Flow Control statement:-

Flow control describes the order in which statement will be executed.
In java flow control statements divided into three parts:-

1) Selection Statements
2) Iteration Statements
3) Transfer Statements

Selection Statements:-Through this statement we can perform selection operation.

There are two types of section statement:-

1) if –else
2) switch

if-else :-

Syntax:-

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

Example:-In this application if condition is true then if body will be execute otherwise else body will be execute.

```
class Test
{
public static void main(String[]args)
{
int age=20;
if(age>18)
{
System.out.println("you are eligible for voting");
}
else
{
System.out.println("you are not eligible for voting");
}
}
}
```

Result:-

You are eligible for voting

switch statement:-If multiple selection are available then it not recommended to use nesting of if-else because it reduce the readability of the code overcome these problem we should go for switch statement.

Syntax:-

```
switch(x)//x is argument of switch
{
case 1:
Action-1
break;
case 2:
Action-2
break;
.
.
.
.
case n:
Action-n
break;
default:
Action-default;
}
```

Allowed argument type of switch statement are:-

1) byte, short ,char,int
2) Byte, Short, Character, Integer, enum (1.5version of java)
3) String(1.7version of java)

Fall through inside switch

Inside the switch, any case is matched on that case onward all statements will be executed until the break or end of the statement, this type of event is called fall though inside switch.

Example:-

```
class Test
{
public static void main(String[]args)
{
int x=10;
switch(x)
```

```
{
case 10:
System.out.println(10);
case 20:
System.out.println(20);
case 30:
System.out.println(30);
break;
case 40:
System.out.println(40);
default:
System.out.println("no case match");
}
}
}
```

Result:-

10
20
30

The main advantage of this approach is a common action for multiple cases.

Example:-

```
class Test
{
public static void main(String[]args)
{
String month="Dec";
switch(month)
{
case "Dec":
case "Jan":
case "Feb":
System.out.println("winter");
break;
case "Mar":
case "Apr":
case "May":
System.out.println("summer");
break;
case "Jun":
case "July":
case "Aug":
```

```
case "Sept":
System.out.println("monsoon");
break;
case "Oct":
case "Nov":
System.out.println("post monsoon");
default:
System.out.println("no month");
}
}
}
Result:-winter
```

Iteration statement:-By using iteration statements cause statement will be executed zero or more no times. The main advantage of iteration statements is to execute the task multiple times without writing a code number of times.

There are four types of iteration statements:-

1) while loop
2) do-while loop
3) for loop
4) for each loop

while loop:-

Generally, we have used a while loop if we don't know the number of iterations in advance.

Example:-To retrieve the record from the database by using a while loop because we don't know how many records are stored in the database.

Example:-

```
while(rs.next)//rs represent result set
Syntax of while loop:-
while(b)
{
Action if b is true
}
```

Note:-while loop parameter must be boolean type otherwise we will get compile time error.

do-while loop:-

- If we want to loop body will be executed at least once either condition is true or false then we should go do-while loop.
- do-while loop body will be executed first then only condition is checked.
- do-while the while must end with a semicolon otherwise we will get compile time error.

Syntax:-

```
do
{
//loop body;
}
while(condition);
```

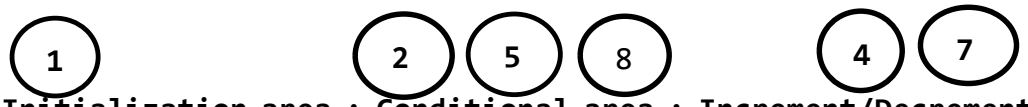
Example:-

```
class Test
{
public static void main(String[]args)
{
int x=1;
do
{
System.out.println("Hello");
x++;
}
while(x<5);
}
}
```

Result:-

```
Hello
Hello
Hello
Hello
```

for loop:-The most commonly used loop in java is for a loop. If we know the number of iterations in advance then we should go for a loop.
for loop syntax:-




```
for(Initialization area ; Conditional area ; Increment/Decrement area)
{

//loop Body

}
```

Initialization area of for loop:-Here we initialize the variable.

Conditional area:- This area is a conditional expression that must evaluate to a boolean value.

