# Unit VII : Conditional Statement in Java

## if statement

This statement checks the condition first. If the condition is true then the next statement is executed otherwise ignored.

Eg:- 
```
if (a>10)
    c = a*a;
System.out.println(c)
```

Explaination

If (a>10), then only the value of c will be (a*a) else not

## if-else statement

If we want want to execute either of the two statements depending upon a given condition then 'if-else' statement is used.

Eg:- 
```
if (a>b)
    max = a;
else
    max = b;
```

Explaination

if a is greater than b. then the value of max is a or else b.

## if-else-if statement

Sometimes, it may happen that the given condition is false and the user wants to check another condition for taking necessary action, under the situation if-else-if statement is used.

Eg:- 
```
if (marks >= 75)
System.out.println ("Distinction");
else if (marks >= 60)
System.out.println ("First Division");
elseif (marks >= 50)
System.out.println ("Second Division");
else
System.out.println ("Third Division");
```

Explaination

if marks is greater than or equal to 75 it will print "Distinction" otherwise it will check whether marks is greater than or equal to 60. If the condition satisfies it will print "First Division" or else it will again check whether marks is greater than or equal to 50, if condition satisfies it will print "Second Division" otherwise it will print "Third Division".

## Nested if statement

When if statement is used within another if statement, the construct is said to be a 'Nested if'

Example

```
if (a) =10)
   { if (a<100)
         c=a*a;
   }
else
   c = a *a * a;
```

Explaination

if a is greater than or equal to 10, it will check whether a is less than 100 or not, if a is less than 100 then c = a*a. If at first a is not greater than or equal to 10, then c=a*a*a.

## Unusual Termination of a program (System.exit(0))

Sometimes we may need to terminate a program even if all its statements have not been executed This function System.exit(0) is used when we want to terminate the execution at any instance of the program.

Syntax : System.exit(0);

## Switch Statement

A switch statement in Java is a control flow statement that allows a program to execute different code blocks based on the value of an expression.

Example

```
switch(num) {
    case 1:
        System.out.println("You entered 1");
        break;
    case 2:
        System.out.println("You entered 2");
        break;
    case 3:
        System.out.println("You entered 3");
        break;
    default:
        System.out.println("Invalid number");
        break;
}
```

Explaination

We use a switch statement to check the value of 'num'. If the num is equal to 1, we print out "You entered 1". If 'num' is equal to 2, we print out "You entered 2". If 'num' is equal to 3, we print out "You entered 3". If 'num' is not equal to 1, 2 or 3, we print out "Invalid Number".

The break statements are used to exit the switch statements after each case is executed. If we didn't include the 'break' statement, the program would continue executing the code in the subsequent classes.

# Terms related to Switch Statement

## (i) Control Variable

The variable passed as an argument to the switch statement the decides which case is to be executed is known as control variable.

## (ii) Break Statement

This statement is used at the end of each case which acts as the case terminator. As soon as break is encountered, the control is forced to move out of the switch block.

## (iii) Default Case

If no case is matched in the switch block for a given value of control variable, default case is executed implicitly

## (iv) Fall Through

In case a break statement is not used at the end of a case, the control enters into the next case of execution. This condition is said to be fall through.