Java: Java is a widely-used programming language for coding web application. It platform Any hardware or software environment in which a program runs, is known as a platform. Since Java has a runtime environment (JRE) and API, it is called a platform. Java have its own runtime environment to run program. Java is a high level, robust, object-oriented and secure programming language. Java program run on any operating system so it is platform independent

Types of java Application

) 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.

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), [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.

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, [EJB](https://www.javatpoint.com/ejb-tutorial) 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.

Core feature of java

Simple, object oriented, multithreded, platform independent, portable, secure, robust,

neutral. Dynamic, distributed, high performance Simple : Java is very easy to learn, and its syntax is simple, clean and easy to understand

Object Oriented : Java is an [object-oriented](https://www.javatpoint.com/java-oops-concepts) 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 behavior.

Multithreded : 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.

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

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#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)

* What happens in compile time : At compile time java file is compiled by java compiler and converts java code into bytecode
* **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 :** Bytecode in Java is a set of instructions for the Java Virtual Machine. Bytecode is a platform-independent code. Bytecode is a code that lies between low-level language and high-level language. After the Java code is compiled, the bytecode gets generated, which can be executed on any machine using JVM

**A compiler translates complete high-level programming code into machine code at once.** **An interpreter translates one statement of programming code at a time into machine code**

JVM(Java Virtual Machine) : JVM (Java Virtual Machine) is an abstract machine. It is a specification that provides runtime environment in which java bytecode can be executed.It is called virtual machine because it does no exit physically

1. **A specification** where working of Java Virtual Machine is specified. But implementation provider is independent to choose the algorithm. Its implementation has been provided by Oracle and other companies.
2. **An implementation** Its implementation is known as JRE (Java Runtime Environment).

* Loads code
* Verifies code
* Executes code
* Provides runtime environment

JVM provides definitions for the:

* Memory area
* Class file format
* Register set
* Garbage-collected heap
* Fatal error reporting etc.

Q . What is JDK? : JDK is an acronym for Java Development Kit. The Java Development Kit (JDK) is a software development environment which is used to develop Java applications and [applets](https://www.javatpoint.com/java-applet). It physically exists. It contains JRE + development tools. The JDK contains a private Java Virtual Machine (JVM) and a few other resources such as an interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc), etc. to complete the development of a Java Application.

Q. What is JRE? JRE is an acronym for Java Runtime Environment. It is also written as Java RTE. The Java Runtime Environment is a set of software tools which are used for developing Java applications. It is used to provide the runtime environment. It is the implementation of JVM. It physically exists. It contains a set of libraries + other files that JVM uses at runtime

Q. What is keyword? : Java keywords are also known as reserved words. They have specific meaning . Keywords are particular words that act as a key to a code. These are predefined words by Java so they cannot be used as a variable or object name or class name

* Q. What is Just In Time compiler in java? It requires less memory usages.
* The code optimization is done at run time.
* It uses different levels of optimization.
* It reduces the page faults. Runtime Environment that improves the performance of Java applications at run time

### **8) What is the platform?**

A platform is the hardware or software environment in which a piece of software is executed. There are two types of platforms, software-based and hardware-based. Java provides the software-based platform.

### **17) What are the various access specifiers in Java?**

In Java, access specifiers are the keywords which are used to define the access scope of the method, class, or a variable. In Java, there are four access specifiers given below.

* **Public** The classes, methods, or variables which are defined as public, can be accessed by any class or method.
* **Protected** Protected can be accessed by the class of the same package, or by the sub-class of this class, or within the same class.
* **Default** Default are accessible within the package only. By default, all the classes, methods, and variables are of default scope.
* **Private** The private class, methods, or variables defined as private can be accessed within the class only.

Operator in java :

**Operator** in [Java](https://www.javatpoint.com/java-tutorial) is a symbol that is used to perform operations. Operators are **used to perform operations on variables and values**

* **Unary operator:** (++,--)The Java unary operators require only one operand. Unary operators are used to perform various operations i.e.: incrementing/decrementing a value by one
* negating an expression
* inverting the value of a boolean

**Ex int a =10:**

**Arithmatic Operator:** Java arithmetic operators are used to perform addition, subtraction, multiplication, and division. They act as basic mathematical operations

Relational Operator: Relational operators compare numeric, character string, or logical data. The result of the comparison, either true ( 1 ) or false ( 0 ), can be used to make a decision regarding program flow (see the IF statement).

