Java is platform independent.

**JDK Java Development kit**

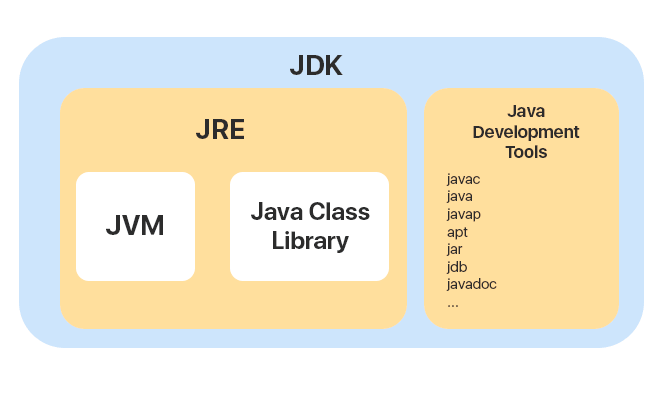
It is a software development environment which is used to develop Java applications and [applets](https://www.javatpoint.com/java-applet). It physically exists. It contains JRE + development tools.

**JRE**

It is used to provide the runtime environment. It is the implementation of JVM.

**JVM**

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.



## 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 applications are Media player, antivirus, etc.

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.

4) Mobile Application

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

**FirstJavaProgram.java,**

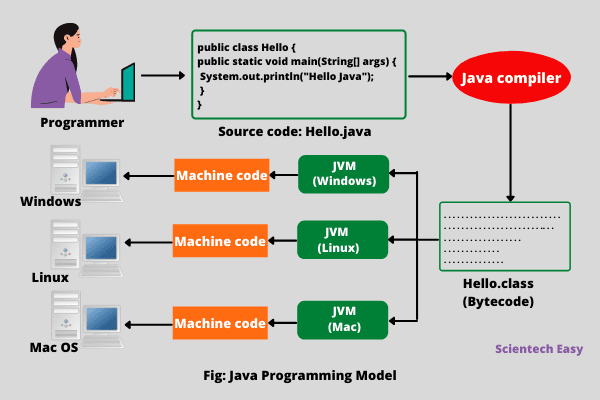
public class FirstJavaProgram {

public static void main(String[] args){

System.out.println("This is my first program in java");

}//End of main

}//End of FirstJavaProgram Class



Compile:

**Set Path in Windows:**  
Open command prompt (cmd), go to the place where you have installed java on your system

javac FirstJavaProgram.java

## 

## This compiled byte code (.class file) is platform-independent code that can be run on any different computer machine on which JVM interpreter is installed. In simple words, write once, compile and run anywhere (WOCRA).

## 

## Difference between Bytecode vs Machine code

The main difference between the byte code and machine code is that byte code can be run only on JVM whereas machine code is a set of instructions in machine language that can be directly run by the CPU.

**Object Class Method**

**Object**

**An object is an instance of a class.** A class is a template or blueprint from which objects are created.

**Example: Dog , cat .. cow … objects ,,,,, --- ANIMAL (class)**

**Dog**

**State : color, height**

**Behavior: bow bow**

* **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.

**Syntax:**

**classname objectname = new classname();**

Syntax for object creation : Animal cat = new Animal();

* className object = new className();
* // for Bicycle class
* Bicycle mybike= new Bicycle();
* Bicycle touringBicycle = new Bicycle();

**Class**

A class is a group of objects which have common properties.

It is a template or blueprint from which **objects are created.**

Syntax: class Hello

class ClassName {

// fields

int a;

int b;

int c;

// methods

returnType methodName() {

// method body a= b\*c;

}

}

fields (variables) and methods represent the **state** and **behavior** of the object respectively.

**Method**

A **method** is a block of code which only runs when it is called.

methodname { method body - business logic }

The syntax to declare a method is:

returnType methodName() {

// method body a= b\*c;

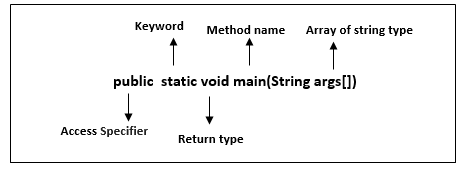
}

}

Java main() method

The main() is the starting point for JVM to start execution of a Java program. Without the main() method, JVM will not execute the program.

