JAVA

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

Java can be used to create complete applications that can run on a single computer or be distributed across servers and clients in a network. As a result, you can use it to easily build mobile applications or run on desktop applications that use different operating systems and servers, such as Linux or Windows.

One of the most significant advantages of Java is its **ability to move easily from one computer system to another**. The ability to run the same program on many different systems is crucial to World Wide Web software, and Java succeeds at this by being platform-independent at both the source and binary levels

jvm

Ex: class Simple{

    public static void main(String args[]){

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

    }

}

Java Platforms:

Java SE- Standard Edition

Java EE- Enterprise Edition

Java ME- Micro Edition

Features:

The primary objective of [Java programming](https://www.javatpoint.com/java-tutorial) 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*.



OOPs (Object-Oriented Programming System)

**Object** means a real-world entity such as a pen, chair, table, computer, watch, etc. **Object-Oriented Programming** is a methodology or paradigm to design a program using classes and objects. It simplifies software development and maintenance by providing some concepts:

* [Object](https://www.javatpoint.com/object-and-class-in-java)
* Class
* [Inheritance](https://www.javatpoint.com/inheritance-in-java)
* [Polymorphism](https://www.javatpoint.com/runtime-polymorphism-in-java)
* [Abstraction](https://www.javatpoint.com/abstract-class-in-java)
* [Encapsulation](https://www.javatpoint.com/encapsulation)



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.

Variable is a name of memory location. There are three types of variables in java: local, instance and static.

Types of Variables:

1. Local
2. Instance
3. Static

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

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

#### 3) Static variable

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

Ex: class A{

int data=50;//instance variable

static int m=100;//static variable

void method(){

int n=90;//local variable

}

}//end of class

Ex: class Simple{   //to add 2 numbers

public static void main(String[] args){

int a=10;

int b=10;

int c=a+b;

System.out.println(c);

}}

Types of data-types:

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).
3. Primitive:

There are 8 types of primitive data types:

* boolean data type- 1 bit
* byte data type- 1 byte
* char data type-2 bytes
* short data type- 2 bytes
* int data type- 4
* long data type- 8
* float data type- 4
* double data type- 8

Boolean: The Boolean data type is used to store only two possible values: true and false. This data type is used for simple flags that track true/false conditions.

Ex: boolean x= false;

Byte: It stores 1 byte of data.

Ex: byte a=10;

Short: It stores 2 bytes of data

Ex: short a= 10000;

Integer: It stores 4 bytes of data

Ex: int a= 100000;

Long: It stores 8 bytes of data

Ex: long b= 100000L;

Float: It stores 4 bytes of data. Stores fractional numbers and is sufficient for storing 6 to 7 decimal digits

Ex: float f1= 234.5f;

Double: It stores 8 bytes of data. Stores fractional numbers and is sufficient for storing 15 decimal digits

Ex: double d1= 12.78839;

Char: It stores 2 bytes of data. It stores a single character or ASCII values.

Ex: char c= ‘@’;

Operators:

Operator in [Java](https://www.javatpoint.com/java-tutorial) is a symbol which is used to perform operations.

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

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Operators in Java: Operator in [Java](https://www.javatpoint.com/java-tutorial) is a symbol which 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 | = += -= \*= /= %= &= ^= |= <<= >>= >>>= |   Control Statements:  The [Java](https://www.javatpoint.com/java-tutorial) *if statement* is used to test the condition. It checks [boolean](https://www.javatpoint.com/boolean-keyword-in-java) condition: *true* or *false*. There are various types of if statement in Java.   1. If-else 2. If-else-if ladder |

If-else:

Ex: public class IfElseExample {

public static void main(String[] args) {

    //defining a variable

    int number=13;

    //Check if the number is divisible by 2 or not

    if(number%2==0){

        System.out.println("even number");

    }

else{

        System.out.println("odd number");

    }

}

}

If-else-if:

Ex: if(condition1){

//code to be executed if condition1 is true

}else if(condition2){

//code to be executed if condition2 is true

}

else if(condition3){

//code to be executed if condition3 is true

}

...

