Introduction of Java:

JAVA was developed by James Gosling at Sun Microsystems Inc in the year 1995, later acquired by Oracle Corporation. It is a simple programming language. Java makes writing, compiling, and debugging programming easy. It helps to create reusable code and modular programs. Java is a class-based, object-oriented programming language.

Features of Java:

The primary objective of Java programming language creation was to make it portable, simple and secure programming language.

1. **Simple**
2. **Object-Oriented**
3. **Portable**
4. **Platform independent**
5. **Secured**
6. **Robust**
7. **Architecture neutral**
8. **Interpreted**
9. **High Performance**
10. **Multithreaded**
11. **Distributed**
12. **Dynamic**

Simple:

Java is very easy to learn, and its syntax is simple, clean and easy to understand. According to Sun Microsystem, 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, etc.
- 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 language. Everything in Java is an object. Object-oriented means we organize our software as a combination of different types of objects that incorporate both data and behaviour.

object-oriented programming (OOPs) is a methodology that simplifies software development and maintenance by providing some rules.

Basic concepts of OOPs are:

1. **Object**
2. **Class**
3. **Inheritance**
4. **Polymorphism**
5. **Abstraction**
6. **Encapsulation**

Platform Independent:

Java is platform independent because it is different from other languages like C, C++, 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.

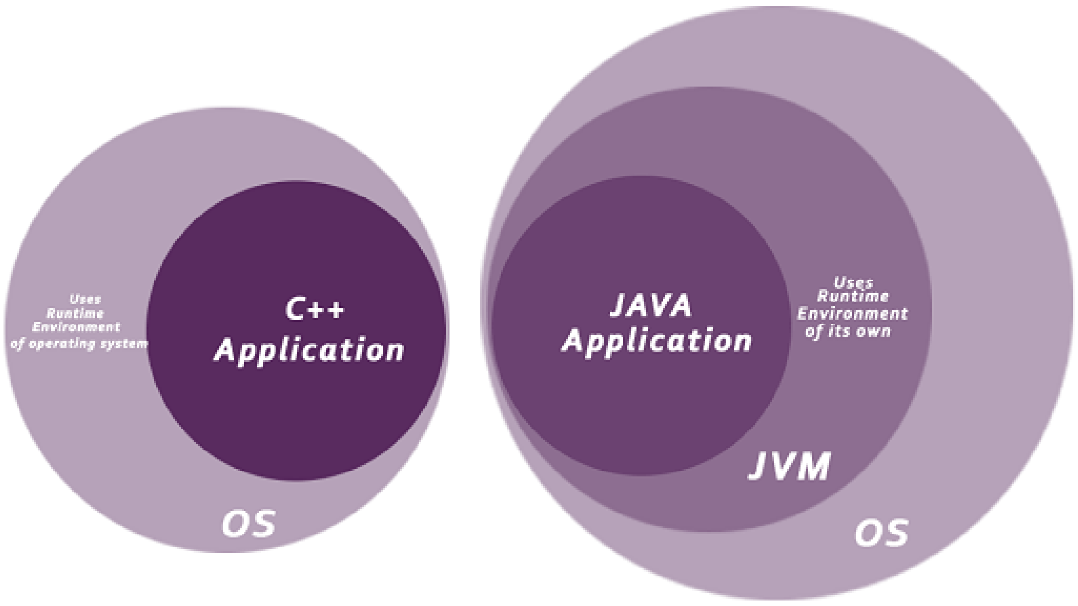
Example:

Windows, Unix. Linux….

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**


Uses Runtime Environment of operating system
C++ Application
OS
JAVA Application
Uses Runtime Environment of its own
JVM
OS

- **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.

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.

Portable:

- Java is portable because it facilitates you to carry the Java bytecode to any platform.
- It doesn't require any implementation. Java programs can be easily moved from one system to another system, anywhere and anytime.
- We can change and upgrade system processor and system resources without any changes in java.

High-performance:

- Java is faster than other traditional interpreted programming languages because Java bytecode is "close" to native code.
- It is still a little bit slower than a compiled language (e.g., C++). Java is an interpreted language that is why it is slower than compiled languages, e.g., C, C++, etc.

Distributed:

- Java designed as a distributed language for creating applications on network.
- It has ability to sharw both data and programs. Java applications can open and access from remote objects easily and can do in local systems.
- Java is distributed because it facilitates users to create distributed applications in Java.
- RMI and EJB 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:

- Multithread means it can handle multiple tasks simultaneously,
- java support the multithreaded programs, no need to wait until the previous task is completed or not
- 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++.

Introduction:

The Java Development Kit (**JDK**) is software used for Java programming, along with the Java Virtual Machine (**JVM**) and the Java Runtime Environment (**JRE**). The JDK includes the compiler and class libraries, allowing developers to create Java programs executable by the JVM and JRE.