The syntax of the main() method is:

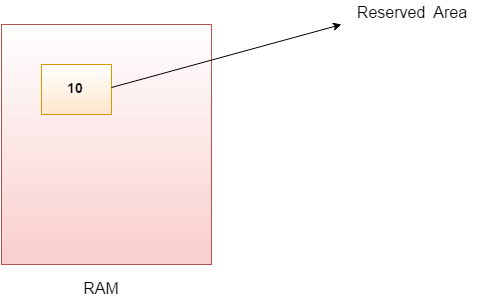


**Variables in Java**

It is a container which holds the value.

**Variable** is name of *reserved area allocated in memory*

*var accoutno = 111111;*



Example:

int i; created a variable

int i =10; assigning a variable

‘i’ ---- is VARIABLE name

‘10’ ---- is value which associated with ‘i’ variable

‘int’- is a data type that represents that this variable can hold integer values.

 ….Variable names are case sensitive in Java.

int cAmera;

int camera;

## Variables naming convention in java

1. Variables naming cannot contain white spaces. java lowercase
2. It can begin with special characters such as $ and \_.
3. As per the java coding standards the variable name should begin with a lower case letter.
4. Variable names are case sensitive in Java.

**Data Types in Java**

**Data type** defines the values that a variable can store.

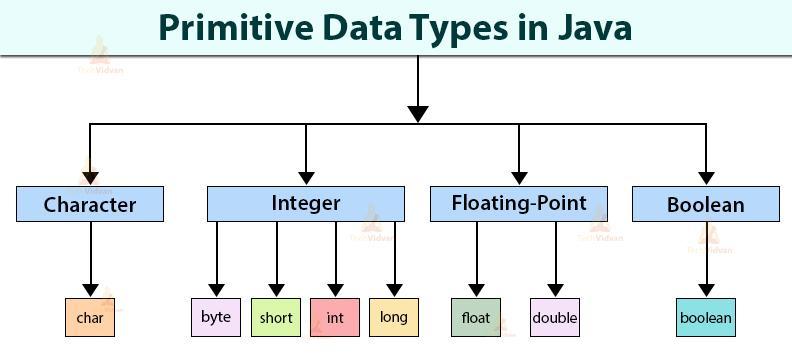
**Example** :

int data type, it can only take integer values.

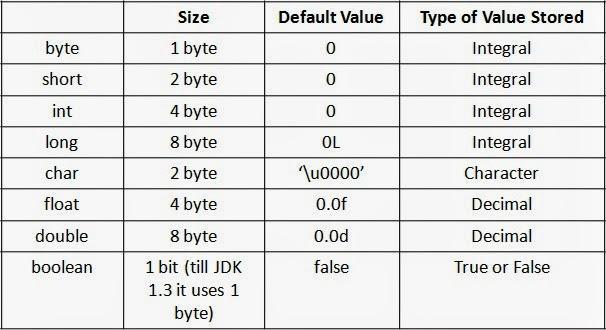
int i =11111;

**Types**

1. **Primitive data types:**
2. **Non-primitive data types:**



**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).



class JavaExample {

public static void main(String[] args) {

byte num;

num = 113;

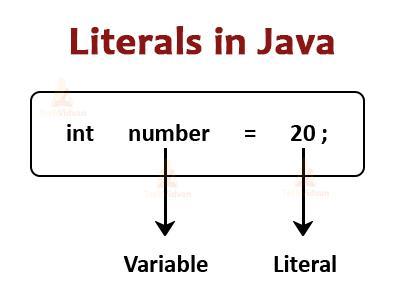
System.out.println(num);

