Control Statements

In Java programs, statements are executed sequentially in the order in which they appear in the program. But sometimes we may want to use a condition for executing only a part of program. Also many situations arise where we may want to execute some statements several times. Control statements enable us to specify the order in which the various instructions in the program are to be executed. This determines the flow of control. Control statements define how the control is transferred to other parts of the program. Java language supports four types of control-statements, which are as

1. if...else
2. switch
3. loop
    - while
    - do...while
    - for
    - enhanced for each

A compound statement or a block is a group of statements enclosed within a pair of curly braces {} The statements inside the block are executed sequentially.

The general form is

```
 {
  statement l;
  Statement 2;
 }
```

For example { l=4; b=2; area=l*b; System.outprintf("%d",area) ; }

A compound statement is syntactically equivalent to a single statement and can appear anywhere in the program where a single statement is allowed.

If….else Statement

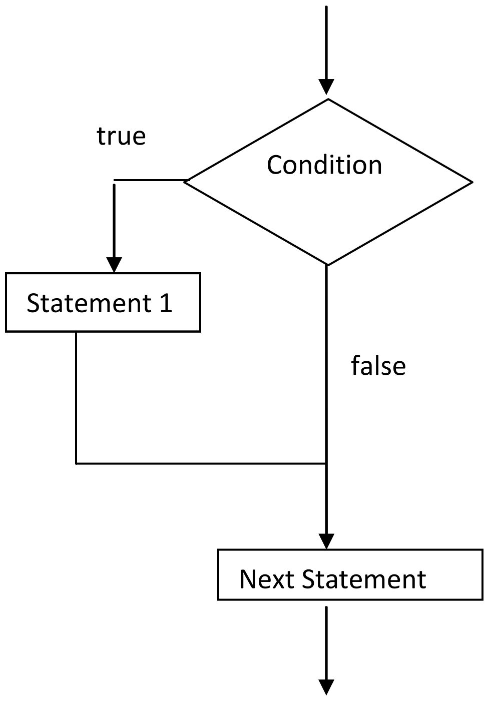
This is a bi-directional conditional control statement. This statement is used to test a condition and take one of the two possible actions. If the condition is true then a single statement or a block of statements is executed (one part of the program), otherwise another single statement or a block of statements is executed (other part of the program).

Syntax 1:

```
if(condition)
      statement 1;
```

```
if(condition)
        {
           Statement 1;
           Statement 2;
        }
```

There can be a single statement or a block of statements after the if part.

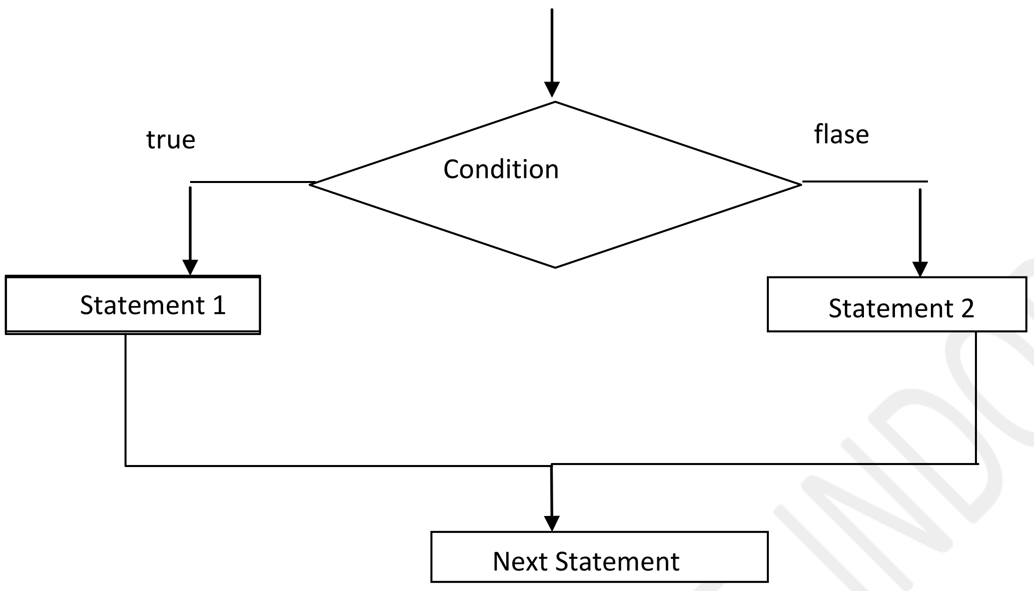


Here if the condition is true then statement1 is executed, and if it is false, then the next statement which is immediately after the if control statement is executed.

Syntax 2 :

```
if(condition)
        statement 1;
 else
       statement 2;
```

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



Flow chart of if. ..else control statement

Here if the condition is true then statement1 is executed and if it is false then statement2 is executed. After this the control transfers to the next statement which is immediately after the if. ..else control statement.