Check if Java Is Installed:

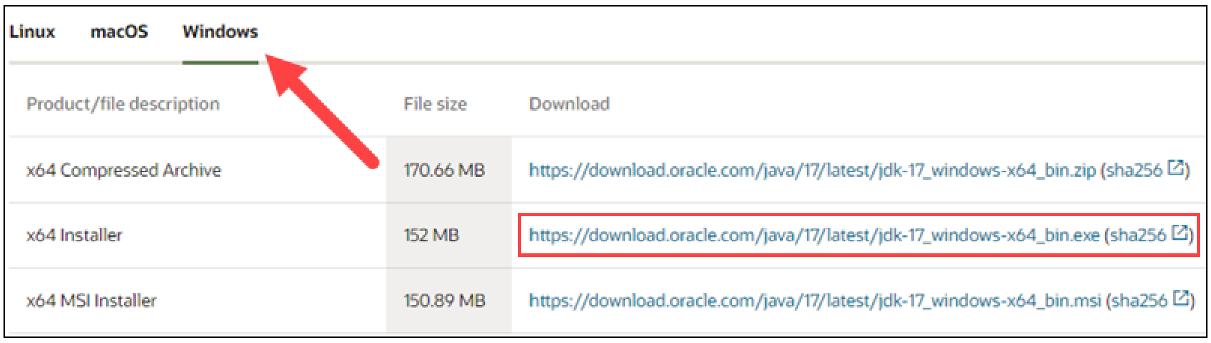
Before installing the Java Development Kit, check if a Java version is already installed on Windows. Follow the steps below:

1. Open a command prompt by typing *cmd* in the search bar and press **Enter**.
2. Run the following command:

Download Java for Windows 10

Download the latest Java Development Kit installation file for Windows 10 to have the latest features and bug fixes.

1. Using your preferred web browser, navigate to the Oracle Java Downloads page.
2. On the *Downloads* page, click the **x64 Installer** download link under the **Windows** category. At the time of writing this article, Java version 17 is the latest long-term support Java version.



Product/file description	File size	Download
x64 Compressed Archive	170.66 MB	https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.zip (sha256)
x64 Installer	152 MB	https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.exe (sha256)
x64 MSI Installer	150.89 MB	https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.msi (sha256)

Install Java on Windows 10

After downloading the installation file, proceed with installing Java on your Windows system.

Follow the steps below:

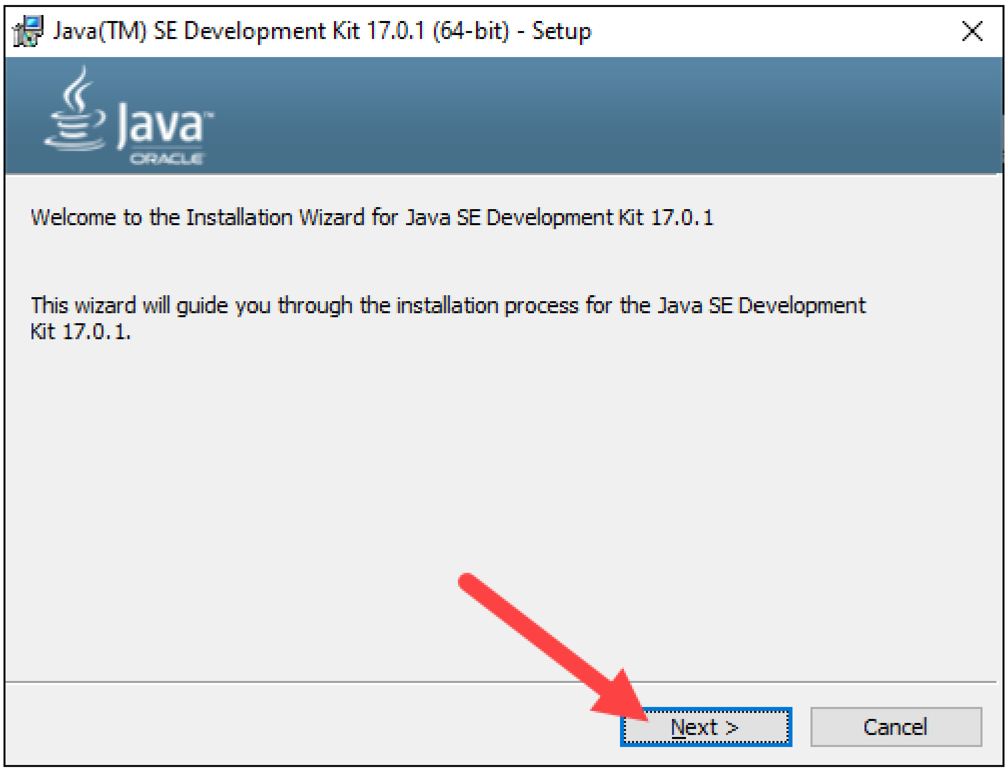
Step 1: Run the Downloaded File

Double-click the **downloaded file** to start the installation.

Step 2: Configure the Installation Wizard

After running the installation file, the installation wizard welcome screen appears.

1. Click **Next** to proceed to the next step.


Java(TM) SE Development Kit 17.0.1 (64-bit) - Setup

Welcome to the Installation Wizard for Java SE Development Kit 17.0.1

This wizard will guide you through the installation process for the Java SE Development Kit 17.0.1.

