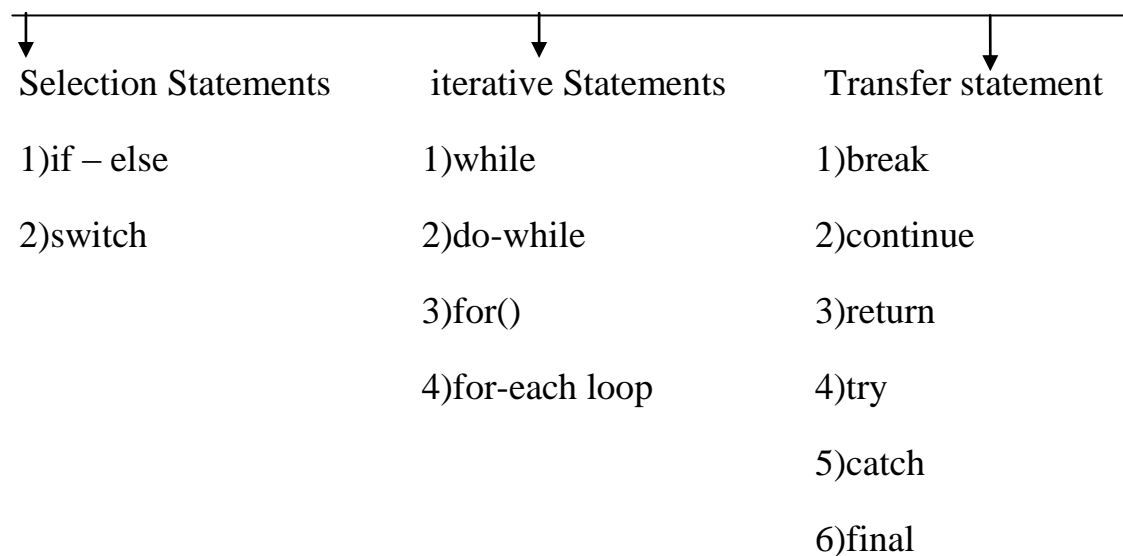


Flow Control

➤ **Flow Control:**

Flow Control describes the order in which the statements will be executed at runtime.

Flow - Control



(a) Selection Statement :-

(1) If – else :-

Syntax :-

```
If(condition)
{
    Action if b is true
}
else
{
    Action if b is false
}
```

1) The argument to the if Statement should be of Boolean type.
If we are providing any other type we will get compile time error.

Ex:-

1) `int x=0`

`if(x)` \longleftrightarrow C.E:-incompatable type found:int
required:boolean

```
{  
System.out.println("Hello");  
}  
else  
{  
    System.out.println("Hi");  
}
```

2)

`int x=10`

`if(x=20)` \longrightarrow C.E:-incompatable type found:int
required:boolean

```
{  
System.out.println("Hello");  
}  
else  
{  
    System.out.println("Hi");  
}
```

```
3)
int x=10
if(x==20)
{
System.out.println("Hello");
}
else
{
System.out.println("Hi");
}
```

O/P :- Hi

```
4)
boolean b=false;
if(b==true)
{
System.out.println("Hello");
}
else
{
System.out.println("Hi");
}
```

O/p :- Hello

5)

```
boolean b=false;
if(b=true)
{
System.out.println("Hello");
}
else
{
System.out.println("Hi");
}
```

O/p :-Hi

(2)Curly braces({,}) are optional& without curly braces

We can take only one statement which should not be declarative statement.

Ex :-

```
if(true)
System.out.println("Hello");
//valid
if(true)
int x=10;
//Not valid
```

```
if(true)
{
int x=10;
}
//valid
```

```
if(true);
//valid
```

Switch Statement:-

If several options are possible then it is never recommended to use if – else, we should go for Switch Statement.

Syntax:-

```
Switch(x)
{
    Case1:
        Action 1;
    Case2:
        Action 2;
    .
    .
    default:
        default Action ;
}
```

Cruelly braces are mandatory.

Both case & default are optional inside a switch

Ex:-

```
int x=10;

switch(x)
```

Within the switch, every statement should be under some case by default
Independent statement are not allowed.

Ex:-

```
int x=10;

switch(x)
{
    System.out.println("Hello");
```

}

C.E:-case ,default or ‘}’ expected

Until 1.4v The allowed data type for switch argument are

byte

short

int

char

But from 1.5v onwards in addition to these the corresponding wrapper classes(Byte,Short,Character,Integer) types are allowed.