## Logical Operator : 1. Logical ‘AND’ Operator (&&)

This operator returns true when both the conditions under consideration are satisfied or are true. If even one of the two yields false, the operator results false. In Simple terms, ***cond1 && cond2 returns true when both cond1 and cond2 are true (i.e. non-zero***).

## ****2. Logical ‘OR’ Operator (||)****

This operator returns true when one of the two conditions under consideration is satisfied or is true. If even one of the two yields true, the operator results true. To make the result false, both the constraints need to return false.

Ternary operator: Java Ternary operator is used as one line replacement for if-then-else statement and used a lot in Java programming. It is the only conditional operator which takes three operands.

Assignment operator: Java assignment operator is one of the most common operators. It is used to assign the value on its right to the operand on its left.

Control statement in Java :  [Java](https://www.javatpoint.com/java-tutorial) provides statements that can be used to control the flow of Java code. Such statements are called control flow statements. A **control statement in java** is a statement that determines whether the other statements will be executed or not. It controls the flow of a sttament

Java provides three types of control flow statements

1)decision making statement: : decision-making statements decide which statement to execute and when. Decision-making statements evaluate the Boolean expression and control the program flow depending upon the result of the condition provided.

a) simple if statement : "if" statement is used to evaluate a condition It is the most basic statement among all control flow statements in Java. It evaluates a Boolean expression and enables the program to enter a block of code if the expression evaluates to true.

Syntax:

i**f**(condition) {

statement 1; //executes when condition is true

}

Example :

**public** **class** Student {

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

**int** x = 10;

**int** y = 12;

**if**(x+y > 20) {

System.out.println("x + y is greater than 20");

}   }      }

output  : true

### **2) if-else statement**

The [if-else statement](https://www.javatpoint.com/java-if-else) is an extension to the if-statement, which uses another block of code, i.e., else block. The else block is executed if the condition of the if-block is evaluated as false.

## Using Ternary Operator

We can also use ternary operator (? :) to perform the task of if...else statement. It is a shorthand way to check the condition. If the condition is true, the result of ? is returned. But, if the condition is false, the result of : is returned.

**public** **class** IfElseTernaryExample {

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

**int** number=13;

    //Using ternary operator

    String output=(number%2==0)?"even number":"odd number";

    System.out.println(output);

}

}

Syntax:

**if**(condition) {

statement 1; //executes when condition is true

}

**else**{

statement 2; //executes when condition is false

}

Example:

**public** **class** Student {

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

**int** x = 10;

**int** y = 12;

**if**(x+y < 10) {

System.out.println("x + y is less than      10");

}   **else** {

System.out.println("x + y is greater than 20");

}}}

Output :   x + y is greater than 20

### **3) if-else-if ladder:**

The if-else-if statement contains the if-statement followed by multiple else-if statements. In other words, we can say that it is the chain of if-else statements that create a decision tree where the program may enter in the block of code where the condition is true. We can also define an else statement at the end of the chain.

Syntax:

**if**(condition 1) {

statement 1; //executes when condition 1 is true

}

**else** **if**(condition 2) {

statement 2; //executes when condition 2 is true

}

**else** {

statement 2; //executes when all the conditions are false

}

1. Example: **public** **class** Student {

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

String city = "Delhi";

**if**(city == "Meerut") {

System.out.println("city is meerut");

}**else** **if** (city == "Noida") {

System.out.println("city is noida");

}**else** **if**(city == "Agra") {

System.out.println("city is agra");

}**else** {

System.out.println(city);

}

}  }

nested if-statements, :

n nested if-statements, the if statement can contain a **if** or **if-else** statement inside another if or else-if statement.