Program to print a message if negative number is entered

```
Import java.util.Scanner;
class IfDemo1
{
public static void main (String[] args )
 {
   Scanner sc = new Scanner(System.in);
    int num;
    System.out.println("Enter a number :");
   num = sc.nextInt();
   if (num<0)
     System.out.println( ("Number entered is negative");
 System.out.printf ("Value of num is : %d",num);
 }
}
```

```
1st run:-
Enter a number: -6
Number entered is negative
Value of num is -6
```

```
2nd run:-
Enter a number : 8
Value of num is 8
```

Program to print the larger and smaller of the two numbers

```
Import java.util.Scanner;
class IfDemo2
{
public static void main (String[] args )
 {
   Scanner sc = new Scanner(System.in);
    int a,b;
   System.out.println ("Enter the first number") ;
   a = sc.nextInt();
   System.out.println ("Enter the second number") ;
   b = sc.nextInt();
   if (a>b)
       System.out. printf(" first number %d is larger than second number %d ",a,b);
   else
       System.out. printf(" second number %d is larger than first number %d ",b,a);
    }
  }
```

Program to print whether the number is even or odd

```
Import java.util.Scanner;
class IfDemo3
{
public static void main (String[] args )
 {
   Scanner sc = new Scanner(System.in);
    int num;
    System.out.println("Enter a number :");
   num = sc.nextInt();
   if (num%2==0)
     System.out.println( ("Number entered is Even");
  else
     System.out.println( ("Number entered is odd"); }
}
```

Nesting of if...else

We can have another if. ..else statement in the if block or the else block. This is called nesting of if...else statements.

Here is an example of nesting where we have if...else inside both if block and else block.

```
if (condition 1)
       {
         if (condition 2)
        statementA1;
        else
        statementA2;
       }
else
      {
if(condition 3)
       statementB1;
 else
       statementB2;
      }
```

Program to find largest number from three given numbers

```
Import java.util.Scanner;
class IfDemo4
{
public static void main (String[] args )
 {
   Scanner sc = new Scanner(System.in);
    int a,b,c,larger;
   System.out.println ("Enter the first number”) ;
   a = sc.nextInt();
   System.out.println ("Enter the second number”) ;
   b = sc.nextInt();
   System.out.println ("Enter the second number”) ;
   c = sc.nextInt();
   if (a>b)
     {
      If(a>c)
          larger = a;
      else
          larger = c;
     }
   else
      {
      If(b>c)
          larger = b;
      else
          larger = c;
     }
```

```
  System.out. printf("larger number among the three is : %d",larger);
}
}
```

program finds whether a given year is leap or not
A centennial(divisible by 100) year is leap, if it is divisible by 400, and a non centennial year is leap if it is divisible by 4.

```
Import java.util.Scanner;
class IfDemo5
{
public static void main (String[] args )
 {
  Scanner sc = new Scanner(System.in);
   int year;
  System.out.println ("Enter the first number") ;
  year = sc.nextInt();
   if (year%100==0) {
              if (year%400==0)
                    System.out. println ("Leap year");
             else
                    System.out. println ("Not Leap Year");
                 }
     else
           {
           if (year%4==0)
                System.out. println ("Leap year");
          else
               System.out. println ("Not Leap year");
            }
      }
  }
```

Note that we can write this program using a single if condition and && and II operators.

```
 if (year%4==0 && year%100! =0 II year%400==0)
             System.out. println ("Leap year");
 else
       System.out. println ("Not Leap year");}
```

else if Ladder
This is a type of nesting in which there is an if. ..else statement in every else part except the last else part. This type of nesting is frequently used in programs and is also known as else if ladder.

```
If(condition1)
    StatementA;
else
    If(condition2)
          StatementB;
    else
         If(condition3)
            StatementC;
         else
             If(condition4)
                  StatementD;
```

Here each condition is checked, and when a condition is found to be true, the statements corresponding to that are executed, and the control comes out of the nested structure without checking remaining conditions. If none of the conditions is true then the last else part is executed.

Program to find out the grade of a student when the marks of 4 subjects are given. The method of assigning grade is as

per>=85	**grade=A**
per<85 and per>=70	**grade=B**
per<70 and per>=55	**grade=C**
per<55 and per>=40	**grade=D**
per<40	**grade=E**

```
Import java.util.Scanner;
class IfDemo6
{
public static void main (String[] args )
 {
   Scanner sc = new Scanner(System.in);
    float m1,m2,m3,m4,total;
   char grade;
   System.out.println ("Enter the first subject marks");
   m1 = sc.nextFloat();
   System.out.println ("Enter the second subject marks");
   m2 = sc.nextFloat();
   System.out.println ("Enter the third subject marks");
   m3 = sc.nextFloat();
   System.out.println ("Enter the forth subject marks");
   m4 = sc.nextFloat();
   total=ml+m2+m3+m4;
    per=total/4;
    if (per>=85)
          grade= 'A' ;
    else
          if(per>=70)
```

```
             grade='B';
            else
                if (per>=55)
                    grade= 'C' ;
                else
                    if (per>=40)
                        grade= 'D' ;
                    else
                         grade='E';
System.out.printf ("Percentage is %f\n Grade is %c\n", per, grade) ;
} }
```

If we don't use the else if ladder, the equivalent code for this problem would be

```
if(per>=85) grade=' A' ;
else if(per<85&&per>=70) grade='B' ;
      else if(per<70&&per>=S5) grade=' C' ;
           else if(per<55&&per>=40) grade= 'D' ;
                  else(per<40) grade=' E' ;
```

In if.. .else ladder whenever a condition is found true other conditions are not checked, while in this case all the conditions will always be checked wasting a lot of time, and moreover the conditions here are more lengthy.