How to set the Temporary Path of JDK in Windows

To set the temporary path of JDK, you need to follow the following steps:

- Open the command prompt
- Copy the path of the JDK/bin directory
- Write in command prompt: set path=copied_path

For Example:

set path=C:\Program Files\Java\jdk1.6.0_23\bin

Compile: javac filename.java

Ex: javac SampleOne.java

Run: java filename

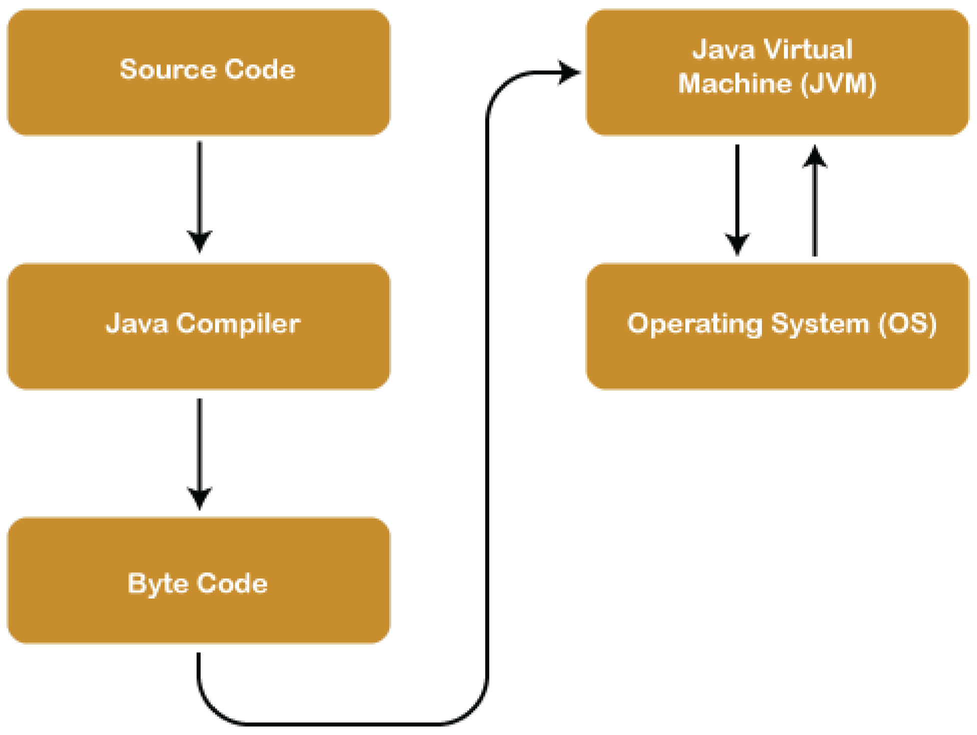
Ex: java SampleOne

Java Architecture:

Java Architecture is a collection of components, i.e., **JVM, JRE,** and **JDK**. **It** integrates the process of interpretation and compilation. It defines all the processes involved in creating a Java program. **Java Architecture** explains each and every step of how a program is compiled and executed.

Java Architecture can be explained by using the following steps:

- There is a process of compilation and interpretation in Java.
- Java compiler converts the Java code into byte code.
- After that, the JVM converts the byte code into machine code.
- The machine code is then executed by the machine.



Components of Java Architecture:

The Java architecture includes the three main components:

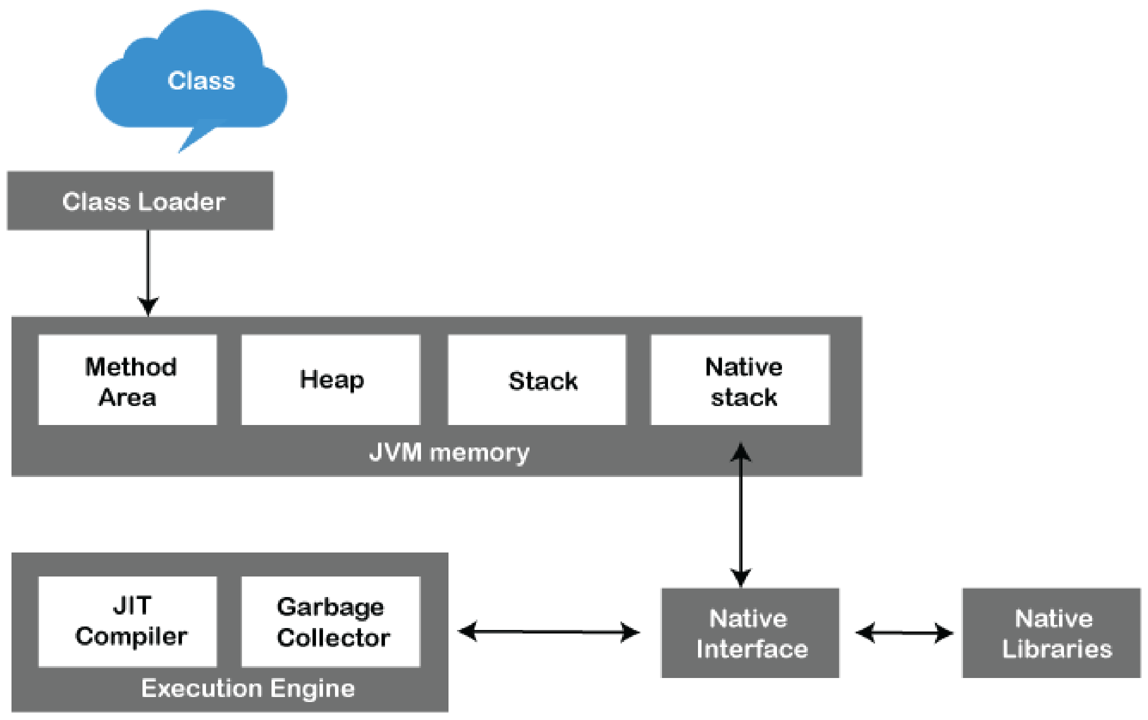
- Java Virtual Machine (JVM)
- Java Runtime Environment (JRE)
- Java Development Kit (JDK)

