**Control statement**

These are the statement which helps in controlling the flow of our code.

**Control statement**

**Decision or Branch statement**

**Loop statement**

**Decision or Branch statement**

* These are statements allows us to create programs where decisions are taken based on expression specified in statement.

**Classified into three types-**

1. if statement
2. if ….. else statement
3. switch statement

**If is a statement block Which contains set of code Which is dependent on expression**

1. **if statement**

If

<expression>

True

False

Execute if block

Continues

**Syntax:**

if(condition){

}

Examples:

1. if(condition);

statement1; //independent

1. if(condition)

statement1; //dependent on condition

statement2; //independent

if block is executed only when the if expression is true. Otherwise the if block is skipped. We must use boolean expression in the if statement make decision.

Assignment- WAP to verify that the person age is eligible for DL.

**Note:** If there is no braces then after if(condition) block only first executable statement is a part of if block after that statement all are independent which are not part of if block.

1. **IF …… else**

True

Execute else block

Execute if block

If

<expression>

False

Continues

Assignment: WAP to find largest number between two numbers.

**Multiple if …. else statement (Nested if)**

Assignment: WAP to find largest, mean and smallest number.

|  |  |
| --- | --- |
| class Demo{  psvm(String args[]){  int a=10, b=20, c=30;  if(a>b && a>c){  sop(“a is greater”);  }  if(b>a && b>c){  sop(“b is greater”);  }  if(c>a && c>b){  sop(“a is greater”);  }  }  } | class Demo{  psvm(String args[]){  int a=10, b=20, c=30;  if(a>b && a>c){  sop(“a is greater”);  }  else if(b>a && b>c){  sop(“b is greater”);  }  else(c>a && c>b){  sop(“a is greater”);  }  }  } |

**Note:**

1. An if statement can be used without an else statement.
2. Multiple if …. else statement can be used in a program.
3. Once an if …. else statement causes an action in a program, then the remaining if …. else statements will be ignored.

Assignment: WAP to develop movie tickets calculation.

1. **Switch case statement**

The switch case statement is used to select an action from a given set of actions, based on a specified expression.

**Syntax:**

switch(expression/variable)

{

case value1: statement1;

break;

case value1: statement1;

break;

case value1: statement1;

break;

[default: default\_statement;]

}

The expression/variable in the preceding code. Snippet can be any expression depicting a char, byte, short, int or enum variable. The switch case also supports some wrapper classes like Integer, Byte, and Short etc.

(In JDK 1.7 and 1.8, we can also use string in values)

Assignment: Red(stop), Green(go), Amber(caution)

**Loop statement**

There may be a situation when you need to execute a block of code several number of times. In general, statements are executed sequentially. The first statement in a function is executed first, followed by the second, and so on.

A **loop** statement allows us to execute a statement or group of statements multiple times.

**Classified into three types-**

1. for loop
2. while loop
3. do…. while loop
4. **for loop**

Syntax:

**for(Initialization; expression/Condition; Updating){**

**Body**

**}**

It is a loop which executes a sequence of statements multiple times.

for…. Loop is initialized 1st and then the boolean expression is checked. If the expression is evaluates to true, then the for block is executed, otherwise the loop terminates. If the for block executed, then the increment or decrement expression is updated to continue the loop.

Initialize the loop

Execute Increment Decrement expression

Execute for loop

Loop terminate

**Check (expression) condition**

**False**

**True**

Assignment: WAP to print 10 numbers.

**Enhanced for loop (For each loop):**

The for …. Each loop is used to iterate over arrays and collection. The for…. each loop has only two parts, unlike the traditional which has three parts.

The two parts of the for…. each loop are variable and array/collection.