<u>1.4v</u>	<u>1.5v</u>	<u>1.7v</u>
byte	Byte	String
short	Short	
int	Character	
char	Integer + enum	

if we are passing any other type we will get Compile time Error.

Ex :-

byte b=10;	char ch='a';	long l=10 l;	boolean b=true;
switch(b)	switch(ch)	switch(l)	switch(b)
{	{	{	{
}	}	}	}
Valid	Valid	Not Valid	Not Valid

Every case labels should be within the range of switch argument type otherwise we will get compile time error.

Ex:-

```
byte b=10;
```

```
Switch(b)
```

```
{
```

```
Case 10:
```

```
System.out.println("10");
```

```
Case 100:
```

```
System.out.println("100");
```

```
Case 1000:
```

```
System.out.println("1000");
```

```
}
```

```
byte b=10;
```

```
Switch(b+1)
```

```
{
```

```
Case 10:
```

```
System.out.println("10");
```

```
Case 100:
```

```
System.out.println("100");
```

```
Case 1000:
```

```
System.out.println("1000");
```

```
}
```

C.E:-Possible loss of precision

found: byte int

Every case label should be a valid Compile time Constant .if we are taking a variable as case label we will get Compile time Error.

Ex :-

```
int x=10;
```

```
int y=20;
```

```
Switch(x)
```

```
{
```

```
Case 10:
```

```
System.out.println("10");
```

```
Case y:
```

```
System.out.println("20");
```

```
}
```

C.E:- Constant expression required.

If we declare y as final then we won't to get any Compile time Error.

Expressions are allowed for both Switch argument &case label but case label should be constant expression.

EX :-

```
int x=10;

Switch(x+1)

{

Case 10:

System.out.println("10");

Case 10+20:

System.out.println("10+20");

}
```

Duplicate case label are not allowed.

EX:-

```
int x=10;

Switch(x)

{

Case 91:

System.out.println("97");

Case 98:

System.out.println("98");

Case 99:

System.out.println("99");

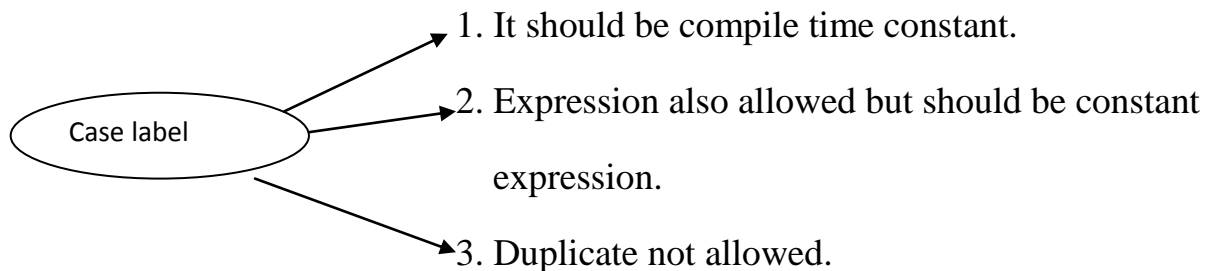
Case 'a':

System.out.println("a");

}
```

C.E:-Duplicate case label.

Summary:-



Fall-through inside Switch:-

Within the switch statement if any case is matched from that case onwards all statements will be executed until break statement or end of the switch .This is called fall through inside switch.

EX:-

Switch(x)

{ Case 0:

System.out.println("0");

Case 1:

System.out.println("1");

break;

Case 9:

System.out.println("9");

default:

System.out.println("default");

}

O/p:-

if x=0 if x=1 if x=2 if x=3

0 1 2 def

It is useful to define some common action for several cases.

EX:-

Switch(x)

{

Case 3:

Case 4:

Case 5:

System.out.println("Summer");

break;

Case 6:

Case 7:

Case 8:

Case 9:

System.out.println("Rainny");

break;

Case 10:

Case 1:

Case 2:

System.out.println("Winter");

break;

```
}
```

Default Case:-

We can use default case to define default action.

This case will be executed iff no other case is matched.

EX:-

```
Switch(x)
```

```
{
```

```
default:
```

```
System.out.println("def"); x=0,x=1
```

```
Case 0:
```

```
System.out.println("0"); x=2 , x=3
```

```
break;
```

```
}
```

(b) Iterative Statement:-

1) While:-

If we don't know the no. of iteration in advance then the best suitable loop is while loop.

