**What is Java?**

Java is a **programming language** and a **platform**. Java is a high level, robust, object-oriented and secure programming language.

Java was developed by *Sun Microsystems* (which is now the subsidiary of Oracle) in the year 1995. *James Gosling* is known as the father of Java. Before Java, its name was *Oak*. Since Oak was already a registered company, so James Gosling and his team changed the name from Oak to Java.

Java development typically requires the Java Development Kit (JDK), which includes the Java compiler and other essential tools. You can download the JDK from the official Oracle website and follow the installation instructions for your operating system.

Once we have the JDK installed, you can use a text editor or an Integrated Development Environment (IDE) like IntelliJ IDEA, Eclipse, or NetBeans to write and run your Java code. IDEs provide features such as code completion, debugging, and project management, making them invaluable tools for developers.

**Application:**

**There are various 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, javatpoint.com, etc.**
3. **Enterprise Applications such as banking applications.**
4. **Mobile**
5. **Embedded System**
6. **Smart Card**
7. **Robotics**
8. **Games, etc.**

**Types of Java Applications**

**4-types of applications:**

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 and Swing are used in Java for creating standalone applications.

1. **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), [Struts](https://www.javatpoint.com/struts-2-tutorial), [Spring](https://www.javatpoint.com/spring-tutorial), [Hibernate](https://www.javatpoint.com/hibernate-tutorial), [JSF](https://www.javatpoint.com/jsf-tutorial), etc. technologies are used for creating web applications in Java.

1. **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, [EJB](https://www.javatpoint.com/ejb-tutorial) is used for creating enterprise applications.

1. **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:

**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, Multithreading, I/O Stream, Networking, AWT, Swing, Reflection, Collection, etc.

**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), etc.

**Java ME (Java Micro Edition)**

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

* **History of Java:**

[**James Gosling**](https://www.javatpoint.com/james-gosling-father-of-java)**, Mike Sheridan**, and **Patrick Naughton** initiated the Java language project in June 1991. The small team of sun engineers called **Green Team**.

1. JDK Alpha and Beta (1995)
2. JDK 1.0 (23rd Jan 1996)
3. JDK 1.1 (19th Feb 1997) – JavaBeans, JDBC
4. J2SE 1.2 (8th Dec 1998) - Collections framework,
5. J2SE 1.3 (8th May 2000)
6. J2SE 1.4 (6th Feb 2002) - IPv6 support, Improved libraries.
7. J2SE 5.0 (30th Sep 2004)
8. Java SE 6 (11th Dec 2006)
9. Java SE 7 (28th July 2011)
10. Java SE 8 (18th Mar 2014)
11. Java SE 9 (21st Sep 2017)
12. Java SE 10 (20th Mar 2018)
13. Java SE 11 (September 2018)
14. Java SE 12 (March 2019)
15. Java SE 13 (September 2019)
16. Java SE 14 (Mar 2020)
17. Java SE 15 (September 2020)
18. Java SE 16 (Mar 2021)
19. Java SE 17 (September 2021)
20. Java SE 18 (to be released by March 2022)

**Most important features of the Java language:**

1. [**Simple**](https://www.javatpoint.com/features-of-java#Simple) **-** Java syntax is based on C++ and there is an Automatic Garbage Collection in Java.
2. [**Object-Oriented**](https://www.javatpoint.com/features-of-java#Object-Oriented) **-** Object-oriented programming (OOPs) is a methodology that simplifies software development and maintenance by providing some rules.
3. [**Platform independent**](https://www.javatpoint.com/features-of-java#Platform-independent) **-** A platform is the hardware or software environment in which a program runs. Java is a software-based platform that runs on top of other hardware-based platforms. It has two components: **Runtime Environment and API (Application Programming Interface).**
4. [**Secured**](https://www.javatpoint.com/features-of-java#Secured) **-** Java is best known for its security. With Java, we can develop virus-free systems. **Java is secured because**: No **explicit pointer and Java Programs run inside a virtual machine sandbox**
5. [**Robust**](https://www.javatpoint.com/features-of-java#Robust) **-** Java is robust because:

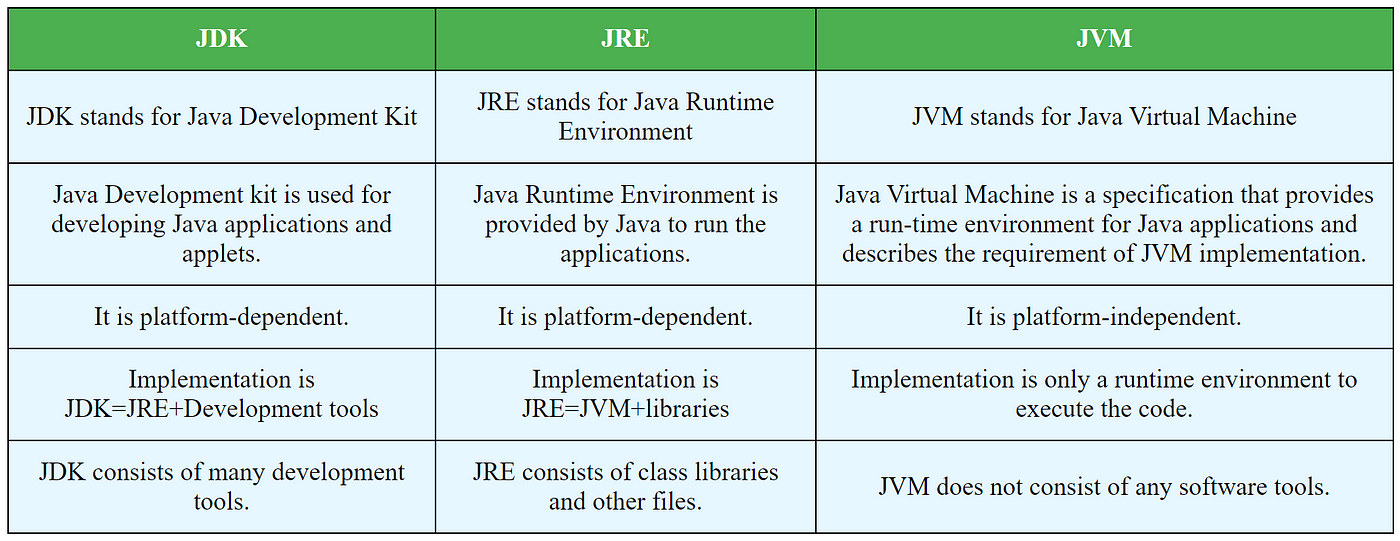
* It uses **strong memory management.**
* There is a **lack of pointers that avoids security problems.**
* Java provides **automatic garbage collection which runs on the Java Virtual Machine** to get rid of objects which are not being used by a Java application anymore.
* There are **exception handling**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

1. [**Multithreaded**](https://www.javatpoint.com/features-of-java#Multithreaded) **-** 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.

**Parameters used in Java Program:**

* **class** keyword is used to declare a class in Java.
* **public** keyword is an access modifier that represents visibility. It means it is visible to all.
* **static** is a keyword. If we declare any method as static, it is known as the static method. The core advantage of the static method is that there is no need to create an object to invoke the static method. The main() method is executed by the JVM, so it does not require creating an object to invoke the main() method. So, it saves memory.
* **void** is the return type of the method. It means it does not return any value.
* The **main()** method represents the starting point of the program.
* **String[] args** or **String args[]** is used for [command line argument](https://www.javatpoint.com/command-line-argument). We will discuss it in coming section.
* **System.out.println()** is used to print statement on the console. Here, System is a class, out is an object of the PrintStream class, println() is a method of the PrintStream class.

**Difference between JDK, JRE, and JVM :**



|  |  |  |
| --- | --- | --- |
| **JDK** | **JRE** | **JVM** |
| * Standard Edition Java Platform * Enterprise Edition Java Platform * Micro Edition Java Platform | It contains a set of libraries + other files that JVM uses at runtime. | * Loads code * Verifies code * Executes code * Provides runtime environment |

**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.

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

* **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.
* **instance variable** - A variable declared inside the class but outside the body of the method, is called an instance variable.
* **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.