else{

//code to be executed if all the conditions are false

}

Ex: public class IfElseIfExample {

public static void main(String[] args) {

    int marks=65;

    if(marks<50){

        System.out.println("fail");

    }

    else if(marks>=50 && marks<60){

        System.out.println("D grade");

    }

    else if(marks>=60 && marks<70){

        System.out.println("C grade");

    }

    else if(marks>=70 && marks<80){

        System.out.println("B grade");

    }

    else if(marks>=80 && marks<90){

        System.out.println("A grade");

    }else if(marks>=90 && marks<100){

        System.out.println("A+ grade");

    }else{

        System.out.println("Invalid!");

    }  }  }

Switch Statement:

The Java *switch statement* executes one statement from multiple conditions. It is like [if-else-if](https://www.javatpoint.com/java-if-else) ladder statement.

Syntax:

switch(expression){

case 1:

 //code to be executed;

 break;  //optional

case value2:

 //code to be executed;

 break;  //optional

......

default:

 code to be executed if all cases are not matched;

}

Ex: public class SwitchExample {

public static void main(String[] args) {

    //Declaring a variable for switch expression

    int number=20;

    //Switch expression

    switch(number){

    //Case statements

    case 10: System.out.println("10");

    break;

   case 20: System.out.println("20");

   break;

   case 30: System.out.println("30");

   break;

    //Default case statement

    default: System.out.println("Not in 10, 20 or 30");

    }  } }

Ex: public class SwitchMonthExample {

public static void main(String[] args) {

    //Specifying month number

    int month=7;

    String monthString="";

    //Switch statement

    switch(month){

    //case statements within the switch block

    case 1: monthString="1 - January";

    break;

    case 2: monthString="2 - February";

    break;

    case 3: monthString="3 - March";

  break;

    case 4: monthString="4 - April";

    break;

    case 5: monthString="5 - May";

    break;

    case 6: monthString="6 - June";

    break;

    case 7: monthString="7 - July";

    break;

    case 8: monthString="8 - August";

    break;

    case 9: monthString="9 - September";

    break;

    case 10: monthString="10 - October";

    break;

    case 11: monthString="11 - November";

    break;

    case 12: monthString="12 - December";

    break;

    default:System.out.println("Invalid Month!");

    }

    //Printing month of the given number

    System.out.println(monthString);

}    }

Program to check if an alphabet is a vowel or consonant: a,f,o, E, i, x, z

package com.functions;

import java.util.Scanner;

\* what is function or method?

function is the one to reduce the complexity of the program.

Breaking the program into modules by giving specific names is called function or method.

EXAMPLE:

int sum() {

int a,b,c;

a=44;

b=33;

c=a+b;

return c; }

Types of function?

ans)There are two types of functions.

a)Non parameterized function

b)Parameterized function

--->Note: Function must be declared outside the main method or function

public class FunctionsExample

{

int x,y;

//Non parameterized function

void sum() //void is a return type and sum is the function name

{

int a,b,c;

a=33;

b=55;

c=a+b;

System.out.println(c); }

//Non Parmeretized function

void sub() //void means the function doesn’t have to return a value

{

int x,y,z;

x=1000;

y=2000;

z=x-y;

System.out.println(z); }

//Parameterized function

int sum(int f,int g)//method signature

{

return f+g;

}

public static void main(String[] args) //main method or function

{

//If anything is outside main method we must have to access or call them

//To access or call them we must need to create an object

int r1,r2;

FunctionsExample fone=new FunctionsExample();

//FunctionsExample ftwo=new FunctionsExample(); //Creating an object for FunctionsExample

fone.sum();//calling a function

fone.sub();//calling a function

//-----give an input to read

System.out.println("Enter the values here");

Scanner s1=new Scanner(System.in);

r1=s1.nextInt();

r2=s1.nextInt();

int foo= fone.sum(r1,r2);//calling a parameterized function

System.out.println(foo);

System.out.println(fone.sum(22,33));

} }

Looping Statements:

Loops are used to execute a set of instructions/functions repeatedly when some conditions become true. There are three types of loops in Java.