Java Virtual Machine: (JVM)

- The main feature of Java is **WORA**.
- WORA stands for **Write Once Run Anywhere**.
- The feature states that we can write our code once and use it anywhere or on any operating system.
- Our Java program can run any of the platforms only because of the Java Virtual Machine.
- It is a Java platform component that gives us an environment to execute java programs.
- JVM's main task is to convert byte code into machine code.
- JVM, first of all, loads the code into memory and verifies it.
- After that, it executes the code and provides a runtime environment.
- Java Virtual Machine (JVM) has its own architecture, which is given below:

JVM Architecture:

JVM is an abstract machine that provides the environment in which Java bytecode is executed. The falling figure represents the architecture of the JVM.



ClassLoader:

ClassLoader is a subsystem used to load class files. Class Loader first loads the Java code whenever we run it.

Class Method Area:

In the memory, there is an area where the class data is stored during the code's execution. Class method area holds the information of static variables, static methods, static blocks, and instance methods.

Heap:

The heap area is a part of the JVM memory and is created when the JVM starts up. Its size cannot be static because it increases or decrease during the application runs.

Stack:

It is also referred to as thread stack. It is created for a single execution thread. The thread uses this area to store the elements like the partial result, local variable, data used for calling method and returns etc.

Native Stack:

It contains the information of all the native methods used in our application.

Execution Engine:

It is the central part of the JVM. Its main task is to execute the byte code and execute the Java classes. The execution engine has three main components used for executing Java classes.

Interpreter:

It converts the byte code into native code and executes. It sequentially executes the code. The interpreter interprets continuously and even the same method multiple times. This reduces the performance of the system, and to solve this, the JIT compiler is introduced.

JIT Compiler:

JIT compiler is introduced to remove the drawback of the interpreter. It increases the speed of execution and improves performance.

Garbage Collector:

The garbage collector is used to manage the memory, and it is a program written in Java. It works in two phases, i.e., **Mark** and **Sweep**. Mark is an area where the garbage collector identifies the used and unused chunks of memory. The Sweep removes the identified object from the **Mark**

Java Native Interface:

Java Native Interface works as a mediator between Java method calls and native libraries.

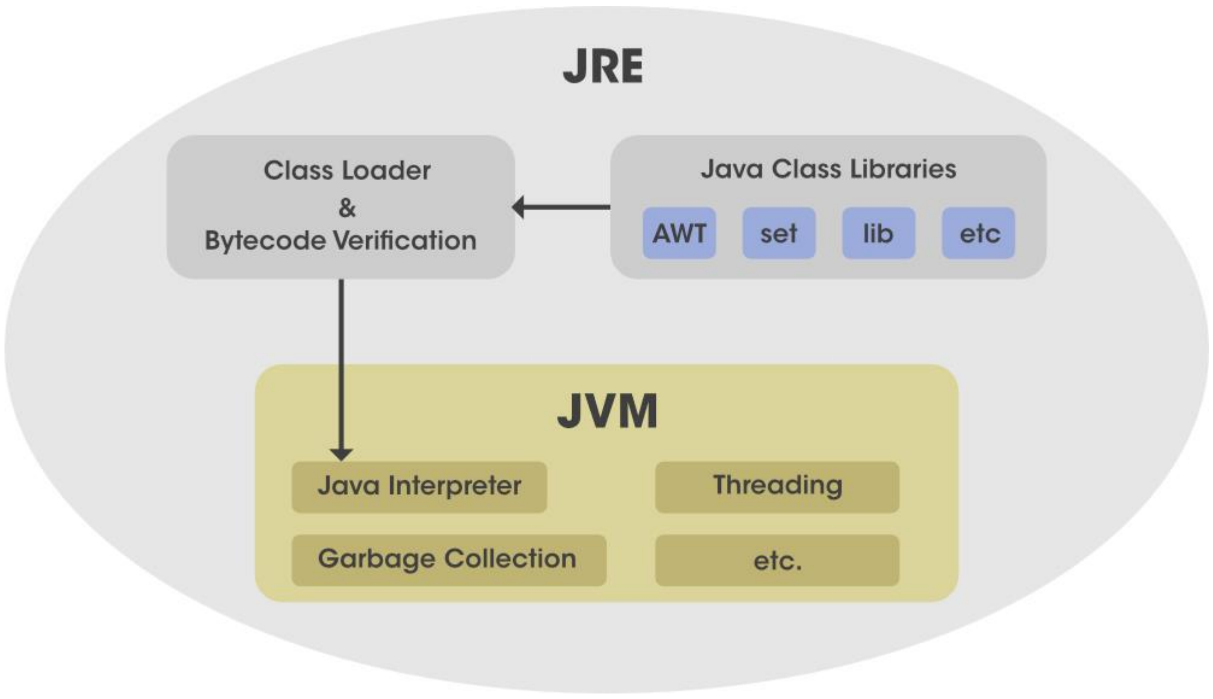
Java Runtime Environment:(JRE)

