (**FX is normally the abbreviation given to special effects mostly sound or visual.)

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

**Some Sort Forms**

**JDK**: Java Development Kit.

**JRE**: Java Runtime Environment.

**JVM**: Java Virtual machine. (Interpreter of java)

**API**: Application programming Interface. (Library)

JDK : API + Compiler + JVM

JRE: API + JVM

# **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:

1. **Primitive data types:**The primitive data types include Boolean, char, byte, short, int, long, float and double.
2. **Non-primitive data types:**The non-primitive data types include [Classes](https://www.javatpoint.com/object-and-class-in-java), [Interfaces](https://www.javatpoint.com/interface-in-java), and [Arrays](https://www.javatpoint.com/array-in-java).

## Java Primitive Data Types

In Java language, primitive data types are the building blocks of data manipulation. These are the most basic data types available in Java Language.

There are 8 types of primitive data types:

* boolean data type
* byte data type
* char data type
* short data type
* int data type
* long data type
* float data type
* double data type



|  |  |  |
| --- | --- | --- |
| **Data Type** | **Default Value** | **Default 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 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Integer:** | **Keyword** | **Memory** | **Range** |
| **1.** | byte | 1 byte | -128, 0 ,+127 |
| **2.** | short | 2 byte | -32768, 0 , +32767 |
| **3.** | int | 4 byte | +/- 10 digits |
| **4.** | long | 8 byte | +/- 19 digits |

**Character:**

char 2 byte 0, 65535

c/c++ : ASCII(American standered code for information interchange)

Character code: Uni Code

0, 255 English Characters

**Floating Point:**

1. float 4 byte +/- 10 digits

2. double(default) 8 byte +/- 19 digits

**Note**: In java all floating point constant values are double type by default.

c/c++ 1.5 float

java 1.5 double

float f = 1.5; Wrong

float f = 1.5f; Ok

float f = (float)1.5; ok

double f = 1.5; Ok

**Boolean :**

boolean 1 byte true/false words

boolean b = true;

**Data Types:**

**1.Integer**

**2.Floating Point**

**3.Character**

**4.Boolean**

**Integer:**

**Keyword Memory Range**

**byte 1 byte -128, 0 , +127**

**short 2 byte -32768,0, +32767**

**int 4 byte +/- 10 digits**

**long 8 byte +/-19 digits**

**Floating Point:**

**float 4 bytes +/- 10 digits**

**double 8 byte +/- 19 digits**

**c/c++ : float f=1.5; ok**

**java : float f = 1.5; not ok**

**float f = (float)1.5; ok**

**float f = 1.5f; ok**

**double f = 1.5;**

**Note: in java all floating point values are double type by default.**

**Character:**

**char 2 byte 0,65535**

**character code : Unicode**

**0,255 English character**

**Note : in java char data type and String are different, in java char is a data type and String is a class.**

**Boolean:**

**boolean 1 byte true/false**

**boolean b = true;**

Scanner class :

import java,util.Scanner;

Scanner s = new Scanner(System.in);

s.nextInt();

s.nextFloat();

s.nextDouble();

# Features of Java

The primary objective of Java programming language creation was to make it portable, simple and secure programming language. Apart from this, there are also some excellent features which play an important role in the popularity of this language. The features of Java are also known as Java

buzzwords.

A list of the most important features of the Java language is given below.

1. .Simple
2. .Object Oriented
3. [Portable](https://www.javatpoint.com/features-of-java" \l "Portable)
4. [Platform independent](https://www.javatpoint.com/features-of-java" \l "Platform-independent)
5. [Secured](https://www.javatpoint.com/features-of-java" \l "Secured)
6. [Robust](https://www.javatpoint.com/features-of-java" \l "Robust)
7. [Architecture neutral](https://www.javatpoint.com/features-of-java" \l "Architecture-neutral)
8. [Interpreted](https://www.javatpoint.com/features-of-java" \l "Interpreted)
9. [High Performance](https://www.javatpoint.com/features-of-java" \l "High-Performance)
10. [](https://www.javatpoint.com/features-of-java" \l "Multithreaded)Multithreaded
11. [Distributed](https://www.javatpoint.com/features-of-java" \l "Distributed)
12. [Dynamic](https://www.javatpoint.com/features-of-java" \l "Dynamic)

Simple

Java is very easy to learn, and its syntax is simple, clean and easy to understand. According to Sun Micro system, Java language is a simple programming language because:

* Java syntax is based on C++ (so easier for programmers to learn it after C++).
* Java has removed many complicated and rarely-used features, for example, explicit pointers, operator overloading, multiple inheritance etc.
* Pointer: Java have no any pointer. Explicitly java have no any pointer,

but java is a dynamic language,implicitly java totally work on run time and use pointer.

* There is no need to remove unreferenced objects because there is an Automatic Garbage Collection in Java.

### Object-oriented

Java is an [Object-oriented programming (OOPs) is a methodology that simplifies software development and maintenance by providing some rules.](https://www.javatpoint.com/java-oops-concepts)

Basic concepts of OOPs are:

1. [Object](https://www.javatpoint.com/object-and-class-in-java)
2. [Class](https://www.javatpoint.com/object-and-class-in-java" \l "class)
3. [Inheritance](https://www.javatpoint.com/inheritance-in-java)
4. [Polymorphism](https://www.javatpoint.com/runtime-polymorphism-in-java)
5. [Abstraction](https://www.javatpoint.com/abstract-class-in-java)
6. [Encapsulation](https://www.javatpoint.com/encapsulation)

### Platform Independent

Java is platform independent because it is different from other languages like [C](https://www.javatpoint.com/c-programming-language-tutorial), [C++](https://www.javatpoint.com/cpp-tutorial), etc. which are compiled into platform specific machines while Java is a write once, run anywhere language. 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 a software-based platform.