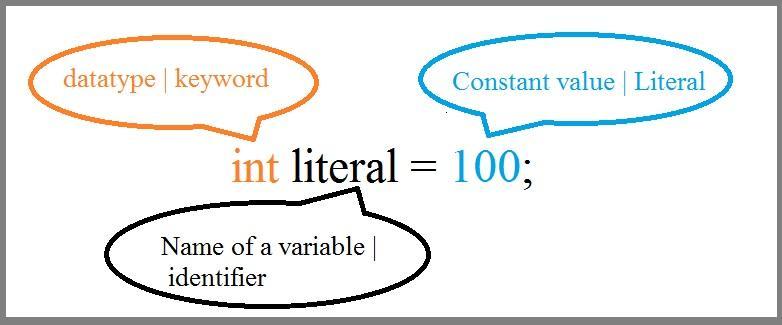
}

}

**Literals in Java**

Any value which can be assigned to the variable is called as ***literal.***

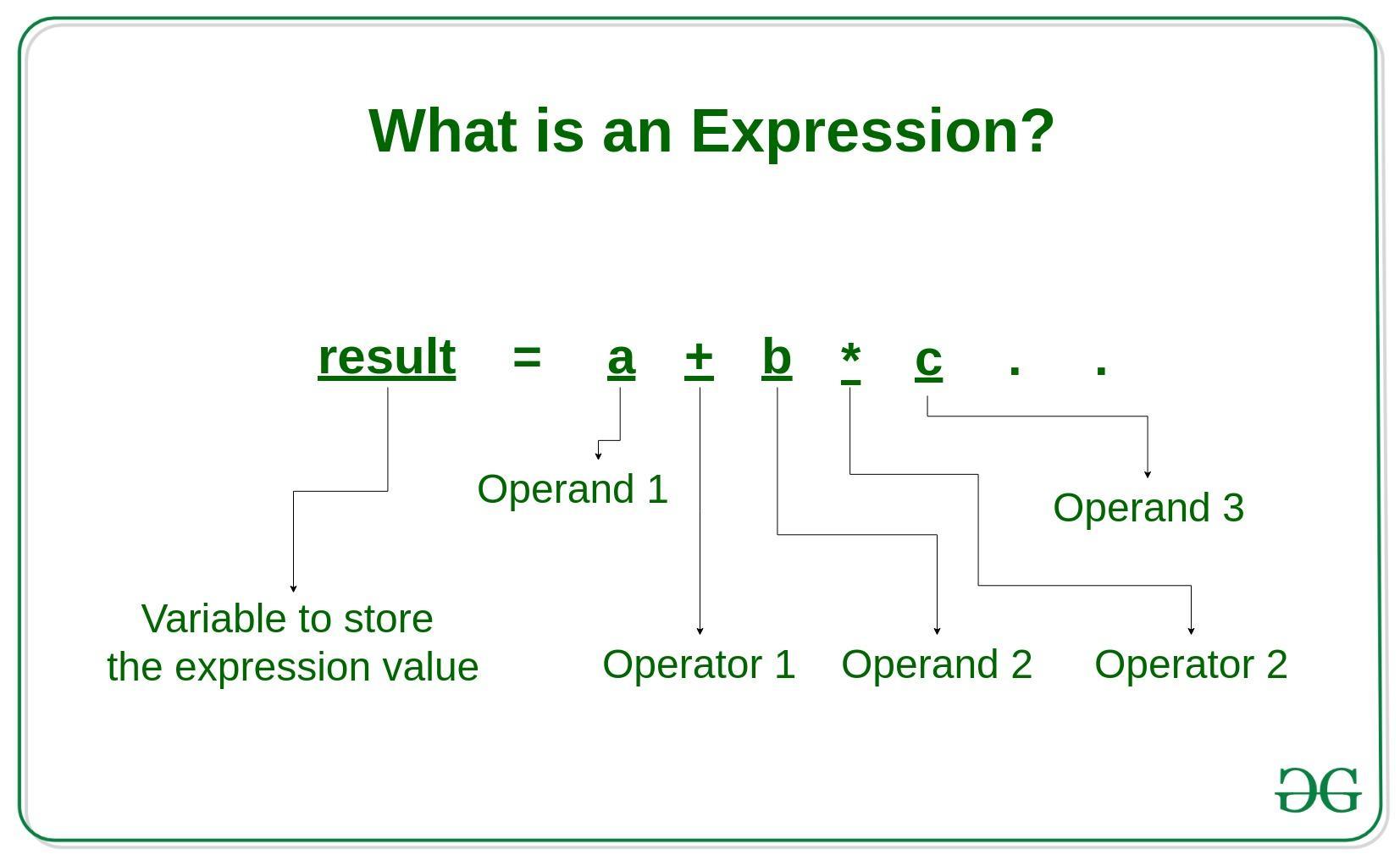




**Operators in Java**

**Operator**  is a symbol which is used to perform operations.





## 1) Basic Arithmetic Operators

Basic arithmetic operators are: +, -, \*, /,%

Example of Arithmetic Operators:

public class ArithmeticOperatorDemo {

public static void main(String args[]) {

int num1 = 100;

int num2 = 20;

System.out.println("num1 + num2: " + (num1 + num2) );

