Unit 4

Control Statements

Control Statements

↳ The statements that control or alter the flow of execution of the program are known as control statements. In Java, control statements are categorized into following groups:-

① Sequential statements
② Selection (Branching) statements:- if, else-if, nested if-else, switch, if-else-if ladder
③ Iteration (Looping) statements :- while, do-while, for, foreach
④ Unconditional (Jumping) statements :- break, continue, ~~goto~~, return

① Sequential Statements:

↳ Sequential statements are executed one instruction after another or in order in which they occurs in the program.

Syntax:

```
Statement 1;
Statement 2;
Statement N;
```

② Selection (Branching) Statements (Conditional/Decision making)

↳ They are control statement that executes statements depending upon condition.

↳ The condition may be either true or false.

↳ If condition is true, set of steps are executed; otherwise another set of steps are executed

↳ Following types of statement in Java :

① Simple if statement
② if-else statement
③ Nested if-else statement
④ if-else-if ladder statement
⑤ Switch statement

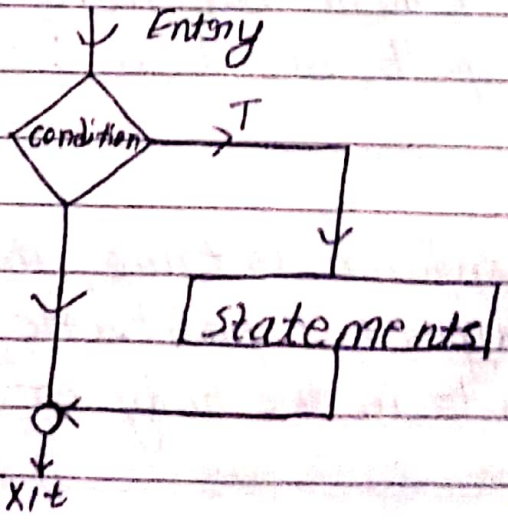
1) Simple if statement

↳ Simple selection statement that executes statement(s) when condition is true.

↳ If condition is false, control enters next part of the program

Syntax :

```
if (condition)
{
  statements;
}
```



// program to calculate commission if sale amount >= 20000
// (commission will be 15% of total sale)

```
import java.util.*;
    class IfTest
{
    public static void main(String [] Jargs)
{
    float sale, comm=0;
    Scanner Sc = new Scanner(System.in);
    System.out.println("Enter total sale amount");
    sale = Sc.nextFloat();
    if (sale >= 20000)
    {
        comm = (sale x 15)/100;
    }
    System.out.println("commision amount:"+
                                    comm);
}
}
```

2) If - - - else statement:

↳ Selection control statement that decides the execution path based on the condition is true or false.

↳ If the condition is true, then the statements in the body of if part are executed otherwise, Statements in the body of else part are executed.

Syntax:

```
if (condition)
{
    statement_1;
}
else
{
    statement_2;
}
```

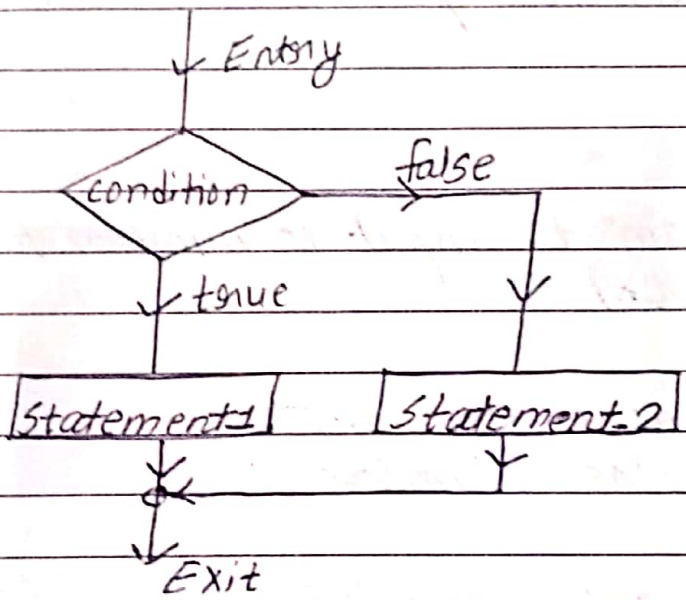


fig:- flowchart for if--else statement

/*Program to entre a number and check whether it is odd or even*/

```
import Java.util.*;
  Class Number
{
    public static void main (String [] args)
{
    int a;
Scanner Sc = new Scanner(System.in);
```

```
System.out.println("Enter a number");
    a = sc.nextInt();
if(a%2==0)
 {
  System.out.println("The number is even:"+a);
 }
else
 {
  System.out.println("The number is odd:"+a);
 }
}
```

/* Program to input to number to find greater number */

```
import java.util.*
 class Number
{
  public ~~void~~ static void main(String[] args)
{
     int a, b;
 Scanner sc = new Scanner(System.in);
 System.out.println("Enter first number");
   a = sc.nextInt();
System.out.println("Enter second number");
    b = sc.nextInt();
 if(a>b)
{
System.out.println(" a is greater number:"a);
}
else
```

```
{
System.out.println("b is greater number:");
}
}
}
```

3) Nested if--else statement

↳ An if statement can be inside another if statement.