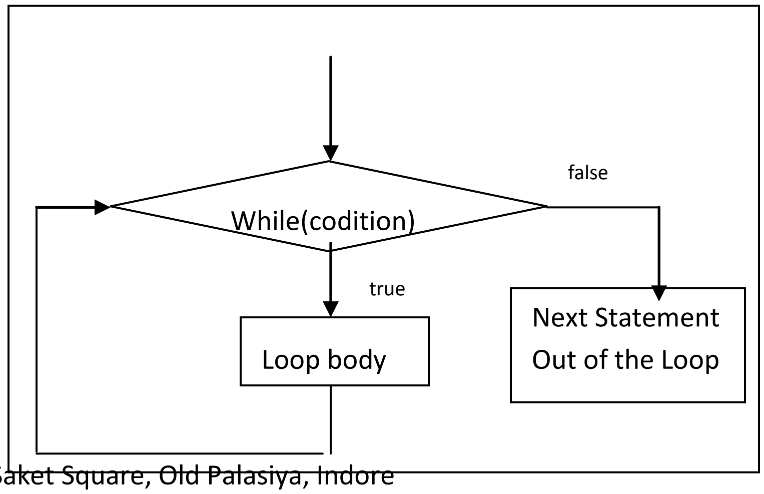
Loops

Loops are used when we want to execute a part of the program or a block of statements several times. For example, suppose we want to print "Java is the best" 10 times. One way to get the desired output is - we write 10 **System.out.println** statements, which is not preferable. Other way out is use loop. Using loop we can write one loop statement and only one **System.out.println** statement, and this approach is definitely better than first one. With the help of loop we can execute a part of the program repeatedly till some condition is true. There are three loop statements in Java.

while loop
The while statement can be written as

```
while(condition)
    statement;

while(condition)
{ statement;
statement;
}
```



Program to print the numbers from 1 to 10 using while loop

```
class WhileDemo1
{
public static void main (String[] args )
 {
  int i=l;
while(i<=10)
{
System.out.printf("%d\t",i) ;
 i=i+1; / * Statement that changes the value of condition * /
}
}
}
```

Here initially the condition (i <= 10) is true. After each iteration of the loop, value of i increases by 1 when the value of i equals 11 the condition becomes false and the loop terminates.

Output:

```
1 2 3 4 5 6 7 8 9 10
```

Note that inside the body of the loop there should be a statement that alters the value of the condition, so that the condition becomes false ultimately at some point.

Program to print numbers in reverse order with a difference of 2

```
class WhileDemo2
{
public static void main (String[] args )
 {
 int k=10;
 while(k>=2)
   {
     System.out.printf("%d\t",k);
     k=k-2;
   }
  }
}
```

Output:

```
10 8 6 4 2
```

Program to print the sum of digits of any number

```
Import java.util.Scanner;
class WhileDemo3
{
public static void main (String[] args )
 { int n,sum=0;rem=0;
   Scanner sc = new Scanner(System.in);
   System.out.println ("Enter the number") ;
   n = sc.nextInt();
   while(n != 0)
    {
     rem = n%10;
     sum=sum+rem;
     n=n/10;
    }
 System.out.println ("Sum of digits of given number is :" + sum) ;
  }
}
```

```
Output:
Enter the number: 1452
Sum of digits = 12
```

Program to find the factorial of any number

```
Import java.util.Scanner;
class WhileDemo3
{
public static void main (String[] args )
 {
    int n, num; long fact=1;
    Scanner sc = new Scanner(System.in);
    System.out.println ("Enter the number") ;
    n = sc.nextInt();
    num=n;
    if(n<0)
       System.out.println ("No factorial of negative number");
   else
         while (n>1)
            {
             fact*=n;
               n- - ;
            }
System.out.printf ("Factorial of %d is %d “,num,fact);
}
}
```

Program to convert a binary number to a decimal number

```
Import java.util.Scanner;
class WhileDemo4
{
public static void main (String[] args )
 {
   int n,nsave,rem,d,j=1,dec=0;
   Scanner sc = new Scanner(System.in);
   System.out.println ("Enter the number in binary”) ;
    n = sc.nextInt();
    nsave=n;
     while (n>0)
        {
          rem=n%10;
          d=rem*j;
          dec+=d;
          j*=2;
          n/=10;
           }
System.out. printf("Binary number = %d, Decimal number = %d\n",nsave,dec);
 }
}
```

Program to reverse a given number

```
Import java.util.Scanner;
class WhileDemo5
{
public static void main (String[] args )
 { int n,rev=0;rem=0;
  Scanner sc = new Scanner(System.in);
  System.out.println ("Enter the number”) ;
  n = sc.nextInt();
  while(n != 0)
   {
    rem = n%10;
    rev=rev*10+rem;
    n=n/10;
   }
 System.out.println ("Reverse number of given number is :" + rev) ;
  }
}
```

```
Output:
Enter the number: 1452
Sum of digits = 2541
```

Program to check palindrome number

```
Import java.util.Scanner;
class WhileDemo6
{
public static void main (String[] args )
 { int n,rev=0;rem=0,m;
   Scanner sc = new Scanner(System.in);
   System.out.println ("Enter the number") ;
   n = sc.nextInt();
  m=n;
   while(n != 0)
    {
     rem = n%10;
     rev=rev*10+rem;
     n=n/10;
    }
if(rev == m)
 System.out.println ("Given number is palindrome ") ;
else
 System.out.println ("Given number is not palindrome ") ;

  }
}
```

```
Output:
Enter the number: 14141
Given number is palindrome
```

Program to check whether given no is Armstrong or not

```
Import java.util.Scanner;
class WhileDemo7
{
```

```
public static void main (String[] args )
 { int n,sum=0;rem=0,m;
   Scanner sc = new Scanner(System.in);
   System.out.println ("Enter the number") ;
   n = sc.nextInt();
  m=n;
   while(n != 0)
    {
     rem = n%10;
     sum=sum+rem*rem*rem;
     n=n/10;
    }
if(sum == m)
   System.out.println ("Given number is armstrong ") ;
else
 System.out.println ("Given number is not armstrong ") ;

  }
}
```