Increment/Decrement section:-In this section conditional variable will increment or decrement.

Example:-

```
class Test
{
public static void main(String[]args)
{
for(int i=0;i<5;i++)
{
System.out.println("Hello");
}
}
}
```

Result: - Five times Hello print.

for-each loop:-Enhancement for-loop, introduced in the 1.5 version of java, through this we can easy to retrieve the element from array and collection.

it is specially designed to retrieve the element from the array and collection.

Example:-Normal approach to retrieve the element from the one dimension array (without for-each loop).

```
class Test
{
public static void main(String[]args)
{
int[]x={1,2,3,4,5,6,7,8,9};
for(int i=0;i<x.length;i++)
{
System.out.print(x[i]);
}
}
}
```

Result:- 123456789

Example:-for-each loop approach to retrieve the element from the one dimension array.

```
class Test
{
public static void main(String[]args)
{
int[]x={1,2,3,4,5,6,7,8,9};
for(int x1:x)
{
System.out.print(x1);
}
}
}
```

Result:-123456789

Example:-Normal approach to retrieve the element from the two dimension array (without for-each loop).

```
class Test
{
public static void main(String[]args)
{
int[][]x={{1,2,3,4},{5,6,7,8}};
for(int i=0;i<x.length;i++)
{
for(int j=0;j<x[i].length;j++)
{
System.out.print(x[i][j]);
}
```

```
}
}
}
```

Result:-12345678

Example:- for-each loop approach to retrieve the element from the two dimension array.

```
class Test
{
public static void main(String[]args)
{
int[][]x={{1,2,3,4},{5,6,7,8}};
for(int[]x1:x)
{
for(int x2:x1)
{
System.out.print(x2);
}
}
}
}
```

Result:-1234678

- if we want to perform any other activity except retrieving then it is not recommended to use it for each-loop.

Transfer Statement:-If we want to transfer the flow of execution from one place to another place then we should go for a transfer statement.

Transfer statements are:-

1) break
2) continue
3) return
4) try

break statement:-break statement we can use to stop the execution. There are three places where we can use the break statements.

A) Inside switch
B) Inside loop
C) Inside label

break statement inside switch:-To stop fall through inside the switch.

```
class Test
{
public static void main(String[]args)
{
int x=10;
switch(x)
{
case 10:
System.out.println(10);
case 9:
System.out.println(9);
break;
default:
System.out.println("default case");
}
}
}
```

Result:-10
9

break statement inside loop:-break statement used inside the loop to break the loop of execution based on some condition.

Example:-

```
class Test
{
public static void main(String[]args)
{
for(int i=0;i<=10;i++)
{
if(i==5)
break;
System.out.print(i);
}
}
}
```

Result:-01234

break statement label block:- break statement used inside the label to break the block of execution based on some condition.

Example:-

```
class Test
{
public static void main(String[]args)
{
int x=5;
label:
{
System.out.println("label start");
if(x==5)
break label;
System.out.println("end label");
}
System.out.println("out side loop");
}
}
```

Result:-

```
label start
out side loop
```

Note:-If we want to use a break statement in any other place remaining the above then we will get compile time error.

Example:-

```
class Test
{
public static void main(String[]args)
{
int x=20;
if(x==20)
break;
System.out.println("hello");
}
}
```

Result:-

```
Error:break outside switch or loop
break;
```

continue statement:-we can use the continue statement inside the loop to skip the current iteration and it continues the rest of the iteration.

Example:-

```
class Test
{
public static void main(String[]args)
{
for(int i=5;i<10;i++)
{
if(i%2==0)
continue;
System.out.print(i);
}
}
}
```

Result:-5 7 9

return statement:-we can use return statement for execution jumps immediately back to the calling method.

Example:-In this program loop body statement will be only once printed because the return cause execution to leave not only from the loop but the entire method .so the iteration expression never runs in that case.

Example:-

```
class Test
{
static boolean m1()
{
for(int x=0;x<3;x++)
{
System.out.println("for loop");
return true;
}
return true;
}
public static void main(String[]args)
{
m1();
System.out.println("back to the caller method");
}
}
```

Result:-

```
for loop
back to the caller method
```