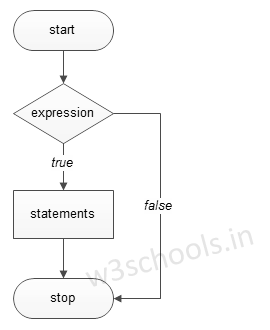
**Control Statements**

The statements that control the execution flow of the program are known as control statements.

The flowchart of Decision-making technique in Java can be expressed as:



**Decision Making Statements in Java**

1. if Statement

* if statement
* if-else statement
* else-if statement

1. switch statement

**if Statement:**

If statements in Java is used to control the program flow based on some condition, it's used to execute some statement code block if the expression evaluated to true; otherwise, it will get skipped.

**Syntax:**

if(condition){

//code to be executed

}

**Example:**

public class Sample{

public static void main(String args[]){

int a=20, b=30;

if(b>a)

{

System.out.println("b is greater");

}

}

}

**Output:**

B is greater

**If-else:**

If else statements in Java is also used to control the program flow based on some condition, only the difference is: it's used to execute some statement code block if the expression is evaluated to true, otherwise executes else statement code block.

**Syntax:**

if(condition){

//code if condition is true

}else{

//code if condition is false

}

**Example:**

public class IfElseExample {

public static void main(String[] args) {

int number=13;

if(number%2==0){

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

}else{

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

}

}

}

**Output:**

odd number

**else-if:**

**else if statements** in Java is like another if condition, it's used in the program when **if statement** having multiple decisions.

**Syntax**:

if(test\_expression)

{

//execute your code

}

else if(test\_expression n)

{

//execute your code

}

else

{

//execute your code

}

**Example:**

public class Sample {

public static void main(String args[]) {

int a = 30, b = 30;

**if (b == a) {**

**System.out.println("b is greater");**

**}**

else if(a > b){

System.out.println("a is greater");

}

else {

System.out.println("Both are equal");

}

}

}

**Output:**

Both are equal

**Java Nested if statement**

The nested if statement represents the if block within another if block. Here, the inner if block condition executes only when outer if block condition is true.

**Syntax:**

if(condition){

//code to be executed

if(condition){

//code to be executed

}

}

**Example:**

//Java Program to demonstrate the use of Nested If Statement.

public class JavaNestedIfExample {

public static void main(String[] args) {

//Creating two variables for age and weight

int age=20;

int weight=80;

//applying condition on age and weight

if(age>=18 ){

if(weight>50)

{

System.out.println("You are eligible to donate blood");

}

}

}}

**Output:**

You are eligible to donate blood

**Switch:**

Java switch statement is used when you have multiple possibilities for the if statement.

The basic format of the switch statement is:

switch(variable)

{

case 1:

//execute your code

break;

case n:

//execute your code

break;

default:

//execute your code

break;

}

**Example:**

public class Sample {

public static void main(String args[]) {

int a = 5;

switch (a) {

case 1:

System.out.println("You chose One");

break;

case 2:

System.out.println("You chose Two");

break;

case 3:

System.out.println("You chose Three");

break;

case 4:

System.out.println("You chose Four");

break;

case 5:

System.out.println("You chose Five");

break;

default:

System.out.println("Invalid Choice. Enter a no between 1 and 5");

break;

}

}

}

**Output:**

You choose five

**Java Nested Switch Statement**

We can use switch statement inside other switch statement in Java. It is known as nested switch statement.

**Example:**

//Java Program to demonstrate the use of Java Nested Switch

public class NestedSwitchExample {

public static void main(String args[])

{

//C - CSE, E - ECE, M - Mechanical

char branch = 'C';

int collegeYear = 4;

switch( collegeYear )

{

case 1:

System.out.println("English, Maths, Science");

break;

case 2:

switch( branch )

{

case 'C':

System.out.println("Operating System, Java, Data Structure");

break;

case 'E':

System.out.println("Micro processors, Logic switching theory");

break;

case 'M':

System.out.println("Drawing, Manufacturing Machines");

break;

}

break;

case 3:

switch( branch )

{

case 'C':

System.out.println("Computer Organization, MultiMedia");

break;

case 'E':

System.out.println("Fundamentals of Logic Design, Microelectronics");

break;

case 'M':

System.out.println("Internal Combustion Engines, Mechanical Vibration");

break;

}

break;

case 4:

switch( branch )

{

case 'C':

System.out.println("Data Communication and Networks, MultiMedia");

break;

case 'E':

System.out.println("Embedded System, Image Processing");

break;

case 'M':

System.out.println("Production Technology, Thermal Engineering");

break;

}

break;

}

}

}

**Output:**

Data Communication and Networks, MultiMedia

**Loops in Java**

In programming languages, loops are used to execute a set of instructions/functions repeatedly when some conditions become true. There are three types of loops in java.

* do-while loop
* for loop
* while loop

# Java 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. For loop is called entry check loop.

**Syntax:**

for ( init; condition; increment/decrement )

{

statement(s);

}

**Example**:

public class ForExample {

public static void main(String[] args) {

//Code of Java for loop

for(int i=1;i<=5;i++){

System.out.println(i);

}

}

}

**Output:**

1

2

3

4

5

**Java While Loop**

The Java while loop is used to iterate a part of the program several times. If the number of iteration is not fixed, it is recommended to use while loop. While loop is also called as entry check loop.

**Syntax:**

while(condition){

//code to be executed

}

Example:

public class WhileExample {

public static void main(String[] args) {

int i=1;

while(i<=5){

System.out.println(i);

i++;

}

}

}

**Output:**

1

2

3

4

5

**Java 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. It is called as exit check 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);

Example:

public class DoWhileExample {

public static void main(String[] args) {

int i=1;

do{

System.out.println(i);

i++;

}while(i<=5);

}

}

**Output**:

1

2

3

4

5

**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* is used to break loop or switch statement. It breaks the current flow of the program at specified condition. In case of inner loop, it breaks only inner loop.

**Syntax:**

jump-statement;

break;

**Example:**

public class BreakExample {

public static void main(String[] args) {

//using for loop

for(int i=1;i<=10;i++){

if(i==5){

//breaking the loop

break;

}

System.out.println(i);

}

}

}

**Output**:

1

2

3

4

**Java Continue Statement**

The continue statement is used in loop control structure when you need to jump to the next iteration of the loop immediately. It can be used with for loop or while loop.

The Java *continue statement* is used to continue the loop. It continues the current flow of the program and skips the remaining code at the specified condition. In case of an inner loop, it continues the inner loop only.

**Syntax:**

jump-statement;

continue;

Example:

public class ContinueExample {

public static void main(String[] args) {

//for loop

for(int i=1;i<=10;i++){

if(i==5){

//using continue statement

continue;//it will skip the rest statement

}

System.out.println(i);

}

}

}

**Output**:

1

2

3

4

6

7

8

9

10