**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).

**Primitive data types:**

1. boolean data type
2. byte data type
3. char data type
4. short data type
5. int data type
6. long data type
7. float data type
8. 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 |

**Non-Primitive Data Types:**

**Class**

One common non-primitive data type in Java is the class. Classes are used to create objects, which are instances of the class. A class defines the properties and behaviors of objects, including variables (fields) and methods.

**Interface**

Interfaces are another important non-primitive data type in Java. An interface defines a contract for what a class implementing the interface must provide, without specifying how it should be implemented. Interfaces are used to achieve abstraction and multiple inheritance in Java, allowing classes to be more flexible and reusable.

**Arrays**

Arrays are a fundamental non-primitive data type in Java that allow you to store multiple values of the same type in a single variable. Arrays have a fixed size, which is specified when the array is created, and can be accessed using an index. Arrays are commonly used to store lists of values or to represent matrices and other multi-dimensional data structures.

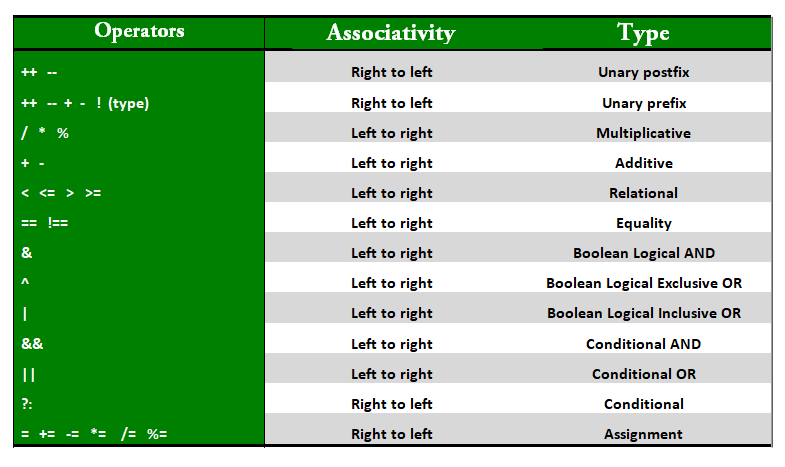
**Unicode System**

|  |
| --- |
| **Unicode is a universal international standard character encoding that is capable of representing most of the world's written languages.** |

**Why java uses Unicode System?**

|  |
| --- |
| **Before Unicode, there were many language standards:** |
| * **ASCII (American Standard Code for Information Interchange) for the United States.** * **ISO 8859-1 for Western European Language.** * **KOI-8 for Russian.** * **GB18030 and BIG-5 for chinese, and so on.** |

**Operators in Java:**



* **Unary Operator:**

|  |
| --- |
| public class Operator Example{  public static void main(String args[]){  int x=10;  System.out.println(x++);//10 (11)  System.out.println(++x);//12  System.out.println(x--);//12 (11)  System.out.println(--x);//10  }}  **Output:**  *10*  *12*  *12*  *10* |

* **Arithmetic Operator:**

|  |
| --- |
| public class OperatorExample{  public static void main(String args[]){  int a=10;  int b=5;  System.out.println(a+b);//15  System.out.println(a-b);//5  System.out.println(a\*b);//50  System.out.println(a/b);//2  System.out.println(a%b);//0  }}  **Output:**  *15*  *5*  *50*  *2*  *0* |

* **Shift Operator**
* **Relational Operator**
* **Bitwise Operator**
* **Logical Operator**
* **Ternary Operator**
* **Assignment Operator**