It provides an environment in which Java programs are executed. JRE takes our Java code, integrates it with the required libraries, and then starts the JVM to execute it. To learn more about the Java Runtime Environment.

What is a Java runtime environment (JRE)

A Java runtime environment (JRE) is a set of components to create and run a Java application. A JRE is part of a Java development kit (JDK).

A JRE is made up of a Java virtual machine (JVM), Java class libraries, and the Java class loader. JDKs are used to develop Java software; JREs provide programming tools and deployment technologies; and JVMs execute Java programs.



1.Class Loader:

At this step, the class loader loads various classes which are essential for running the program. The class loader dynamically loads the classes in the Java Virtual Machine.

When the JVM is started, three class loaders are used:

1. Bootstrap class loader
2. Extensions class loader
3. System class loader

2.Byte code verifier:

Byte code verifier can be considered as a gatekeeper. It verifies the bytecode so that the code doesn't make any sort of disturbance for the interpreter. The code is allowed to be interpreted only when it passes the tests of the Bytecode verifier which checks the format and checks for illegal code.

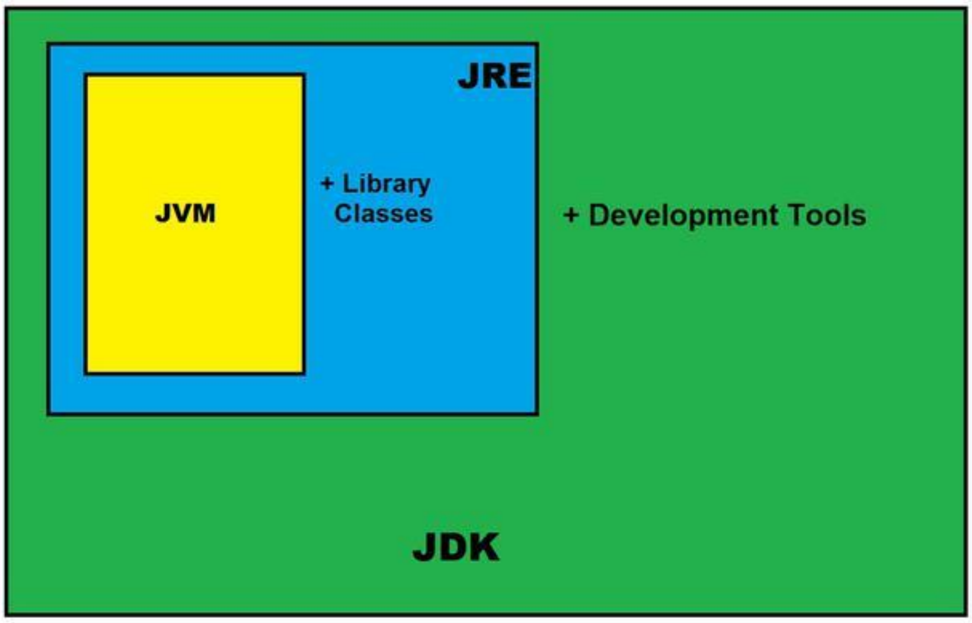
3.Interpreter:

Once the classes get loaded and the code gets verified, then interpreter reads the assembly code line by line and does the following two functions:

- Execute the Byte Code
- Make appropriate calls to the underlying hardware

Java Development Kit:

- Java development Kit is a bundle of software development tools and supporting libraries and combination of JRE and JVM.
- It is a software development environment used in the development of Java applications and applets.
- Java Development Kit holds JRE, a compiler, an interpreter or loader, and several development tools in it.



The Java Development Kit is an implementation of one of the Java Platform:

- Standard Edition Java Platform
- Enterprise Edition Java Platform
- Micro Edition Java Platform

JDK contains:

- Java Runtime Environment (JRE),
- An interpreter/loader (Java),
- A compiler (javac),
- An archiver (jar) and many more. (java Archive it is file format Ex:Zip)

JDK:

The Java Development Kit (JDK) is a software development environment which is used to develop java applications and applets.