**Syntax:**

**for(Declaration of variable : Collection/Array){**

**Body**

**}**

Example

Class ForEachDemo{

public static void main(String args[]){

int marks[] = {70,90,60,30,75};

for(int i:marks){

System.out.println(i);

}

}

}

Class Orange{

public static void main(String args[]){

Orange o1 = new Orange();

Orange orange[]={new Orange(),new Orange(),o1};

for(Orange oran:orange){

System.out.println(oran);

}

}

}

**While**

Loop terminates

**while(expression)**

Execute while block

**True**

**False**

Assignment: WAP to print all even numbers between 1-20.

class Demo{

public static void main(String args[]){

int i=1;

while(i<=20){

System.out.println(i);

i+=2;

}

}

}

**Note:** while loop, can be used with non-numeric conditions also like checking a character in a variable or checking for a strong value in a variable.

Example: char ch;

While(ch!=’y’){

}

**do while loop**

do…. while loop works similar to while loop, but the difference is do…. while loop block executes at least once irrespectively whether the expression evaluates to true or false.

Loop terminates

**while(expression)**

Execute while block

**True**

**False**

**Note:** while loop, is an entry check loop and do while, is an exit check loop.

**What is the difference between for loop and while/do while loop?**

for loop checks for numeric conditions and have definite iterations.

while/do while loop can have numeric and non-numeric condition.

It has definite iterations with numeric condition and not finite iteration with non-numeric condition.

Example:

class Demo{

public static void main(String args[]){

int i=1;

do{

System.out.println(i);

i++;

}while(i<=20);

}

}

Example: for while loop non-numeric condition;

import java.util.Scanner;

class WhileDemo{

public static void main(String args[]){

Scanner sc = new Scanner(System.in);

char ch = ‘n’;

String ans = “ “;

while(ch!=’y’){

System.out.println(“Will you listen to me?”);

ans = sc.nextLine();

ch = ans.charAt(0);

}

}

**Jump Statement**

Implementing jump statement

1. Break Statement
2. Continue
3. Return
4. **Break Statement**

Helps to terminate the loop and transfer the control in decisional and iteration statement.

Break applied only on a condition. Allows control to come out of the loop.

Example

class Demo{

public static void main(String…args){

int i=1;

while(true){

System.out.println(i++);

if(i==100)

break;

}

}

}

WAP to print ‘Z’ by using nested for loop.

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

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

if(i==1 || i==5)

System.out.print(“-“);

else{

if((i+j)==6)

System.out.print(“-“);

else

System.out.print(“ “);

}

}System.out.println();}

**Labelled Break Statement**

It is used only in nested, decisional and iteration statement. A label in a labelled break statement is similar to an identifier, which is used before the loop. The identifier is always followed by a colon (:)

Example:

class Demo{

public static void main(String args[]){

demo:

for(int i=0;i<=3;i++){

for(int j=2;j<=4;j++){

if(i==j)

break demo;//if we just use break then inner loop will get terminate //if we use break with label then it will terminate the labelled loop.

}

}

1. **Continue Statement**

In unlabelled statement, the continue statement causes the current iteration of a loop to stop and continue with the next iteration. However, in labelled statement, the continue statement stops the current iteration of the outer loop and continue with the next iteration.

Example:

class ContinueDemo{

public static void main(String args[]){

for(int i=0;i<=10;i++){

if(i==5){

continue;

}

System.out.println(i);

}

}

}

class ContinueDemo{

public static void main(String args[]){

Outer: for(int j=0;j<10;j++){

Inner : for(int i=0;i<10;i+=2){

if(j==5){

continue Outer;

}

System.out.println(i+” “+j);

}

}

}

}

1. **Return Statement**

It causes the control of execution to return the caller of a method. While using the return statement, you must be careful about the return type of the method. If the method is of the void type then you should not use return or use just return.

Return statement used in method only and it should be the last statement.

Example:

class ReturnDemo{

public static void main(String args[]){

int additionResult = add();

System.out.println(additionResult);

Or

System.out.println(add());

}

static int add(){

int a=10;

int b=20;

int result=a+b;

return result;

}

}

class Apple{

public static void main(String args[]){

Apple a2=apple();

}

static Apple apple(){

Apple a1 = new Apple();

return a1 OR new Apple();

}

}