Output:

```
Enter the number: 153
Given number is armstrong
```

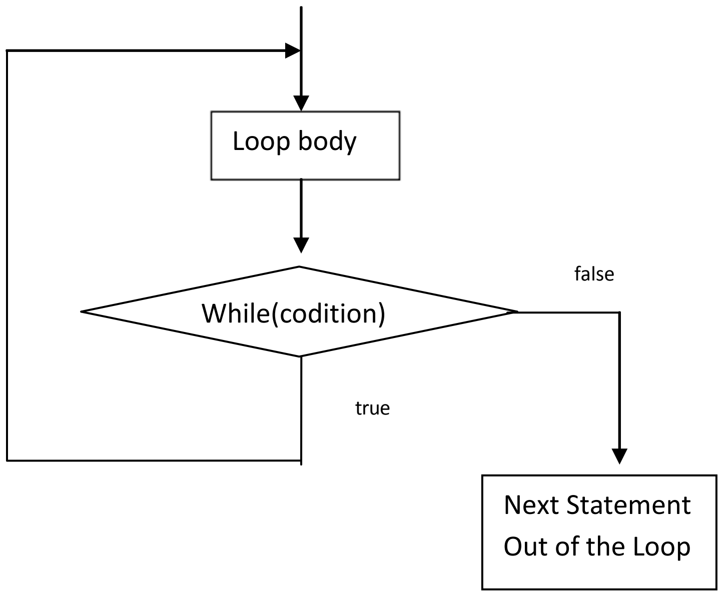
do...while loop

The 'do...while' statement is also used for looping. The body of this loop may contain a single statement or a block of statements. The syntax for writing this loop is:

The do..while statement can be written as

```
 do
     statement;
while(condition);

 do
{ statement;
statement;
}
while(condition);
```



Program to print the numbers from 1 to 10 using do...whi1e loop

```
Import java.util.Scanner;
class DoWhileDemo1
{
public static void main (String[] args )
 {
   int i=1;
   do (
        System.out. printf("%u\t",i) ;
        i=i+1;
       }while(i<=10) ;
 }
}
```

Program to count the digits in any number

```
Import java.util.Scanner;
class WhileDemo7
{
public static void main (String[] args )
 { int n,count=0;rem=0;
   Scanner sc = new Scanner(System.in);
   System.out.println ("Enter the number”) ;
   n = sc.nextInt();
do
  {
   n=n/10;
   count++;
  }
while(n > 0);

System.out.println (“Number of digits : “+count);
}
}
```

for loop

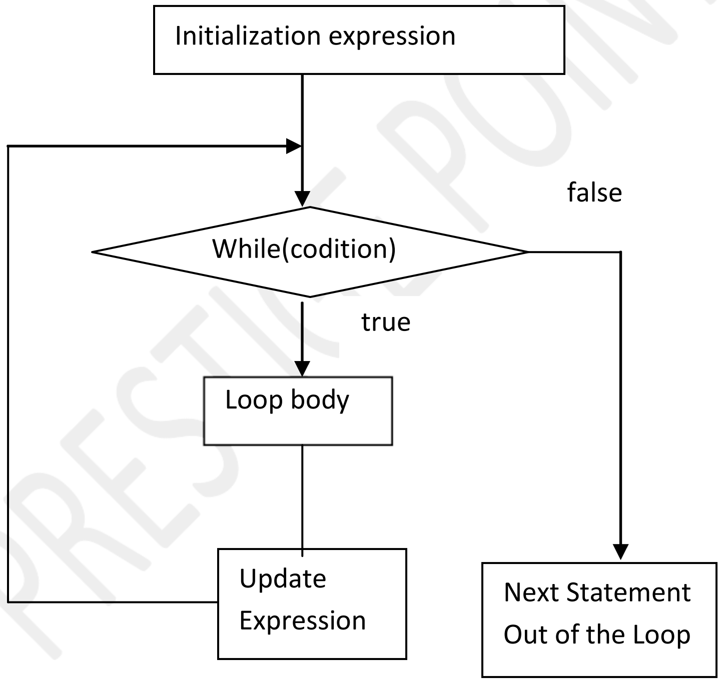
The 'for' statement is very useful while programming in java. It has three expressions and semicolons are used for separating these expressions. The 'for' statement can be written as

```
for(expression1;expression2;expression3)
       statement;
```

```
for(expression1;expression2;expression3)
    { statement;
     statement;
  }
```

The loop body can be a single statement or block of statements.
Expressionl1 is an **initialization expression**, expression2 is a **test expression** or **condition** and expression3 is an **update expression**.
expression1 is executed only once when the loop starts and is used to initialize the loop variables. This expression is generally an assignment expression.
expression2 is a condition and is tested before each iteration of the loop. This condition generally uses relational and logical operators.
expression3 is an update expression and is executed each time after the body of the loop is executed.

Now let us see how this loop works. Firstly the initialization expression is executed and the loop variables are initialized, and then the condition is checked, if the condition is true then the body of loop is executed. After ,executing the loop body, control transfers to expression3(update expression) and it modifies the loop variables and then again the condition is checked, and if it is true, the body of loop is executed. This process continues till the condition is true and when the condition becomes false the loop is terminated and control is transferred to the statement following the loop.



Although the task of while and for loops is same, the for loop is generally used when the number of iterations are known in advance and while loop is used where number of iterations are not known.