System.out.println("num1 - num2: " + (num1 - num2) );

System.out.println("num1 \* num2: " + (num1 \* num2) );

System.out.println("num1 / num2: " + (num1 / num2) );

System.out.println("num1 % num2: " + (num1 % num2) );

}

}

## 2) Assignment Operators

Assignments operators in java are: =, +=, -=, \*=, /=, %=

### Example of Assignment Operators

public class AssignmentOperatorDemo {

public static void main(String args[]) {

int num1 = 10;

int num2 = 20;

num2 = num1;

System.out.println("= Output: "+num2);

num2 += num1;

System.out.println("+= Output: "+num2);

num2 -= num1;

System.out.println("-= Output: "+num2);

num2 \*= num1;

System.out.println("\*= Output: "+num2);

num2 /= num1;

System.out.println("/= Output: "+num2);

num2 %= num1;

System.out.println("%= Output: "+num2);

}

}

## 3) Auto-increment and Auto-decrement Operators

++ and --

**num++** is equivalent to num=num+1; ++num prefix num++ postfix

**num–-** is equivalent to num=num-1;

### Example of Auto-increment and Auto-decrement Operators

public class AutoOperatorDemo {

public static void main(String args[]){

int num1=100;

int num2=200;

num1++;

num2--;

System.out.println("num1++ is: "+num1);

System.out.println("num2-- is: "+num2);

}

}

## 4) Logical Operators

Logical operators in java are: AND &&, OR || , NOT !

## 5) Comparison(Relational) operators

We have six relational operators in Java: ==, !=, >, <, >=, <=

Example operator ‘>’:

public class Main {

public static void main(String[] args) {

int x = 5;

int y = 3;

System.out.println(x > y); // returns true because 5 is greater than 3

}

}

## 6) Bitwise Operators

Control Flow Statements in Java

Java compiler executes the java code from top to bottom.

When we need to execute a set of statements based on a condition then we need to use **control flow statements.**

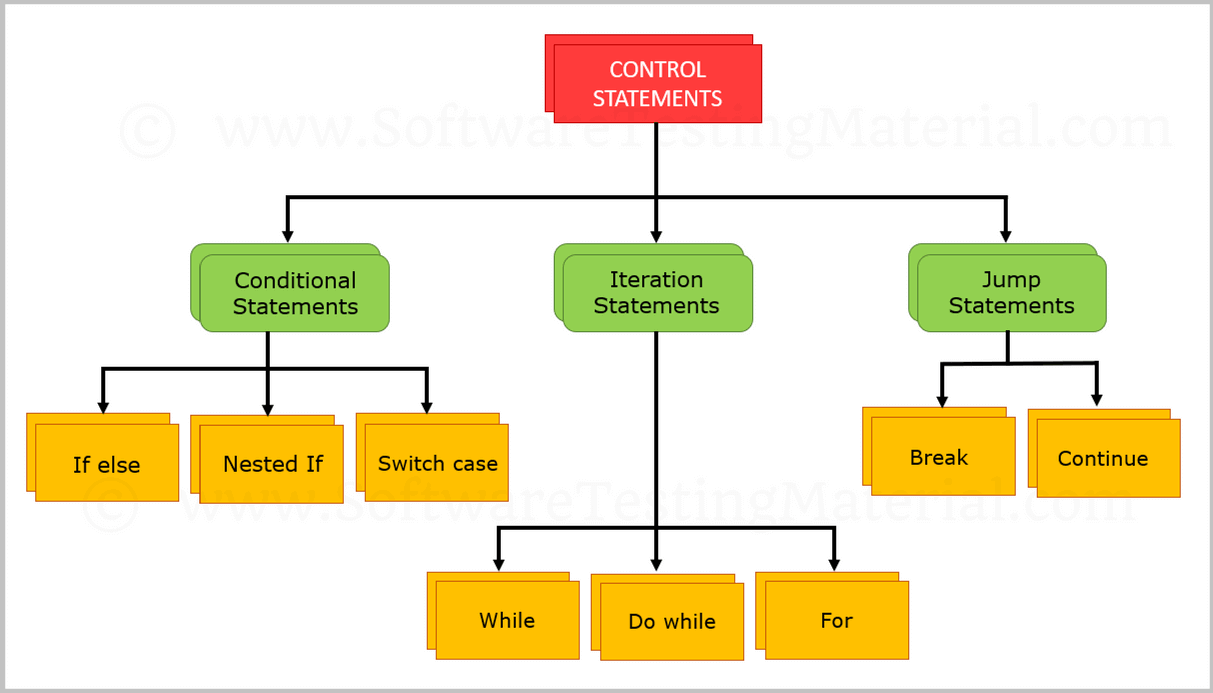
Example:

Number >0 ---------- “postitive number”

Number <0 -------- “Negative number” -- only one statement prints based on given number.