**List of Java Keywords:**

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. [**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.
12. [**else**](https://www.javatpoint.com/java-if-else)**:** Java else keyword is used to indicate the alternative branches in an if statement.
13. [**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.
14. [**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.
15. [**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.
16. [**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.
17. [**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.
18. [**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.
19. [**if**](https://www.javatpoint.com/java-if-else)**:** Java if keyword tests the condition. It executes the if block if the condition is true.
20. [**implements**](https://www.javatpoint.com/interface-in-java)**:** Java implements keyword is used to implement an interface.
21. [**import**](https://www.javatpoint.com/package)**:** Java import keyword makes classes and interfaces available and accessible to the current source code.
22. [**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.
23. [**interface**](https://www.javatpoint.com/interface-in-java)**:** Java interface keyword is used to declare an interface. It can have only abstract methods.
24. [**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.
25. [**new**](https://www.javatpoint.com/new-keyword-in-java)**:** Java new keyword is used to create new objects.
26. [**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.
27. [**package**](https://www.javatpoint.com/package)**:** Java package keyword is used to declare a Java package that includes the classes.
28. [**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.
29. [**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.
30. [**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.
31. [**return**](https://www.javatpoint.com/return-keyword-in-java)**:** Java return keyword is used to return from a method when its execution is complete.
32. [**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.
33. [**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.
34. [**strictfp**](https://www.javatpoint.com/strictfp-keyword)**:** Java strictfp is used to restrict the floating-point calculations to ensure portability.
35. [**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.
36. [**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.
37. [**synchronized**](https://www.javatpoint.com/synchronization-in-java)**:** Java synchronized keyword is used to specify the critical sections or methods in multithreaded code.
38. [**this**](https://www.javatpoint.com/this-keyword)**:** Java this keyword can be used to refer the current object in a method or constructor.
39. [**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.
40. [**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.
41. [**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.
42. [**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.
43. **void:** Java void keyword is used to specify that a method does not have a return value.
44. [**volatile**](https://www.javatpoint.com/volatile-keyword-in-java)**:** Java volatile keyword is used to indicate that a variable may change asynchronously.
45. [**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.

**Methods:**

|  |  |
| --- | --- |
| **Methods** | **Description** |
| [**after()**](https://www.javatpoint.com/java-timestamp-after-method) | **Returns Boolean value true if this Timestamp object comes later than given Timestamp object.** |
| [**before()**](https://www.javatpoint.com/java-timestamp-before-method) | **Returns Boolean value true if this Timestamp object comes earlier than given Timestamp object.** |
| [**compareTo()**](https://www.javatpoint.com/java-timestamp-compareto-method) | **Compares this Timestamp object to the given Timestamp object or to the given date object** |
| [**equals()**](https://www.javatpoint.com/java-timestamp-equals-method) | **Returns a Boolean value true if this Timestamp object is equal specified object or to the given Timestamp object .** |
| [**from()**](https://www.javatpoint.com/java-timestamp-from-method) | **Obtains an instance of Timestamp from an Instant object** |
| [**getTime()**](https://www.javatpoint.com/java-timestamp-gettime-method) | **Returns the number of milliseconds since January 1, 1970, 00:00:00 GMT** |
| [**hashCode()**](https://www.javatpoint.com/java-timestamp-hashcode-method) | **Returns a hash code value for this object** |
| [**setTime()**](https://www.javatpoint.com/java-timestamp-settime-method) | **Sets this class's object to indicate a point in time (milliseconds) after January 1, 1970 00:00:00 GMT** |
| [**toInstant()**](https://www.javatpoint.com/java-timestamp-toinstant-method) | **Coverts the Timespan object to an Instant which represents the same point on the time-line as this Timestamp** |
| [**toLocalDateTime()**](https://www.javatpoint.com/java-timestamp-tolocaldatetime-method) | **Converts this Timespan object to a LocalDateTime which represents the same date-time value as this Timestamp** |
| [**toString()**](https://www.javatpoint.com/java-timestamp-tostring-method) | **Converts the Timespan object in JDBC timestamp escape format** |
| [**valueOf()**](https://www.javatpoint.com/java-timestamp-valueof-method) | **Converts the string object to Timestamp value or obtains an instance of Timestamp from a LocalDateTime object.** |