Program to print the numbers from 1 to 10 using for loop

```
class ForDemo1
{
public static void main (String[] args )
 {
     int i;
     for(i=1;i<=10;i++)
           {
           System.out.printf("%d\t",i);
            }
   }
}
Output:·
I         2          3         4         5         6         7         8         9         10
```

Program to print· numbers in reverse order wi th a difference of 2

```
 class ForDemo2
{
public static void main (String[] args )
 {
     int k;
    for(k=10;k>=2;k-=2)
           System.out. printf("%d\t",k) ;
  }
 }
```

Multiply two positive numbers without using * operator

```
Import java.util.*;
 class ForDemo3
{
public static void main (String[] args )
  {
    Scanner s = new Scanner(System.in);
     int a,b,i;
      int result=0;
     System.out.println("Enter two numbers to be multiplied ");
      int a = s.nextInt();
       int b = s.nextInt();
       for(i=l;i<=b;i++)
            result=result+a;
System.out.printf("%d * %d %d”,a,b,result);
  }
}
```

Find the sum of this series upto n terms

1+2+4+7+11+16+……

```
Import java.util.*;
 class ForDemo4
{
public static void main (String[] args )
  {
    Scanner s = new Scanner(System.in);
      int i, n, sum=0, term=1;
    System.out.println ("Enter number of terms ") ;
    n = s.nextInt() ;
    for(i=1;i<=n;i++)
        { sum+=term;
          term=term+i;
        }
   System.out.printf ("The sum of series upto %d terms is %d", n, sum) ;
 }
}
```

Program to generate fibonacci series 1,1,2,3,5,8,13,34,55,89

In this series each number is a sum of the previous two numbers

```
Import java.util.*;
 class ForDemo5
{
public static void main (String[] args )
  {
  Scanner s = new Scanner(System.in);
  long x, y;
   int i,n;
  x=0; y=1;
  System.out.println ("Enter the number of terms ") ;
  n= s.nextInt() ;
  System.out.print(x+”     “ + y);
    for(i=1;i<n;i++)
      {
            z=x+y;
            System.out. print (z+ “ “);
            x=y;
            y=z;
            }
        }
}
```

All the three expressions of the for loop are optional. We can omit anyone or all the three expressions in the for loop but in any case the two separating semicolons should always be present.
Expression1 is omitted when the initialization work is done before entering the loop.
expression2 is a condition and if omitted, it is always assumed to be true and so this type of loop will never stop executing. This type of loop is infinite and to avoid it there should be a statement inside the loop that takes the control out of the loop.
expression3 is an update expression and is omitted when it is present inside the body of the loop.

Program to print the sum of digits -of any number using for loop

```
Import java.util.*;
 class ForDemo6
{
public static void main (String[] args )
  {
 Scanner s = new Scanner(System.in);
 int n, sum=0, rem;
System.out.println ("Enter the number ") ;
Int n = s.nextInt() ;
 for ( ; n>0 ;n /= 10)
      { rem=n%10;      /*taking last digit of number*/
        sum+=rem;
     }
System.out.printf ("Sum of digits = %d", sum) ;
}
}
```

We can also have any number of expressions separated by commas. For example, we may want to initialize more than one variable or take more than one variable as loop variable.

Program to print numbers using for loop

```
class ForDemo7
{
public static void main (String[] args )
  {
 int i, j ;
for(i=0,j=10; i<=j; i++, j--)
    System.out. printf("i = %d j %d",i, j);
   }
}
```

Nesting of loop:

When a loop is written inside the body of another loop, then it is known as nesting of loops. Any type of loop can be nested inside any other type of loop. For example a for loop may be nested inside another for loop or inside a while or do while loop. Similarly while and do while loops can be nested.

Program to understand nesting in for loop

```
class ForDemo8
{
public static void main (String[] args )
 {
   int i,j;
   for(i=1;i<=3;i++)
     {        //outer loop
       System.out.printf("I = %d\n",i);
       for(j=1;j<=4;j++)       /*inner loop*/
           System.out.printf ("j = %d\t", j);
     System.out. println();
     }
  }
}
```

Program to print armstrong numbers from 100 to 999

```
class ArmstrongSeries
{
public static void main (String[] args )
 {
int num,n,cube,d,sum;
System.out. println ("Armstrong numbers are : ") ;
for(num=100;num<=999;num++)
 { / *outer 100p* /
 n=num; sum=0;
 whi1e(n>0) {
                /* inner loop* /
     d=n%10;
     n/=10;
     cube=d*d*d;
     sum=sum+cube;
       } //  End .of while loop*1
     if (num==sum)
```

```
        System.out.print(num+" ") ;
      } // end of for loop
    }
}
```

Program to find the sum of digits of a number until the sum is reduced to 1 digit.
For example: 538769->38->11->2

```
Import java.util.*;
 class  SumOfDigitsReducedToOneDigit
  {
   public static void main (String[] args )
  {
    Scanner s = new Scanner(System.in);
    long num;
    int dig, sum;
   System.out.println ("Enter a number ") ;
    num = s.nextInt();
    do {
        for(sum=0;num!=0;num=/10)
            {
              dig=num%10;
              sum+=dig;
           }
        System.out.printf("%d\t",sum);
        num=sum;
    }while(num/10!=0) ;
}
}
```

break statement

break statement is used inside ,loops and switch statements. Sometimes it becomes necessary to come out of the loop even before the loop condition becomes false. In such a situation, break statement is used to terminate the loop. This statement causes an immediate exit from that loop in which this statement appears. It can be written as

```
break;
```

When break statement is encountered, loop is terminated and the control is transferred to the statement immediately after the loop. The break statement is generally written along with a condition. If break is written inside a nested loop structure then it causes exit from the innermost loop.