# Control Flow in Java

Java provides three types of control flow statements.



**Conditional statements**

1. if statement
2. if-else statement
3. else-if statement
4. Nested if-statement
5. Switch Statements

### Decision-Making statements:

Evaluate the Boolean expression and control the program flow depending upon the condition result.,

### **1. if statement:**

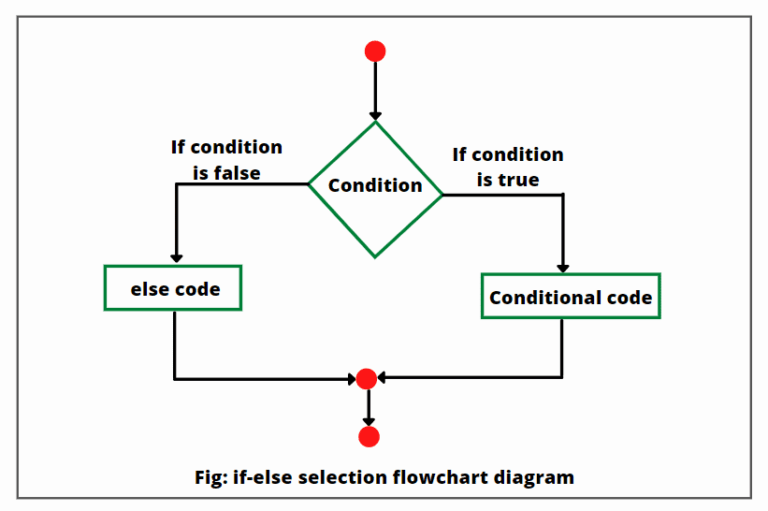
The "if" statement is used to evaluate a condition.

**Syntax:**

**if**(<condition>) {

//block of code

}



**Example:**

**public** **class** Student {

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

**int** x = 10;

**int** y = 12;

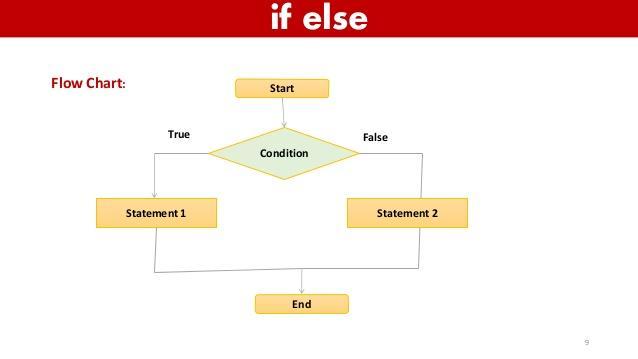
**if**(x+y > 20) {

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

}  }}

**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.



**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");

} else {

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

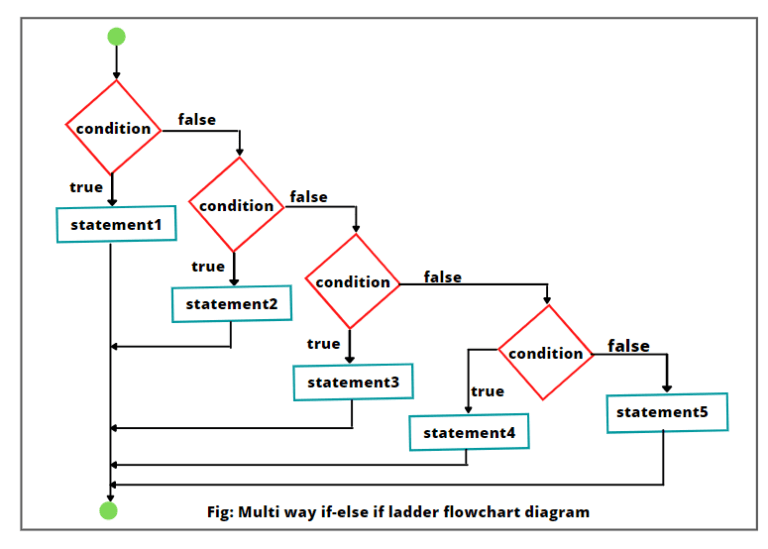
}

}

}

**3. else-if statement**

The else-if statement contains the if-statement followed by multiple else-if statements.