The Java platform differs from most other platforms in the sense that it is 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 executed on multiple platforms, for example, 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).

### Secured

Java is best known for its security. With Java, we can develop virus-free systems. Java is secured because:

* **No explicit pointer**
* **Java Programs run inside a virtual machine sandbox**



* **Classloader:** Classloader in Java is a part of the Java Runtime Environment (JRE) which is used to load Java classes into the Java Virtual Machine dynamically. It adds security by separating the package for the classes of the local file system from those that are imported from network sources.
* **Bytecode Verifier:** It checks the code fragments for illegal code that can violate access rights to objects.
* **Security Manager:** It determines what resources a class can access such as reading and writing to the local disk.

Java language provides these securities by default. Some security can also be provided by an application developer explicitly through SSL, JAAS, Cryptography, etc.

### Robust

The English mining of Robust is strong. Java is robust because:

* It uses strong memory management.
* There is a lack of pointers that avoids security problems.
* Java provides automatic garbage collection which runs on the Java Virtual Machine to get rid of objects which are not being used by a Java application anymore.
* There are exception handling and the type checking mechanism in Java. All these points make Java robust.

### Architecture-neutral

Java is architecture neutral because there are no implementation dependent features, for example, the size of primitive types 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.

### Portable

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

### Distributed

Java is distributed because it facilitates users to create distributed applications in Java. spring and boot are used for creating distributed applications. This feature of Java makes us able to 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 doesn't occupy memory for each thread. It shares a common memory area. Threads are important for multi-media, Web applications, etc.

### Dynamic

Java is a dynamic language. It supports the dynamic loading of classes. It means classes are loaded on demand. It also supports functions from its native languages, i.e., C and C++.

# Internal Details of Hello Java Program

In the previous section, we have created Java Hello World program and learn how to compile and run a Java program. In this section, we are going to learn, what happens while we compile and run the Java program. Moreover, we will see some questions based on the first program.

## What happens at compile time?

At compile time, the Java file is compiled by Java Compiler (It does not interact with OS) and converts the Java code into bytecode.



## What happens at run time?

At run time, the following steps are performed:



**Classloader:** It is the subsystem of JVM that is used to load class files.

**Bytecode Verifier:** Checks the code fragments for illegal code that can violate access rights to objects.

**Interpreter:** Read bytecode stream then execute the instructions.

Setting Java Path in Linux OS

Setting path in Linux OS is the same as setting the path in the Windows OS. But, here we use the export tool rather than set. Let's see how to set path in Linux OS:

export PATH=$PATH:/home/jdk1.6.01/bin/

### Types of Variables

There are three types of variables in [Java](https://www.javatpoint.com/java-tutorial):

* 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 an instance variable. It is not declared as [static](https://www.javatpoint.com/static-keyword-in-java).

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

#### **3) Static variable**

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

### Example to understand the types of variables in java

1. **public** **class** A
2. {
3. **static** **int** m=100;//static variable
4. **void** method()
5. {
6. **int** n=90;//local variable
7. }
8. **public** **static** **void** main(String args[])
9. {
10. **int** data=50;//instance variable
11. }
12. }//end of class

### Java Variable Example: Add Two Numbers

1. **public** **class** Simple{
2. **public** **static** **void** main(String[] args){
3. **int** a=10;
4. **int** b=10;
5. **int** c=a+b;
6. System.out.println(c);
7. }
8. }

**Output:**

20

### Java Variable Example: Widening

1. **public** **class** Simple{
2. **public** **static** **void** main(String[] args){
3. **int** a=10;
4. **float** f=a;
5. System.out.println(a);
6. System.out.println(f);
7. }}

### Java Variable Example: Narrowing (Typecasting)

1. **public** **class** Simple{
2. **public** **static** **void** main(String[] args){
3. **float** f=10.5f;
4. //int a=f;//Compile time error
5. **int** a=(**int**)f;
6. System.out.println(f);
7. System.out.println(a);
8. }}

**Output:**

10.5

10

### Java Variable Example: Overflow

1. **class** Simple{
2. **public** **static** **void** main(String[] args){
3. //Overflow
4. **int** a=130;
5. **byte** b=(**byte**)a;
6. System.out.println(a);
7. System.out.println(b);
8. }}

**Output:**

130

-126

### Java Variable Example: Adding Lower Type

1. **class** Simple{
2. **public** **static** **void** main(String[] args){
3. **byte** a=10;
4. **byte** b=10;
5. //byte c=a+b;//Compile Time Error: because a+b=20 will be int
6. **byte** c=(**byte**)(a+b);
7. System.out.println(c);
8. }}

**Output:**

20

# Operators in Java

**Operator** in [Java](https://www.javatpoint.com/java-tutorial) is a symbol that is used to perform operations. For example: +, -, \*, / etc.

There are many types of operators in Java which are given below:

* Unary Operator,
* Arithmetic Operator,
* Shift Operator,
* Relational Operator,
* Bitwise Operator,
* Logical Operator,
* Ternary Operator and
* Assignment Operator.

## Java Operator Precedence

|  |  |  |
| --- | --- | --- |
| Operator Type | Category | Precedence |
| Unary | postfix | expr++ expr-- |
| prefix | ++expr --expr +expr -expr ~ ! |
| Arithmetic | multiplicative | \* / % |
| additive | + - |
| Shift | shift | << >> >>> |
| Relational | comparison | < > <= >= instanceof |
| equality | == != |
| Bitwise | bitwise AND | & |
| bitwise exclusive OR | ^ |
| bitwise inclusive OR | | |
| Logical | logical AND | && |
| logical OR | || |
| Ternary | ternary | ? : |
| Assignment | assignment | = += -= \*= /= %= &= ^= |= <<= >>= >>>= |