Program to understand the use of break

```
class UseOfBreak
 {
  public static void main (String[] args )
 { int n;

    for(n=1;n<=5;n++)
     {
        if (n==3)
          { System.out.println ("I understand the use of break");
            break;
          }
      System.out.printf ("Number = %d\n", n) ;
    }
    System.out.println ("Out of for loop");
 }
}
```

Program to find whether a number is prime or not

```
import java.lang.Math;
import java.util.*;
class  PrimeNoCheck
 {
  public static void main (String[] args )
 {
   Scanner s = new Scanner(System.in);
  int i, num;
 boolean flag=true;
 System.out.println ("Enter a number ") ;
 num = s.nextInt();
    for(i=2;i<Math.sqrt(num);i++)
     {
      if (n%i==0 )
       {
       System.out.printf("%d is not prime ",num);
       flag=false;
       break;
       }
    }
  if (flag)
      System.out.printf("%d is prime ",num);
 } }
```

continue statement

The continue statement is used when we want to go to the next iteration of the loop after skipping some statements of the loop. This continue statement can be written simply as

continue;

It is generally used with a condition. When continue statement is encountered all the remaining statement (statements after continue) in the current iteration are not executed and the loop continues with the next iteration.

The difference between break and continue is that when break is encountered the loop terminates and the control is transferred to the next statement following the loop, but when a continue statement is encountered the loop is not terminated and the control is transferred to the beginning of the loop.

In while and do-while loops, after continue statement the control is transferred to the test condition and then the loop continues, whereas in for loop after continue statement the control is transferred to update expression and then the condition is tested.

Program to understand the use of· continue statement

```
class UseOfContinue
 {
  public static void main (String[] args )
 {
  int n;
  for(n=1;n<=5;n++)
   {
      if(n==3)
       {
        System.out.println ("I understand the use of continue");
        continue;
       }
      System.out.printf ("Number = %d\n", n) ;
   }
  System.out.println ("Out of for loop\n");
 }
}
```

Program to find the sum and average of 10 positive integers

```
import java.util.*;
 class SumAndAverageOfTenPositiveNumbers
 {
  public static void main (String[] args )
 {
   Scanner s = new Scanner(System.in);
   int i=1, n, sum=0;
 float avg;
 System.out,.println ("Enter 10 positive numbers");
 whi1e(i<=10) {
     System.out.print ("Enter number");
      n= s.nextInt();
      if (n<0)
       {
        System.out.println("Enter only positive numbers\n");
       continue;
       }
    sum+=n;
     i++;
 }
avg=sum/10.0f;
System.out.printf ("Sum = %d     Avg = %f", sum, avg) ;
}
}
```

switch
This is a multi-directional conditional control statement. Sometimes there is a need in program to make choice among number of alternatives. For making this choice, we use the switch statement. This can be written as

```
switch(expression)
   {
 case constant1: statement
case constant2: statement
case constantN: statement
default statement
 }
```

The "expression" following the switch keyword can be any java expression that yields an byte,short,int,char, value. **From Java 1.5 enum is also valid argument for switch**. **From Java 1.7 String is also valid argument for switch statement.**

We can't use floating point or double and long as switch argument.
Multiple constants in a single case are not allowed; each case should be followed by only one constant. Each case can be followed by any number of statements. It is also possible that a case has no statement under it.

Firstly the switch expression is evaluated, then the value of this expression is compared one by one with every case constant. If the value of expression matches with any case constant, then all statements under that particular case are executed. If none of the case constant matches with the value of the expression then the block of statements under default is executed. 'default' is optional, if it is not present and no case matches then no action takes place. These cases and default can occur in any order.

Program to perform arithmetic calculations on integers

```
import java.util.*;
 class AritmeticOperationUsingSwitch
  {
   public static void main (String[] args )
  {
    Scanner s = new Scanner(System.in);
    char op;
    int a,b;
    System.out.println ("Enter number operator and another number ") ;
    a=s.nextInt();
   op=s.next().charAt(0);
   b=s.nextInt();
   switch(op)
       {
      case '+': System.out.printf ("Result = %d\n", a+b);
                    break;
       case '-': System.out.printf ("Result = %d\n",a-b);
                     break;
       case '*': System.out.printf("Result =%d\n",a*b);
                    break;
        case '/': System.out.printf("Result = %d\n",a/b);
                     break;
      case '%': System.out.printfl"Result = %d\n",a%b);
                    break;
        default: System.out.printf ( "Enter valid operator");
      } /*End of switch */
   } //End of main
}
```

Program to find whether the alphabet is a vowel or consonant

```
 import java.util.*;
 class VowelAnd consonantCheck
  {
   public static void main (String[] args )
  {
    Scanner s = new Scanner(System.in);
    char ch;
   System.out.println ("Enter an alphabet ") ;
   ch = s.next().charAt(0);
   switch(ch) {
  case 'a' :
```

```
 case 'e' :
  case I :
  case 'o' :
  case 'u' : System.out.println ("Alphabet is a vowel" ) ;
                 break;
  default: System.out.println ("Alphabet is a consonant" );
  }
}
}
```

Program to check whether a date is valid or not

```
import java.util.*;
 class DateValidation
  {
   public static void main (String[] args )
  {
    Scanner s = new Scanner(System.in);
    int d,m,y;
    boolean flag=true, isleap=false;
    System.out.println ("Enter date (dd/mm/yyyy) ") ;
    d = s.nextInt();
     m=s.nextInt();
     y=s.nextInt();
     if (y%4==0 && y%100! =0 II y%400==0)
           isleap=true;
     if (y<=1850 II y>=2050)
             flag=false;
     else if (m< 1 II m>12)
              flag=false;
               else if (d<1)
                       flag=false;
                          else if (m==2)     /*check for number of days in February* /
                                  { if (d>28)
                                        flag=false;
                                     if(d==29&&isleap)
                                            flag=true;
                                   }
                              else if(m==4 || m==6 || m==9 || m==11)         /*Check days in april, june, sept, nov */
                                     {
                                       if (d>30) flag=false;
                                     } else if (d>31)
                                              flag=false;
if (flag==false)
          System.out.println("Not a valid date\n");
else
         System.out.println("Valid Date \n");
    }/*End of main() */
}
```