1. JDK Alpha and Beta (1995)
2. JDK 1.0 (23rd Jan 1996)
3. JDK 1.1 (19th Feb 1997)
4. J2SE 1.2 (8th Dec 1998)
5. J2SE 1.3 (8th May 2000)
6. J2SE 1.4 (6th Feb 2002)
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).

Popular JDKs:

- **Oracle JDK:** the most popular JDK and the main distributor of Java11,
- **OpenJDK:** Ready for use: JDK 15, JDK 14, and JMC,
- **Azul Systems Zing:** efficient and low latency JDK for Linux os,
- **Azul Systems:** based Zulu brand for Linux, Windows, Mac OS X,
- **IBM J9 JDK:** for AIX, Linux, Windows, and many other OS,
- **Amazon Corretto:** the newest option with the no-cost build of OpenJDK and long-term support.

Java Garbage Collector:

Garbage means unreferenced objects.

Garbage Collection is process of reclaiming the runtime unused memory automatically. In other words, it is a way to destroy the unused objects.

- Java manage the Memory Automatically.
- The Garbage feature not available in C,C++
- Garbage collector is a part of the JVM.
- Garbage collector automatically destroy the object if it is no longer used.
- JVM contains two types of memories Stack and Heap.
- Stack memory is a faster than Heap memory.
- Heap memory is a dynamic
- Object variables are created in Heap memory

  Ex: int a=10;

- When object is unreferenced then the JVM send into garbage collector to destroy the object.

we were using free() function in C language and delete() in C++. But, in java it is performed automatically. So, java provides better memory management.

Advantage of Garbage Collection:

- It makes java **memory efficient** because garbage collector removes the unreferenced objects from heap memory.
- It is **automatically done** by the garbage collector(a part of JVM) so we don't need to make extra efforts.

finalize() method:

The finalize() method is invoked each time before the object is garbage collected. This method can be used to perform cleanup processing.

Syntax:

```
protected void finalize()

{

……………….

……………….

}
```

gc() method:

The gc() method is used to invoke the garbage collector to perform cleanup processing. The gc() is found in System and Runtime classes.

Syntax:

```
public static void gc()

{

………………

………………

}
```

Example:

```
class TestGarbage1
{
public void finalize()
{
System.out.println("object is garbage collected");
}
public static void main(String args[])
{
TestGarbage1 s1=new TestGarbage1();
TestGarbage1 s2=new TestGarbage1();
s1=null;
s2=null;
System.gc();
}
}
```

Output:

```
object is garbage collected

object is garbage collected
```

Lexical Tokens/ Lexical Issues in java

A lexical token may consist of one or more characters, and every single character is in exactly one token.

The tokens can be keywords, comments, numbers, white space, or strings. All lines should be terminated by a semi-colon (;).

- Verilog HDL is a case-sensitive language.
- And all keywords are in lowercase.

White Space

White space can contain the characters for tabs, blanks, newlines, and form feeds. These characters are ignored except when they serve to separate other tokens. However, blanks and tabs are significant in strings.

Comments

There are two types to represent the comments, such as:

1. Single line comments begin with the token // and end with a carriage return. For example, //this is the single-line syntax.
2. Multi-Line comments begin with the token /* and end with the token */ For example, /* this is multiline syntax*/

Numbers:

We can specify constant numbers in binary, decimal, hexadecimal, or octal format. Negative numbers are represented in 2's complement form. The question mark (?) character is the Verilog alternative for the z character when used in a number. The underscore character (_) is legal anywhere in a number, but it is ignored as the first character.

Integer Number

Real Numbers

Signed and Unsigned Numbers

Negative Numbers

Identifiers:

The identifier is the name used to define the object, such as a function, module, or register. Identifiers should begin with alphabetical characters or underscore characters.

For example, A_Z and a_z.

Identifiers are a combination of alphabetic, numeric, underscore, and $ characters. They can be up to 1024 characters long.

- Identifiers must begin with an alphabetic character or the underscore character (**a-z A-Z_**).
- Identifiers may contain alphabetic characters, numeric characters, the underscore, and the dollar sign (**a-z A-Z 0-9 _ $**).
- Identifiers can be up to 1024 characters long.

Escaped Identifiers:

Verilog HDL allows any character to be used in an identifier by escaping the identifier.

Escaped identifiers are including any of the printable ASCII characters in an identifier.

- The decimal values 33 through 126, or 21 through 7E in hexadecimal.

- Escaped identifiers begin with the backslash (\). The backslash escapes the entire identifier.

- The escaped identifier is terminated by white space characters such as commas, parentheses, and semicolons become part of the escaped identifier unless preceded by white space.
- Terminate escaped identifiers with white space. Otherwise, characters that should follow the identifier are considered part of it.

Operators:

Operators are special characters used to put conditions or to operate the variables. There are one, two, and sometimes three characters used to perform operations on variables.

1. Arithmetic Operators:

These operators perform arithmetic operations. The + and -are used as either unary (x) or binary (z-y) operators.