EX:-

```
1)while(rs.next())
```

```
{
```

```
..
```

```
}  
2) while(ifrs.hasNext()  
{  
    ↓  
    ..    Iterator  
}
```

```
3) while(e.hasMoreElements()  
{  
    ↓  
    ..    enumeration  
}
```

Syntax:-

```
while(b)————→ boolean type  
{  
    Action  
}
```

The argument to the while loop should be a boolean type. if we are using any other type we will get compile time error.

EX:-

```
while(1)  
{  
    ↓  
    C.E:-found:int , required:boolean.  
    System.out.println("Hello");  
}
```

Curly braces({,}) are optional & without curly braces

We can take only one statement which should not be declarative statement.

Ex :-

```
while(true)
System.out.println("Hello");
//valid
```

```
while(true)
int x=10;
//Not valid
```

```
while(true)
{
int x=10;
}
//valid
```

```
while(true);
```

```
//valid
```

Ex :-

```
1) while(true)
{
    System.out.println("Hello");
}

    System.out.println("Hi");
```

C.E:- unreachable statements

2) while(false)

```
{
    System.out.println("Hello");
}

    System.out.println("Hi");
```

C.E:- unreachable statements

```
3) int a=10,b=20;

    while(a<b)

    {
        System.out.println("Hello");
    }

    System.out.println("Hi");
```

C.E:- Hello

Hello

Hello

4) final int a=10,b=20

```
    while(a<b)

    {
        System.out.println("Hello");
    }

    System.out.println("Hi");
```

C.E:- unreachable statements

3) do-while:-

If, we want to execute loop body at least once then we should go for do-while loop. Syntax:-

```
do

{ Action }
```

While(b);



Should be boolean.

Curly braces({,}) are options & without curly braces

We can take only one statement which should not be declarative statement.

Ex :-

1)do

System.out.println("Hello");

While(true);

//valid

2)do

while(true);

// valid

3)do

int x=10;

while(true);

//Not valid

4)do

{

int x=10;

}

while(true);

//valid

5)do

While(true);

//Not valid

6)do while(true)

```
{  
    System.out.println("Hello");  
}
```

While(false);

O/P:- Hello

Hello

7) do

while(true)

System.out.println("Hello");

while(false);

Note:-“;” is a valid java statement.

8) do

```
{  
    System.out.println("Hello");  
}
```

while(true);

System.out.println("Hi");

C.E:-unreachable Statement

9) do

```
{  
    System.out.println("Hello");
```

```
}  
while(false);  
System.out.println("Hi");
```

O/p:-Hello

Hi

10)int a=10,b=20;

do

```
{  
    System.out.println("Hello");  
}
```

while(a<b);

System.out.println("Hi");

O/P:-Hello

Hello

11)int a=10,b=20;

do

```
{  
    System.out.println("Hello");  
}
```

while(a>b);

System.out.println("Hi");

O/P:-Hello

Hi

```
12)final int a=10,b=20;

do

{

    System.out.println("Hello");

}

while(a<b);

    System.out.println("Hi");
```

O/P:-unreached statement

```
10)final int a=10,b=20;

do

{

    System.out.println("Hello");

}

while(a<b);

    System.out.println("Hi");
```

O/P:-Hello

Hi

For :-

This is most commonly used loop.

Syntax:-

```
For(initialization-selection;conditional-
expression;increment/decrement)
```

```
{
```

Body

}

Cruelly braces are optional & without Cruelly braces we can take only single statement which should not be declarative statement.

a)initialization-section:-

This will be executed any once.

Usually we are declaring & performing initialization for the variable in this section.

Here we can declare multiple variables of the same type but different datatype variables cannot be declared.

EX:- 1)int i=0,j=0; //Valid

2) int i=0,byte b=0; //Not Valid

3) int i=0,int j=0; //Not Valid

In the initialization section we can take any valid java statement.

EX:- int i=0;

For(System.out.println("Hello u r Sleeping");i<3;i++)

{

System.out.println("No Boss u only Sleeping");

}

O/P:- Hello u r Sleeping

No Boss u only Sleeping

No Boss u only Sleeping

No Boss u only Sleeping

Conditional Expression:-

Here we are take only java expression .

Increment & decrement:-

We can take take java statement including

System.out.println().

EX:-

```
For(System.out.println("Hello ");i<3; System.out.println("Hi"))
{
    i++;
}
```

O/P:- Hello

All 3 parts of for loop are independent of each other.