* for loop
* [while loop](https://www.javatpoint.com/java-while-loop)
* [do-while loop](https://www.javatpoint.com/java-do-while-loop)

|  |  |  |  |
| --- | --- | --- | --- |
| **Comparison** | **for loop** | **while loop** | **do while loop** |
| Introduction | The Java for loop is a control flow statement that iterates a part of the [programs](https://www.javatpoint.com/java-programs) multiple times. | The Java while loop is a control flow statement that executes a part of the programs repeatedly on the basis of given boolean condition. | The Java do while loop is a control flow statement that executes a part of the programs at least once and the further execution depends upon the given boolean condition. |
| When to use | If the number of iteration is fixed, it is recommended to use for loop. | If the number of iteration is not fixed, it is recommended to use while loop. | If the number of iteration is not fixed and you must have to execute the loop at least once, it is recommended to use the do-while loop. |
| Syntax | for(init;condition;incr/decr){  // code to be executed  } | while(condition){  //code to be executed  } | do{  //code to be executed  }while(condition); |
| Example | //for loop  for(int i=1;i<=10;i++){  System.out.println(i);  } | //while loop  int i=1;  while(i<=10){  System.out.println(i);  i++;  } | //do-while loop  int i=1;  do{  System.out.println(i);  i++;  }while(i<=10); |
| Syntax for infinitive loop | for(;;){  //code to be executed  } | while(true){  //code to be executed    } | do{  //code to be executed  }while(true); |

For loop:

for(initialization;condition;incr/decr){

//statement or code to be executed

}

Ex: 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);

    }  } }

Nested for:

Ex: public class NestedForExample {

public static void main(String[] args) {

//loop of i

for(int i=1;i<=3;i++){

//loop of j

for(int j=1;j<=3;j++){

        System.out.println(i+" "+j);  //1 1

}//end of i

}//end of j

}  }

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) several times. If the number of iteration is not fixed, it is recommended to use while [loop](https://www.javatpoint.com/java-for-loop).

while(condition){

//code to be executed

}

Ex: public class WhileExample {

public static void main(String[] args) {

    int i=1;

    while(i<=10){

        System.out.println(i);

    i++;

    }

}  }

Infinitive loop:

public class WhileExample2 {

public static void main(String[] args) {

    while(true){

        System.out.println("infinitive while loop");

    }  } }

Do-while Loop:

The Java *do-while loop* is used to iterate a part of the program several times. If the number of iteration is not fixed and you must have to execute the loop at least once, it is recommended to use do-while loop.

The Java *do-while loop* is executed at least once because condition is checked after loop body.

Syntax: do{

//code to be executed

}while(condition);

public class DoWhileExample {

public static void main(String[] args) {

    int i=1;

    do{

        System.out.println(i);

    i++;

    }while(i<=10);

}

}

Java Break Statement:

When a break statement is encountered inside a loop, the loop is immediately terminated and the program control resumes at the next statement following the loop.

The Java *break* statement is used to break loop or [switch](https://www.javatpoint.com/java-switch) statement. It breaks the current flow of the program at specified condition. In case of inner loop, it breaks only inner loop.

An Object can be defined as an instance of a class. An object contains an address and takes up some space in memory.

**Example:** A dog is an object because it has states like color, name, breed, etc. as well as behaviors like wagging the tail, barking, eating, etc.

An object has three characteristics:

* State: represents the data (value) of an object.
* Behavior: represents the behavior (functionality) of an object such as deposit, withdraw, etc.
* Identity: An object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. However, it is used internally by the JVM to identify each object uniquely.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*End\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Pyramid Example 1:**

**PyramidExample.java**

**public** **class** PyramidExample {

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

**for**(**int** i=1;i<=5;i++){

**for**(**int** j=1;j<=i;j++){

        System.out.print("\* ");

}

System.out.println();//new line

}

}

}

**Output:**

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*End\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*