↳ An entire if---else statement within the body of if part or else part of element of another if--else statement is called nested--ifelse statement.

Syntax:

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

/* Program to input age and weight of a person and check he/she is eligliable to donate blood or not. (hint: age > 18 and weight >=50 */)

```
import java.util.*;
class NestedIf
{
  public static void main(String [] args)
  {
     int age, weight;
    Scanner sc. new Scanner (System.in);
    System.out.println("Enter age and weight");
     age = sc.nextInt();
     weight = sc.nextInt();
   if (age > 18)
    {
     if (weight >= 50)
    {
     System.out.println("You are eligliable
              for ~~voting~~ donate blood"); }
   }else { System.out.println("You are not
              eligliable for donate blood"); }
}
else
   System.out.println("You are not eligliable
              for ~~voting~~, donate blood");
}
}
```

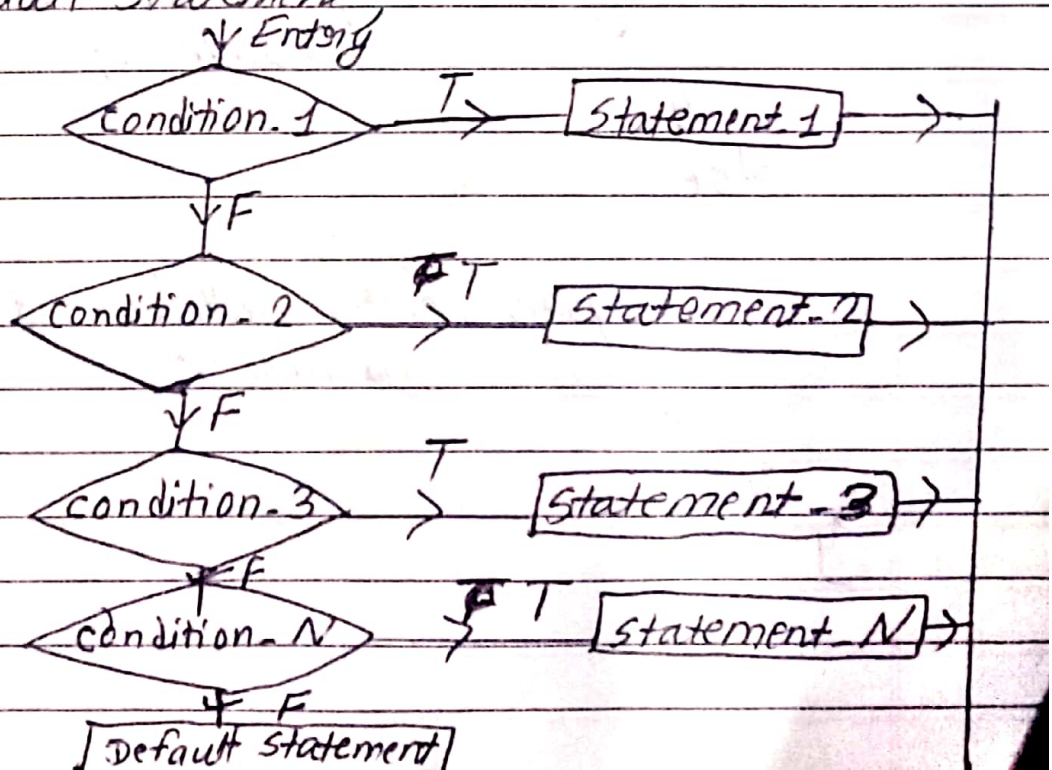
4) if--else if statement (if-else--if ladder)

↳ Type of conditional statement that allows to check multiple condition and executes different code blocks based on those conditions.

↳ It can have multiple branches.

Syntax:

```
if (condition-1)
    statement-1;
else if (condition-2)
    statement-2;
else if (condition-3)
    statement-3;
- - - - - - - - -
else if (condition-N)
    statement-N;
else
    default statement;
```



// Program to find greatest number among three Numbers

```
import java.util.*;
class GreatestNum
{
  public static void main(String [] args)
{
    int a, b, c;
  Scanner sc = new Scanner (System.in);
  System.out.println("Enter three numbers");
      a = sc.~~Int~~nextInt();
      b = sc.nextInt();
      c = sc.nextInt();
    if (a>b && b>c)
    {
      System.out.println(a+" is the greatest number");
    }
  else if (b>c && c>a)
    {
      System.out.println(b+"is the greatest
                    number");
    }
else
    {
      System.out.println(c+"is the greatest
                    number");
    }
  }
}
```

5) Switch statement

↳ Type of conditional statement that allows to check a variable against multiple possible value i.e case values.