All 3 parts of for loop are optional.

EX:- for(;); —————> Statement

Represent infinite loop &it is valid.

Note:- ; is valid java statement.

Ex:-

```
1) for (int i=0;true,i++)
{
    System.out.println("Hello ");
}
    System.out.println("Hi ");
//Not valid
```

C.E:- Unreachable

```
2) for (int i=0;false;i++)
{
    System.out.println("Hello ");
}
    System.out.println("Hi ");
//Not valid
C.E:- Unreachable
```

```
3)for (int i=0;i++)
{
    System.out.println("Hello ");
}
    System.out.println("Hi ");
//Not valid
C.E:- Unreachable
```

```
4)int a=10,b=20;
    for (int i=0;a<b;i++)
    {
        System.out.println("Hello ");
    }
        System.out.println("Hi ");
```

```
O/P:- Hello
      Hi
// valid
```

```
5)final int a=10,b=20;

for (int i=0;a<b;i++)

{
    System.out.println("Hello ");
}

    System.out.println("Hi ");
//Not valid
O/p:- Unreachable Statement.
```

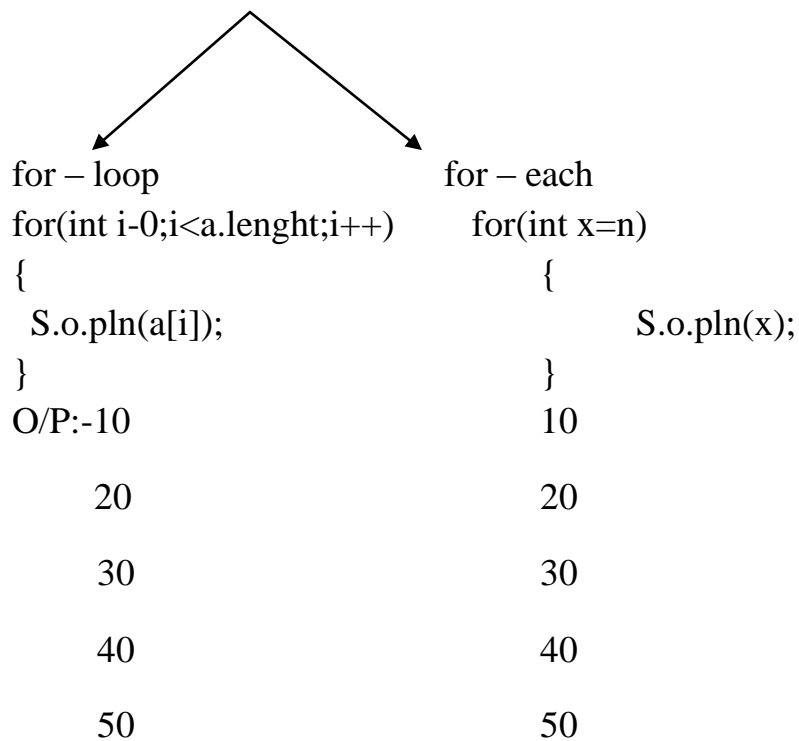
For – each() loop:-(enhanced for loop)

Introduced in 1.5 v .

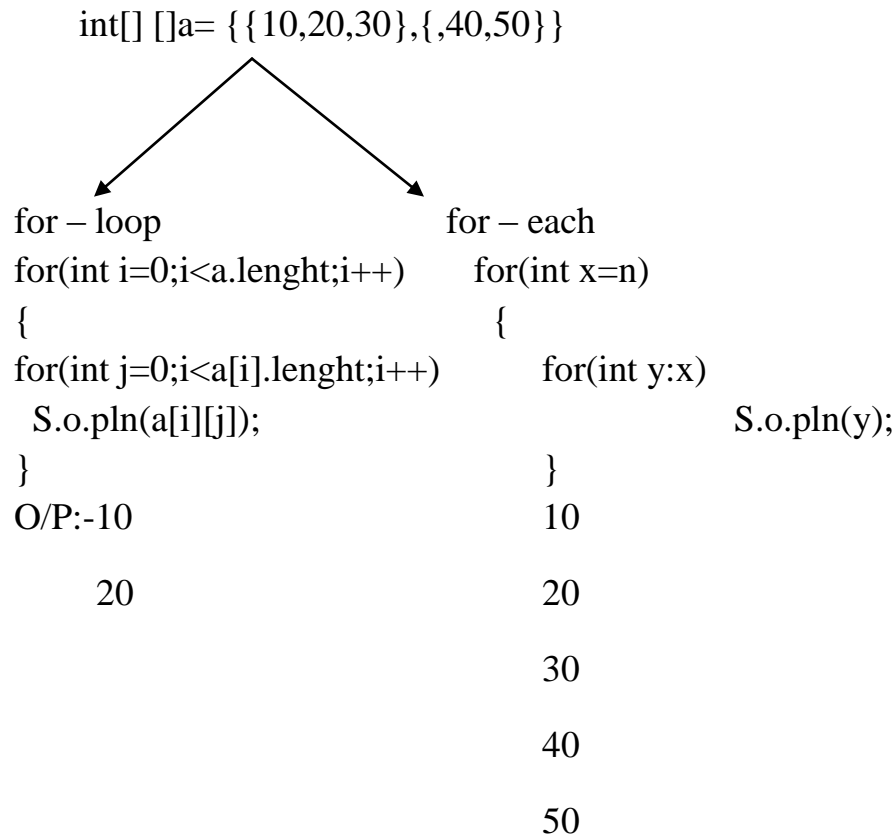
This is the most convenient loop to retrieve the elements of arrays & collection.

