Conditional Statements

* In java, conditional statement is a type of control structure that allows the program to make decisions based on certain conditions.

* Java supports the following conditional statements-

i) if statement -

The if statement is used to execute a block of code if a condition is true.

syntax:

```
if (condition)
{
    //code to execute, if condⁿ is true
}
```

Eg:-

```
Public class Test
{
    Public static void main(String a[])
    {
        int x = 10;
        if (x > 5)
        {
            S.o.P("x is greater than 5");
        }
}
```

ii) if... else statement -

<u>Syntax</u>:

```
if(condition)
{
    // code to execute if condn is true
}
else
{
    // code to execute if condn is false
}
```

Eg:- WAP to check given no. is odd or even -

```
Public class Test
{
  Public static void main(String a[])
  {
    int n;
    Scanner sc = new Scanner(System.in);
    S.o.pln("Enter a number");
    n = sc.nextInt();
    if (n % 2 == 0)
    {
        S.o.Pln("Even number");
    }
    else
    {
        S.o.pln("Odd number");
    }
  }
}
```

iii) else if ladder (if ... else if ... else) -

Syntax :

```
        if (condition)
         {
              // code 1
         }
        else if (condition)
         {
              // code 2
         }
        else if (condition)
         {
              // code 3
         }
        else
         {
              // code 4
         }
```

Eg:- Program to check given no. is positive, negative or zero.

```
Public class Test
{
    Public static void main (String a[])
     {
         int n;

         Scanner sc = new Scanner (System.in);
```

```
        S.o.pln(" Enter a number");
          a= sc.nextInt();
        if (a<0)
            S.o.pln("Negative number");
        else if (a>0)
            S.o.pln(" Positive number");
        else
            S.o.pln(" Zero");
    }
}
```

iv) Nesting of if... else -

When there are another if-else statement in if-block, then it is called as nesting of if-else statement.

Syntax -

```
if (condition)
{
    if (condition)
        code 1;
    else
        code 2;
}
else
{
    if (condition)
        code 3;
    else
        code 4;
}
```

Eg:- Program to find grade of students a/c to their marks-

```
Public class Test
{ Public static void main (String[] args)
  {
      int marks , PassingMarks ;
      char grade;

      PassingMarks = 40;

      Scanner sc = new Scanner (System.in);
      S.o.pln (" Input marks scored by you ");
       marks = sc.nextInt();
       if ( marks >= PassingMarks )
        {       if ( marks > 90)
                     grade = 'A' ;
                else if ( marks >75)
                    grade = 'B' ;
                else if ( marks > 60)
                     grade = 'C' ;
                 else
                    grade = 'D' ;
            S.o.pln (" You passed the exam and your grade
                                             is " + grade);
       else
       {    grade = 'F'
            S.o.pln (" You failed and your grade is " + grade);
       }
  }   }
```

* Switch-Case Statement -

Switch statement allows us to execute one statement from many statement based on the value of an expression.

Syntax:

```
switch (expression)
{
    case value1 :
        // code to execute if expression equals value1
        break;
    case value2 :
        // code to execute if expression equals value2
        break;
    - - - - - - - -
    - - - - - - -
    - - - - - - -
    case value n:
        // code to execute if expression equals value n
        break;
    default:
        // code to execute if expression does not
                         match any of the cases.
        break;
}
```

Eg:- Program to check given alphabet is consonent or vowel-

```
Public class Test
{
 Public static void main(String a[])
 {
     char ch;
    Scanner sc = new Scanner(System.in)
     S.o.pln("Enter any alphabet");
     ch = sc.next().charAt(0);
     switch(ch)
     {
        case 'a':
           S.o.pln("Vowel");
           break;
        case 'e':
           S.o.Pln("Vowel"); break;
        case 'i':
           S.o.pln("Vowel"); break;
        Case 'o':
           S.o.Pln("Vowel"); break;
        case 'u':
           S.o.Pln("Vowel"); break;
        default:
           S.o.Pln("Consonent");
 }  }  }
```

Looping Statement

* It is a block of statement that performs set of instructions.

" The looping is a process of repeating a single statement or group of statement untill some condition for termination of loop is satisfied".

* <u>Types of loops</u> -

i) while loop -

When we want to do something a fixed no. of time but not known about the no. of iteration in a program then while loop is used.

In this loop, first condition is checked, if it is true, body of the loop is executed otherwise control will be come out of the loop.

<u>Syntax</u>:

```
while (condition)
{
    //code to be executed repeatedly

}
```

Eg:- Program to print, table of one using while-loop -

```
Public class Test
{
   Public static void main (String a[])
   {
       int i = 1;
       while ( i <= 10)
        {
           S.o.pln (i);
           i++ ;
        }
    }
}
```

ii) do- while loop -

The do-while loop is similar to the 'while' loop, but the code inside the loop will be executed at least once before the condition is checked.

Syntax:

```
      do
       {
          // code to executed repeatedly
       } while (condition);
```

Eg:- Table of one, using do-while loop -

```
Public class Test
{
   Public static void main(String a[])
    {    int i = 1;

         do
          {
              S.o.pln (i);
              i++;
          } while (i<=10);
    }
```

iii) for loop -

This loop is generally used when the no. of iteration are known in advance.

Syntax:

```
for(initialisation ; condition ; incre/decre)
{
        // code to be executed
}
```

Eg:- Table of one using for loop -

```
Public class Test
{
    Public static void main (String a[])
    {
        ~~in~~
        for (int i=1;  i<=10;  i++)
        {
            S.o.pln (i);
        }
    }
}
```

iv) for-each loop (Range based for loop) -
This loop is work with collection of elements means array or vector type container.

```
for (type var : container)
{
    // code to be executed for each element
}
```

Eg:-

```
int arr[] = ~~new~~ ~~in~~ {1, 2, 3, 4, 5};

for (int x : arr)
{
    S.o.pln (x);
}
```

```
o/p -  1
       2
       3
       4
       5
```

* Break Statement -

The 'break' statement is used to exit a loop or switch statement. When the 'break' statement is encountered, the ~~prog~~ control will immediately exit the loop or switch statement and continue executing the next line of code after the loop or switch.

Eg:-
```
Public class Test
{
  Public static void main (String a[])
  {
    for (int i=1;  i<=10;  i++)
    {
      if (i==6)
          break;
      S.o.pln(i);
    }
    S.o.pln("Loop is terminated");
  }
}
```

o/p -
```
1
2
3
4
5
Loop is terminated
```

* Continue Statement -

The 'continue' statement is used to skip the remaining code in a loop iteration and start the next iteration.

When the 'continue' statement is encountered, the program will immediatly exit the current iteration of the loop and start the next iteration.

Eg:-
```
Public class Test
{
    public static void main(String a[])
    {
        for(int i=1; i<=10; i++)
        {
            if (i % 2 == 0)
                continue;

            S.o.pln(i);
        }
    }
}
```

```
o/p -   1
        3
        5
        7
        9
```