The operators included in arithmetic operation are addition, subtraction, multiplication, division, and modulus.

2. Relational Operators:

These operators compare two operands and return the result in a single bit, 1 or 0. The Operators included in relational operation are:

- == (equal to)
- != (not equal to)
- (greater than)
- >= (greater than or equal to)
- < (less than)
- <= (less than or equal to)

3. Bit-wise Operators:

Bit-wise operators do a bit-by-bit comparison between two operands. The Operators included in Bit-wise operation are:

- & (Bit-wise AND)
- | (Bit-wiseOR)
- ~ (Bit-wise NOT)
- ^ (Bit-wise XOR)
- ~^ or ^~(Bit-wise XNOR)

4. Logical Operators:

Logical operators are bit-wise operators and are used only for single-bit operands. They return a single bit value, 0 or 1. They can work on integers or groups of bits, expressions and treat all non-zero values as 1.

Logical operators are generally used in conditional statements since they work with expressions. The operators included in Logical operation are:

- ! (logical NOT)
- && (logical AND)
- || (logical OR)

5. Reduction Operators:

Reduction operators are the unary form of the bitwise operators and operate on all the bits of an operand vector. These also return a single-bit value. The operators included in Reduction operation are:

- & (reduction AND)
- | (reduction OR)
- ~& (reduction NAND)
- ~| (reduction NOR)
- ^ (reduction XOR)
- ~^ or ^~(reduction XNOR)

6. Shift Operators:

Shift operators are shifting the first operand by the number of bits specified by the second operand in the syntax.

Vacant positions are filled with zeros for both directions, left and right shifts (There is no use sign extension). The Operators included in Shift operation are:

- << (shift left)
- >> (shift right)

7. Concatenation Operator:

The concatenation operator combines two or more operands to form a larger vector. The operator included in Concatenation operation is:

- { }(concatenation)

8. Replication Operator:

The replication operator is making multiple copies of an item. The operator used in Replication operation is:

- {n{item}} (n fold replication of an item)

9. Conditional Operator:

Conditional operator synthesizes to a multiplexer. It is the same kind as is used in C/C++ and evaluates one of the two expressions based on the condition. The operator used in Conditional operation is:

- (Condition) ?

Operands:

Operands are expressions or values on which an ***operator*** operates or works. All expressions have at least one operand.

1. Literals

Literals are constant-valued operands that are used in Verilog expressions. The two commonly used Verilog literals are:

- **String**: A literal string operand is a one-dimensional array of characters enclosed in double quotes (" ").
- **Numeric**: A constant number of the operand is specified in binary, octal, decimal, or hexadecimal number.

2. Wires, Regs, and Parameters:

Wires, regs, and parameters are the data types used as operands in Verilog expressions. Bit-Selection "x[2]" and Part-Selection "x[4:2]"

Bit-selects and **part-selects** are used to select one bit and multiple bits, respectively, from a wire, regs or parameter vector using square brackets "[ ]".

3. Function Calls:

In the Function calls, the return value of a function is used directly in an expression without first assigning it to a register or wire.

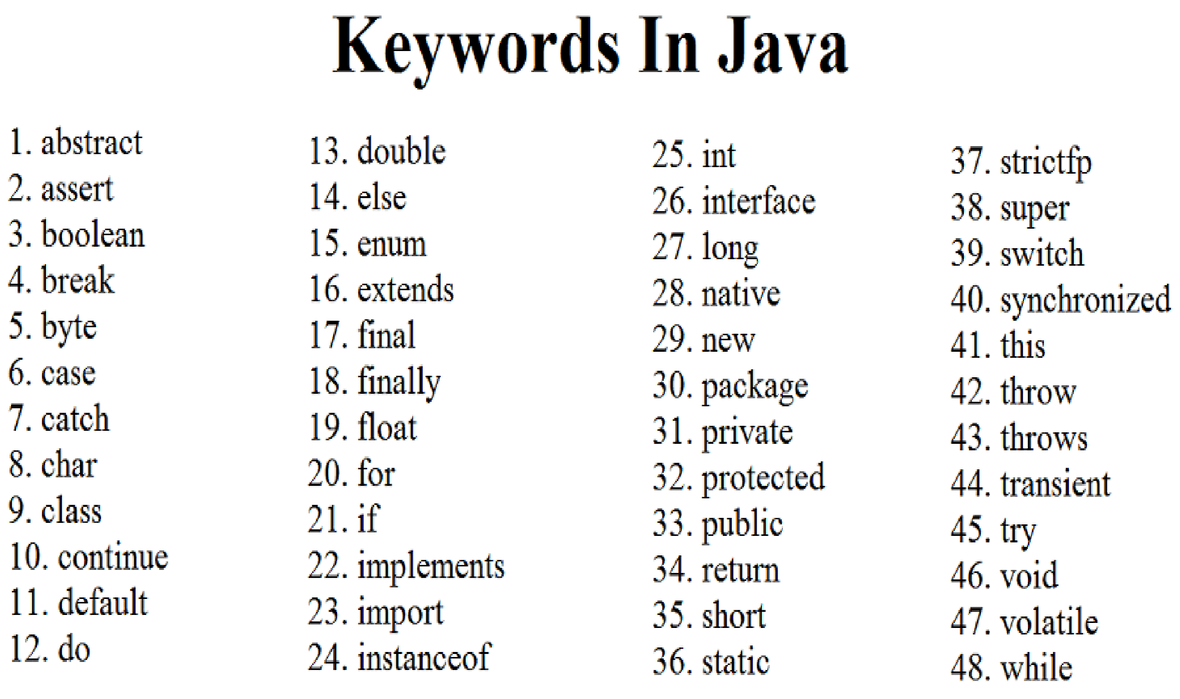
It just places the function call as one of the types of operands. It is useful to know the bit width of the return value of the function call.