Write a program to find difference of two dates in years, months and days. Assume that the dates are entered in valid range and that the first date falls before second date.

```
import java.util.*;
 class DateDifference
  {
   public static void main (String[] args )
  {
    Scanner s = new Scanner(System.in);
    int d1, d2, d, m1, m2, m, y1, y2, y;
     System.out.println ("Enter first date (dd/mm/yyyy) ");
    d1 = s.nextInt();
    m1=s.nextInt();
   y1=s.nextInt();
   System.out.println ("Enter second date (dd/mm/yyyy) ");
    d2 = s.nextInt();
    m2=s.nextInt();
   y2=s.nextInt();
   if(d2<d1)
        {
         if (m2==3)
            {
              if(y2%4==0&& y2%100!=0 II y2%400==0) /*check for leap*/
                        d2=d2+29;
             else d2=d2+28;
           }
        else
            if (m2==5 || m2==7 || m2==10 || m2==12)
                       d2=d2+30;
            else
                        d2=d2+31;
   m2=m2-1;
   }
  if (m2<m1)
      { y2=y2-1;
       m2=m2+12;
 }
  y=y2-y1;
 m=m2-m1;
 d=d2-d1;
 System.out.println ("Difference of the two dates is ") ;
 System.out.printf("%d years %d months %d days\n",y,m,d);
  }          /*End of main()*/
}
```

We have assumed that the second date falls after first date, so y2 will always be greater or equal to y1 and y will always come out positive.
It is possible that m2 is less than m1, in this case m will come out negative, and if d2 is less than d1 then d will come out to be negative. So before calculating y, m and d we should make sure that m2 is greater than or equal to m1 and d2 is greater than or equal to d1.
If d2 is less than d1, then we borrow a month from m2 and add the days of that month to d2. Now since all the months have different number of days, the days added to d2 will depend on the month borrowed. We'll always borrow a month that is before m2. For example if m2 = 5(May), then we'll borrow month April from m2, so we'll add 30 to d2.

```
            d2 = d2+30;                              m2 = m2-1;
```

If m2 = 3(March), then we'll borrow month February from m2, so we'll add 28 or 29 (if it is leap year) to d2.

```
        d2 = d2+28;                m2 = m2-1;
```

In the program we've done this through nested if else statements.
If m2 is less than m1, then we borrow 1 year{12 months) from y2 and add it to m2

```
        m2=m2+12;                 y2=y2-1;
```

Now m2 will ,become greater than m1.

Write a currency program, that tells you how many number of 100, 50, 20, 10, 5, 2 and 1 Rs notes will be needed for a given amount of money. For example if the total amount is Rs 545 then five 100 Rs notes, two 20 Rs note and one 5 Rs note will be needed. This sort of program can be used in ATM machines,

```
import java.util.*;
 class CurrencyProgram
  {
   public static void main (String[] args )
  {
    Scanner s = new Scanner(System.in);
    int n, choice, notes;
    System.out.println ("Enter the total amount in Rs “);
   n= s.nextInt();
   System.out.println ("Enter the value of note from which u want to begin “ );
  choice = s.nextInt();
  switch (choice)
  {
```

```
default: System.out.println ("Enter only valid values “);
              break;
case 100: notes=n/100;
                 System.out.printf("Number of 100 Rs notes %d\n",notes);
                 n=n%100;
 case 50: notes=n/50;
                System.out.printf("Number of 50 Rs notes %d\n",notes);
                  n=n%50;
  case 20: notes=n/20;
                 System.out. printf("Numbel. of 20 Rs notes %d\n",notes);
                   n=n%20;
   case 10: notes=n/10;
                 System.out.printf ("Number of 10 Rs notes %d\n" ,notes) ;
                   n=n%10;
      case 5: notes=n/5;
                   System.out.printf ("Number of 5 Rs notes %d\n" ,notes) ;
                      n=n%5;
        case 2: notes=n/2;
                      System.out.printf ("Number of 2 Rs notes %d\n", notes) ;
                      n=n%2;
         case 1: notes=n/1;
                      System.out.printf("Number of 1 Rs notes %d\n",notes);
        }
}
}
```

Program to find the LCM and HCF of two numbers

```
import java.util.*;
 class LCMandHCF
  {
   public static void main (String[] args )
  {
    Scanner s = new Scanner(System.in);
   int x,y,a,b;
   System.out.println ("Enter two numbers :”);
   x = s.nextInt();
  y = s.nextInt();
  a=x;
  b=y;
  while(a!=b)
     {
      if (a<b) a=a+x;
  else b=b+y;
}
Sytstem.out.printf("LCM of %d and %d is %d\n",x,y,a);
 a=x;
 b=y;
while(a!=b) {
   if(a>b)
          a=a-b;
  else b=b-a;
}
System.out. printf ("HCF of %d and %d is. %d\n", x, y, a) ;
}
}
```

Pyramids

```
*            1            1            1                2            1
**           22           12           2 3              34           01
***          333          123          4 5 6            456          101
****         4444         1234         7 8 9 10         5678         0101
*****        55555        12345        11 12 13 14 15   678910       10101
   (a)         (b)          (c)          (d)              (e)          (f)
```

The program for pyramid (a)

```
import java.util.*;
 class PyramidA
  {
   public static void main (String[] args )
  {
    Scanner s = new Scanner(System.in);
    int i,j,n;
   System.out.println ("Enter n ") ;
   n= s.nextInt();
for(i=1;i<=n;i++)
  {
     for(j=1;j<=i;j++)
         System.out.print("*
         ");
     System.out.prinln("\n");
   }
}
}
```

Here the outer for, loop is for number of lines and the inner loop is for number of stars in each line. We can see that the number of stars is equal to the line number, hence the inner loop will execute once for first line, twice for second line, thrice for third line and so on…….

