CONDITIONAL AND LOOPS

- If-else. Used to check condition and execute according to it.

Syntax

```
if (boolean expression True or False) {
  //If true then execute the statement here
} else {
  //Else execute the stament of else block.
}
```

Eg-

```
public class Main {
    public static void main (String[] args) {
        int salary = 25400;
        if (salary > 10000)  // 25400>10000 (True)
        {
            salary = salary + 2000;     25400+2000
        } else {                        27400 (new value)
            salary = salary + 1000;
        }
        System.out.println(salary); → printing updated value
    }
}
```

Output -

27400

- Multiple If-else. To check multiple conditions

```
if (condition 1) {
  // executes this block if condition 1 is true
} else if (condition 2) {
  // executes this block if condition 2 is true
} else if (condition 3) {
  // execute this block if condition 3 is true
} else {
  // if no condition true then this block.
}
```

Eg-

```
public class MultipleIfElse{
    public static void main(String[] args){
        int salary = 25400;
        if (salary <= 10000){
            salary += 1000;   X condition not true
        }else if (salary <= 20000){
            salary += 2000;   X condition not true
        }else{
            salary += 3000;   ✓ no condition true
        }
        System.out.println(salary);
    }
}
```

- For loop

Syntax-

```
for(initialization; condition; iterement/decrement){
    // statements
}
```

Eg-

```
public class forLoop{
    public static void main(String[] args){
        for(int i=0; i<=3; i++)
        {
            System.out.println(i);
        }
    }
}
```

O/P- 0
1
2
3

① → i=0; i<=3 (true); print(i) = 0; i++
② → i=1; i<=3 (true); print(i) = 1; i++
③ → i=2; i<=3 (true); print(i) = 2; i++
④ → i=3; i<=3

Eg-

```
public class MultipleIfelse {
    public static void main (String[] args) {
        int salary = 25400;
        if (salary <= 10000) {
            salary += 1000;   X condition not true
        } else if (salary <= 20000) {
            salary += 2000;   X condition not true
        } else {
            salary += 3000;   ✓ no condition true
        }
        System.out.println (salary);
    }
}
```

- For loop

Syntax-

```
for (initialization; condition; increment/decrement)
{
    // statements
}
```

Eg-

```
public class forloop {
    public static void main (String[] args) {
        for (int i = 0; i <= 3; i++)
        {
            System.out.println(i);
        }
    }
}
```

O/P- 0
1
2
3

① → i=0 ; i<=3 (true) ; print(i) = 0 ; i++
② → i = 1 ; i<=3 (true) ; print(i) = 1 ; i++
③ → i = 2 ; i<=3 (true) ; print(i) = 2 ; i++
④ → i = 3 ; i<=3 (true) ; print(i) = 3 ; i++
⑤ → i = 4 ; i<=3 (false) Out of Loop

- While loop Used when we dont know ~~how many~~ how many time loop will iterate.

Syntax -

```
while (condition) {
    // code to be executed
    // increment / decrement
}
```

Eg-

```
public class whileloop {
    public static void main(String []args) {
        int num = 1;
        while (num <= 3)
        {
            System.out.println(num);
            num++;
        }
    }
}
```

check condition

initial value num = 1

print

num = 2

repeat until cond. false

num = num + 1

O/P- 1
2
3

- do while Loop Used when we want to execute statement atleast once

```
do {
    // code to be executed
    // update ---> increment/decrement
} while(condition);
```

- It executes statement and then checks condition if true then it executes further in loop until ~~its~~ the condition is false
- exit control loop (as it checks condition after execution of stmt)

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

O/p
1
2
3
4
5

initial value 1
↓
executes statements: print(n) &
increase value by 1
↓
check if condn true then repeats

WHILE LOOP	DO WHILE LOOP
- Used when no. of iteration not fixed	- Used when we want to execute statement atleast once
- Entry controlled loop	- Exit controlled loop
- no semicolon required at end of while condition	- semicolon required at end of while(condition)

- Switch Statements
  - In switch statements, you can jump to various cases based on your expression.
  - duplicate cases not allowed.

- Syntax-

```
switch (expression){
    case one:
    //code block
    break;                    ← terminate the sequence.
    case two:                 (If break not used it will execute
    //code block               below cases code blocks too)
    break;
    default:                  ← default code block will be
    //code block               executed if no cases match
}                             the expression value.
```

- New Syntax: (enhanced switch statement)

```
switch (expression){
    case one -> //do this;
    case two -> //do this;
    default -> //do this;
}
```

* == operator compares reference or memory location of object in heap (address comparison)

* .equals() content comparison.

```
public class Test{
    public static void main(String[] args)
    { String s1="HELLO"; String s2="HELLO";
      String s3= new String("HELLO");
      System.out.println(s1==s2); // true
      System.out.println(s1==s3); // false
      System.out.println(s1.equals(s2)); // true
      System.out.println(s1.equals(s3)); // true
    }
}
```