Keywords

Qns:

What is a keyword and explain different types of keywords in java

Keywords or Reserved words are the words in a language that are used for some internal process or represent some predefined actions. These words are therefore not allowed to use as variable names or objects.



Keywords In Java

1. abstract	13. double	25. int	37. strictfp
2. assert	14. else	26. interface	38. super
3. boolean	15. enum	27. long	39. switch
4. break	16. extends	28. native	40. synchronized
5. byte	17. final	29. new	41. this
6. case	18. finally	30. package	42. throw
7. catch	19. float	31. private	43. throws
8. char	20. for	32. protected	44. transient
9. class	21. if	33. public	45. try
10. continue	22. implements	34. return	46. void
11. default	23. import	35. short	47. volatile
12. do	24. instanceof	36. static	48. while

1. **abstract:**

   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:**

   Java boolean keyword is used to declare a variable as a boolean type. It can hold True and False values only.

3. **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:**

   Java byte keyword is used to declare a variable that can hold 8-bit data values.

5. **case:**

   Java case keyword is used with the switch statements to mark blocks of text.

6. **catch:**

   Java catch keyword is used to catch the exceptions generated by try statements. It must be used after the try block only.

7. **char:**

   Java char keyword is used to declare a variable that can hold unsigned 16-bit Unicode characters

8. **class:**

   Java class keyword is used to declare a class.

9. **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:**

    Java default keyword is used to specify the default block of code in a switch statement.

11. **do**:

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**:

Java double keyword is used to declare a variable that can hold 64-bit floating-point number.

13. **else**:

Java else keyword is used to indicate the alternative branches in an if statement.

14. **enum**:

Java enum keyword is used to define a fixed set of constants. Enum constructors are always private or default.

15. **extends**:

Java extends keyword is used to indicate that a class is derived from another class or interface.

16. **final**:

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**:

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**:

Java float keyword is used to declare a variable that can hold a 32-bit floating-point number.

19.**for:**

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:**

Java if keyword tests the condition. It executes the if block if the condition is true.

21.**implements:**

Java implements keyword is used to implement an interface.

22.**import:**

Java import keyword makes classes and interfaces available and accessible to the current source code.

23.**instanceof:**

Java instanceof keyword is used to test whether the object is an instance of the specified class or implements an interface.

24.**int:**

Java int keyword is used to declare a variable that can hold a 32-bit signed integer.

25.**interface:**

Java interface keyword is used to declare an interface. It can have only abstract methods.

26.**long:**

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:**

Java new keyword is used to create new objects.

29.**null:**

Java null keyword is used to indicate that a reference does not refer to anything. It removes the garbage value.

30.**package:**

Java package keyword is used to declare a Java package that includes the classes.

31.**private:**

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:**

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:**

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:**

Java return keyword is used to return from a method when its execution is complete.

35.**short:**

Java short keyword is used to declare a variable that can hold a 16-bit integer.

36.**static:**

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:**

Java strictfp is used to restrict the floating-point calculations to ensure portability.

38.**super:**

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:**

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:**

Java synchronized keyword is used to specify the critical sections or methods in multithreaded code.

41.**this:**

Java this keyword can be used to refer the current object in a method or constructor.

42. **throw:**

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:**

The Java throws keyword is used to declare an exception. Checked exceptions can be propagated with throws.

44. **transient:**

Java transient keyword is used in serialization. If you define any data member as transient, it will not be serialized.

45. **try:**

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:**

Java volatile keyword is used to indicate that a variable may change asynchronously.

48. **while:**

Java while keyword is used to start a while loop. This loop iterates a part of the program several times. If the number of iterations is not fixed, it is recommended to use the while loop.

Sample program in Java:

In this section, we will learn how to write the simple program of Java. We can write a simple hello Java program easily.

```
class Simple
{
public static void main(String args[])
{
System.out.println("Hello Java");
}
}
```

Save the above file as Simple.java.

To compile:	javac Simple.java
To execute:	java Simple

Output:

```
Hello Java
```

- **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 doesn't require creating an object to invoke the main() method. So, it saves memory.
- **void** is the return type of the method. It means it doesn't return any value.
- **main** represents the starting point of the program.
- **String[] args** or **String args[]** is used for command line argument

- **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()**