In the above program **if we print the value of i, then we'll get the pyramid (b)**, and **on printing the value of j we'll get the pyramid(c)**. **For the pyramid (d)** we'll take a variable p = 1, and write the print statement as printf("%3d",p++);
For pyramid (e) we'll print the value of i+j, for pyramid (f) we'll print 1 if (i+j) is even and print 0 if (i+j) is odd.

```
5                    5
54                   44
543                  333
5432                 2222
54321                11111
(g)                    (h)
```

For pyramid (g) we have to print (n+1-j) and for pyramid (h) we have to print (n+1-i).
These two pyramids can also be written by reversing the loops and then printing the values of i and j
For example code for pyramid (g) can be written as

```
for(i=n;i>=1;i- -)
  {
    for(j=n;,j>=i;j- -)
       System.out. printf ("%3d", j) ;
   System.out. printf("\n") ;
 }
```

```
* * * * *      55555      12345      54321      11111         *           *
* * * *        4444       1234       5432       2222         **          * *
* * *          333        123        543        333         ***        * * *
* *            22         12         54         22         ****      * * *        *
 *             1          1          5          1         *****    * * * *         *
(i)             (j)       (k)        (l)        (m)         (n)            (o)
```

The code for **pyramid (i)** is

```
   for(i=n;b=1;i- -)
        { for(j=1;j<=i;j++)
            System.out.print("* ");
        System.out.println() ;
```

For **pyramids (j), (k), (1), (m)** we'll print values of **i, j, (n+1-j), (n+1-i) respectively**. The pyramids (n) and (m) can also be printed by reversing both the loops and then printing i and j.

For pyramid (n), we have to print spaces before printing stars. The code for it is

```
for(i=1;i<=n;i++)            /*loop for number of lines*/
    { for(j=1;j<=n-i;j++)
        System.out.print("  ");      /*loop for printing spaces* /
          For(j=1;j<=i;j++)
            System.out.print("*");             /*loop for printing stars* /
     System.out.println() ;                    /*for next line of pyramid*/
   }
```

The code for **pyramid (0)** is same as this one, only a space is given after star in the print statement

```
      *             *             1             1             5             *********
     ***          * * *          123           232           545             *******
    *****       * * * * *       12345         34543         54345             *****
   *******    * * * * * * *    1234567       4567654       5432345             ***
                              123456789     567898765     543212345             *
   (P)            (q)            (r)           (s)           (t)               (u)
```

The code for pyramid (p) is

```
 for (i=1; i<=n; i++)
{
for(j=1;j<=n-I;j++)
 System.out. print(" ");
 for(j=1;j<=i;j++)
       System.out. print(" * ");
 for(j=1;j<i;j++)
     System.out. print(" * ");
 System.out.println();
}
```

The code for **pyramid (q)** will be the same, only the range of first for loop for spaces will be **from 1 to 2*(n+1-i)** and there will be a space after star in the print statements.

The loops for pyramids (r), (s) and (t) will be same as that of pyramid (p), but here we'll take a variable p and print its value.
For pyramid (r) we'll initialize the value of p with 1 each time before second inner for loop; and then print the value of p++ in the last two for loops. .
For pyramid (s) we'll initialize the value of p with i before second inner for loop, and then print the value of p++ in second for loop. After this the value of p is decreased by 1 and then the value of - -p is printed in the third for loop.

For **pyramid (t)** we'll initialize the value of p with n before second inner for loop, and then print the value of p- - in second loop and p++ in third loop. The value of p has to be increased by 2 before third for loop.

The code for pyramid (t) is

```
 for (i=l; i<=n;i++)            / *loop for number of lines in pyramid* /
   { for(j=l;j<=n-i;j++ ) /* loop for spaces (first part) */
       System.out.print(" ");
       p=n;
       for(j=l;j<=i;j++)
         System.out. printf ("%d" ,p- -);
       p=p+2;
      for (j =1 ; j<i;j++)
        System.out.printf("%d",p++) ;
     System.out.println();
   }
```

The code for inverted **pyramid (u)** is

```
for(i=l;i<=n;i++)
   {
    for(j=l;j<=i;j++)
       System.out.print(" ");
    for ( j =1; j <= (n- i) ; j ++ )
       System.out.print("*");
    for(j=1;j<(n-i);j++)
       System.out.print("*");
   System.out.println() ;
 }
```

Programming Exercise

- Write a program to print prime numbers from 1 to 99.
- Write a program to enter a number and find the reverse of that number.
- Input a number and a digit and find whether the digit is present in the number or not, if present then count the number of times it occurs in the number.
- Write a program to accept any number n and print the sum of square of all numbers from 1 to n.
- Write a program to accept any number n and print the cube of all numbers from 1 to n which are divisible by 3.
- Write a program to accept any six digit number and print the sum of all even digits of that number and multiplication of all odd digits.
- Write a program to find out the value of x raised to the power y, where x and y are positive integers.
- Write a program to accept any number up to six digits and print that in words. For example-1265 = one two six five

- Write a program to enter a number and test whether it is a fibonacci number or not.
- Write a program to print all the pythagorean triplets less than 50. Any three numbers x, y, z are called pythagorean triplets if $x < y < z$ and $x^2+y^2 = Z^2$
- Find the sum of these series up to n terms where x is an integer entered by the user.

  1 + 2 + 4 + 7 + 11 + 16 + ...
  1+ 11 + 111 + 1111 +.....
  $x + x^2 + x^3 + x^4 +$ .. ..