Syntax:

**if**(condition 1) {

statement 1; //executes when condition 1 is true

**if**(condition 2) {

statement 2; //executes when condition 2 is true

}

**else**{

statement 2; //executes when condition 2 is false

}

}

Example : int age=22; //nested if

float weight =45.3f;

if(age>=18)

{

if (weight>=50)

{

System.out.println("Congratulation you are ready for blood donation");

}else {

System.out.println("Sorry your weight is less than 50kg");

}

}

else {

System.out.println("Your age is not equal or greater than 18");

}

Switch Statement : In Java, [Switch statements](https://www.javatpoint.com/java-switch) are similar to if-else-if statements. The switch statement contains multiple blocks of code called cases and a single case is executed based on the variable which is being switched. The switch statement is easier to use instead of if-else-if statements. It also enhances the readability of the program.

Points to be noted about switch statement:

* The case variables can be int, short, byte, char, or enumeration. String type is also supported since version 7 of Java
* Cases cannot be duplicate
* Default statement is executed when any of the case doesn't match the value of expression. It is optional.
* Break statement terminates the switch block when the condition is satisfied.  
  It is optional, if not used, next case is executed.
* While using switch statements, we must notice that the case expression will be of the same type as the variable. However, it will also be a constant value.

Syntax: **switch** (expression){

**case** value1:

     statement1;

**break**;

    .

    .

    .

**case** valueN:

     statementN;

**break**;

**default**:

**default** statement;

}

Example: String s="green";

**switch** (s) {

**case** "red" :

System.***out***.println("given input is");

**break**;

**case** "blue" :

System.***out***.println("given input is");

**break**;

**case** "green" :

System.***out***.println("given input is green");

**break**;

**case** "black" :

System.***out***.println("given input is");

**break**;

**default**:

System.***out***.println("Nothing is match");

}

2)Loop statement : loop statements are used to execute the set of instructions in a repeated order. The execution of the set of instructions depends upon a particular condition.

a) for loop : The Java for loop is used to iterate a part of the program several times. If the number of iteration is **fixed**, it is recommended to use for loop.  We can initialize the [variable](https://www.javatpoint.com/java-variables), check condition and increment/decrement value. It consists of four parts:

Syntax: for(initialization;condition;increment/decrement){

//statement or code to be executed   }

Example : //Java Program to demonstrate the example of for loop

//which prints table of 1

**public** **class** ForExample {

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

    //Code of Java for loop

**for**(**int** i=1;i<=10;i++){

        System.out.println(i);

    }

}

}

Output: 1,2,3,4,5,6,7,8,9,10

While Loop: The [Java](https://www.javatpoint.com/java-tutorial) while loop is used to iterate a part of the [program](https://www.javatpoint.com/programs-list) repeatedly until the specified Boolean condition is true. As soon as the Boolean condition becomes false, the loop automatically stops.

The while loop is considered as a repeating if statement. If the number of iteration is not fixed, it is recommended to use the while [loop](https://www.javatpoint.com/java-for-loop).

// Java program to illustrate while loop.

class whileLoopDemo {

    public static void main(String args[])

    {

        // initialization expression

        int i = 1;

        // test expression

        while (i < 6) {

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

            // update expression

            i++;

        }

    }

}

Explanation or logic of above program

1. Program starts.

2. i is initialized with value 1.

3. Condition is checked. 1 < 6 yields true.

3.a) "Hello World" gets printed 1st time.

3.b) Updation is done. Now i = 2.

4. Condition is checked. 2 < 6 yields true.

4.a) "Hello World" gets printed 2nd time.

4.b) Updation is done. Now i = 3.

5. Condition is checked. 3 < 6 yields true.

5.a) "Hello World" gets printed 3rd time

5.b) Updation is done. Now i = 4.

6. Condition is checked. 4 < 6 yields true.

6.a) "Hello World" gets printed 4th time

6.b) Updation is done. Now i = 5.

7. Condition is checked. 5 < 6 yields true.

7.a) "Hello World" gets printed 5th time

7.b) Updation is done. Now i = 6.

8. Condition is checked. 6 < 6 yields false.

9. Flow goes outside the loop. Program terminates.