***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** {

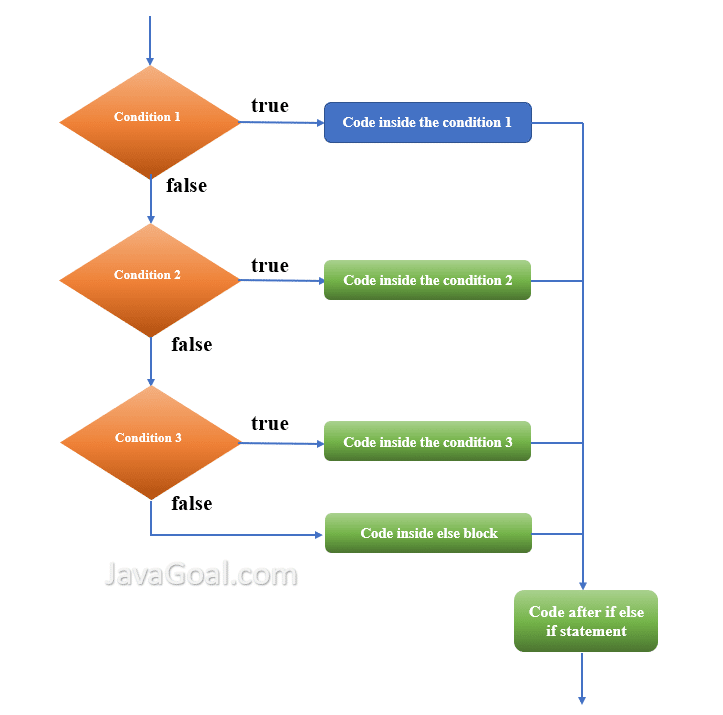
System.out.println(city);

}

}  }

**4. Nested if-statements**

In nested if-statements, the if statement contains multiple if-else statements as a separate block of code.



***Example:***

public class Test {

public static void main(String[] args)

{

int x = 20, y = 30, z = 10;

if(x == 20)

{

// First inner if statments inside outer if statement.

if(y < 50) {

System.out.println("ABC");

}

// Second inner if-else statement inside outer if statement.

if(z > 30)

System.out.println("DEF");

else

System.out.println("PQR");

}

else {

System.out.println("XYZ");

}

}

}

### Switch Statement:

The value of expression compared with value of each case.

It also enhances the readability of the program.

***Example:***

public class Main {

public static void main(String[] args) {

int day = 4;

switch (day) {

case 1:

System.out.println("Monday");

break;

case 2:

System.out.println("Tuesday");

break;

case 3:

System.out.println("Wednesday");

break;

case 4:

System.out.println("Thursday");

break;

case 5:

System.out.println("Friday");

break;

case 6:

System.out.println("Saturday");

break;

case 7:

System.out.println("Sunday");

break;

}

}

}

**Loops in Java** are the processes that execute a block of statements repeatedly until a termination condition is met.

In simple words, a loop in java is used to execute the same block of statements repeatedly. Each execution of the loop is called **iteration in java**.

## Steps to process loops in Java

There are four steps to process loops in Java that are as follows:

* Setting and initialization of a counter.
* Execution of statements in the body of loop.
* Evaluate for a specified condition for the execution of loop.
* Incrementing the counter.

## 

## 

## Types of Loop (Iteration) Statements in Java

Java language provides three types of loop statements for performing loop operations. They are:

1. while loop
2. do-while loop
3. for loop

# While Loop in Java |

# 

# The while loop in Java is the simplest of all the looping structures. It is an entry-controlled loop statement.

The general syntax for the while loop is as follows:

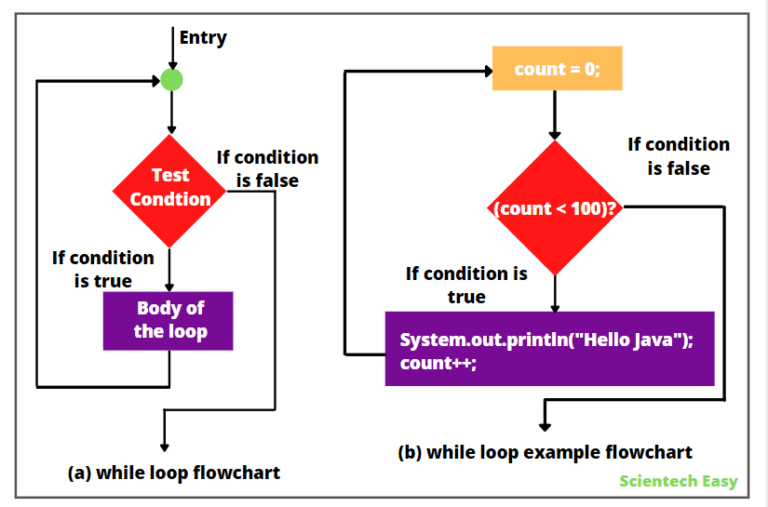
while (test condition)

{

// Loop body

Statement(s);

}



Ex:

int i =1;

while(i<100) {

System.out.println(i);

i++;

}

# 

# 

# 

# Do While Loop in Java

A **do-while loop in Java** is a variant form of [while loop](https://www.scientecheasy.com/2021/05/while-loop-in-java.html/). It is the same as a while loop except that it executes the body of the loop first and then evaluates the loop continuation condition.

The general syntax for do-while [loop in Java](https://www.scientecheasy.com/2021/04/loops-in-java.html/) is as follows:

// Initialization:

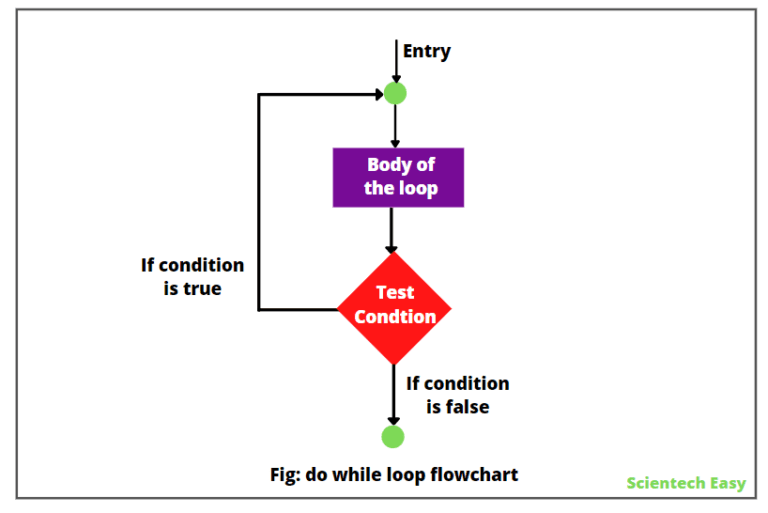
do {

// Loop body;

Statement(s);

Increment/decrement;

} while (test conditional expression);



Ex:

int x;

x = 1; // starting number is 1.

do {

System.out.println(x); // print x value.

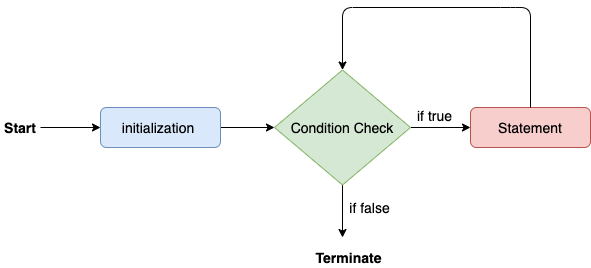
x++; // increment x value by 1.

} while(x <= 6); // This statement will be executed as long as x <= 6.

}

# For Loop in Java |

The **for loop in Java** is an entry-controlled loop structure that executes a set of statements a fixed number of times.



The general syntax of the for loop in Java is as follows:

for **(i = initialValue*;* i < endValue; i++)** {

// Loop body

Statement(s);

}

#### 1) Local Variable

A variable declared inside the body of the method is called local variable.

#### 2) Instance Variable

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

#### 3) Static variable

A variable which is declared as static is called static variable. You can create a single copy of static variable and share among all the instances of the class.

**class** A{

**static** **int** m=100;//static variable  **int** data=50;//instance variable

**void** method()

{

**int** n=90;//local variable

}

}//end of class