EX:- 1)Print element of Single dimension array by using General & enhanced for loop.

```
int[] a= { 10,20,30,4,50}
```



2) Print element of Single dimension array by using General & enhanced for loop.



Even though for- each loop is more convenient to use, but it has the following limitation.

- 1) It is not general purpose loop.
- 2) It is applicable for array & collection.
- 3) By using for-each loop we should retrieve all values of array & collection & can be used to retrived a particular set of value.

c) Transfer Statement:-

1)break:-

We can use break statement in following cases

- 1) Within the switch to stop fall through.
- 2) Inside loops to break the loop execution based on some condition.
- 3) Inside labeled blocks to break block execution on some condition.

EX:-

1) Switch(b)

```
{  
    .  
    Break;  
    .  
}
```

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

```
{  
    If(i==5)  
    }  
    Break;  
    System.out.println(i);
```

If we are using break statement anywhere else we will get compile time error.

EX:-

Class Test

```
{  
    Public static void main(String args[])  
    {
```

```
int x=10;

    if(x==0)

        break;

    System.out.println("Hello");

}

}
```

C.E:-break outside Switch or loop.

2) Continue Statement :-

We can use continue Statement to skip current iteration & continue for next iteration inside the loop.

EX:-

```
For(int i=0;i<=10;i++)

{

    If(i%2==0)

        Continue;

    System.out.println("i");

}
```

If we are using continue outside of loops we will get compiletime errors.

EX:- int x=10

```
    if(x==10)

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

C.E:-continue outside loop

3)Labeled break&Continue Statements:-

In the case of nested loops to break & continue a particular loop we should go for labeled break &continue statement.

EX:-

L1:

For(.....)

{

L2:

For(.....)

{

For(.....)

{

breakL1;

breakL2;

break;

.....

.....

.....

EX:-

L1:

For(int i=0;i<3;i++)

{

L2:

```
For(int j=0;j<3;i++)
```

```
{
```

```
    if(i==j)
```

```
        break;
```

```
        System.out.println(i+.....+j);
```

```
    }
```

```
}
```

```
break;
```

```
    1.....0
```

```
    2.....0
```

```
    2.....0
```

```
break;
```

No output

Continue:

```
    0....1
```

```
    0....2
```

```
    1....0
```

```
    1....2
```

continueL1:

```
    1....0
```

do- while Vs continue:-(very hot combination)

EX:-

```
int x=10;
```

```
do
```

```
{
```

```
    x++;
```

```
System.out.println(x);
```

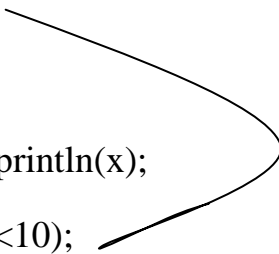
```
if(++x<5)
```

```
Continue;
```

```
    x++;
```

```
System.out.println(x);
```

```
}while(++x<10);
```



Imp Note:-

Compiler will check for unreachable statements only in the case of loops but not in if...else

EX:-

```
1) if (true)
```

```
{
```

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

```
}
```

```
Else
```

```
{
```

```
    System.out.println("Hi");
```

```
}
```

O/P:-Hello

```
2) while(true)
{
    System.out.println("Hello");
}
System.out.println("Hi");
```

O/P:-unreached statement