↳ Executes different code blocks based on which case value matches with variable.

↳ If one case value matches, then default block statement is executed.

Syntax:

```
switch (variable)
{
    case value1:
        statement1;
        break;
    case value 2:
        statement2;
        break;
    - - - - - - - -
    case value N:
        statement N;
        break;
    default:
        default statement;
        break;
}
```

/* Program to display name of daggy according to user choice (1 - 7) */

```
import java.util.*;
  Class SwitchTest
{
  public static void main (String [] args)
{
    int choice;
Scanner sc = new Scanner (System.in)
System.out.println("Enter your choice (1-7)");
   choice=sc.nextInt();
  Switch (choice)
    {
      case 1:
          System.out.println("Sunday");
          break;
     case 2:
         System.out.println("Monday");
          break;
    case 3:
      System.out.println("Tuesday");
      break;
  case 4:
    System.out.println("Wednesday");
    break;
 case 5:
   System.out.println("Thursday");
   break
```

```
case 6:
    System.out.println("Friday");
    break;
case 7:
    System.out.println("Saturday");
    break;
default:
    System.out.println("Sorry your choice is
                        wrong !!");
    break;
}
}
}
```

/* Program to perform basic arithmetic operation based on user choice.

```
import java.util.*;
class ArithmeticOperators
{
    public static void main(String [] args)
        int a b result, choice;
        Double result;
    Scanner sc = new Scanner(System.in);
    System.out.println("**** menu ****");
    System.out.println("1. Addition\n 2. Subtraction
    \n 3. Multiplication\n 4. Division\n 5. Modulus\n");
    System.out.println("Enter your choice (1-5):");
        choice = sc.nextInt();
    System.out.println("Enter two number:");
        a = sc.nextInt();
        b = sc.nextInt();
```

```
switch(choice)
{
    case 1:
        result = a+b;
    System.out.println("The sum is = "+result);
        break;
    case 2:
        result = a-b;
    System.out.println("The sub is = "+ result);
        break;
    case 3;
    System.out.println("The Mul is = "+result);
        break;
    case 4:
    System.out.println("The Quotient is = "+result);
        break;
    case 5:
    System.out.println("The Remainder is = "+result);
        break;
    default:
    System.out.println("Your choice is invalid!");
        break;
    }
}
}
```

③ Iteration (Looping) statements:

↳ Looping is the process of executing the some program statement or block of statements repeatedly for specified no. of times or until given condition is

satisfied.

↳ Looping statements in java are:

① While loop
② do--while loop
③ for loop
④ Foreach loop

① While Loop

↳ Looping statement that executes program statement repeatedly until given condition is true.

↳ Also known as entry-controlled or pre-test loop because condition is checked initially.

↳ Statements are not executed if the condition is false initially.

Syntax:

```
initialization;
while (condition)
{
    statement(s);
    increment decrement;
}
```

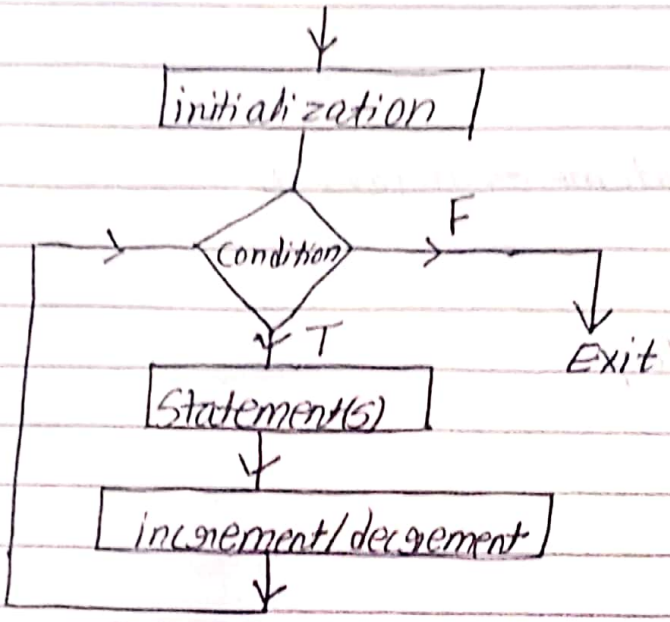


fig: - Flowchart ~~for loop~~ of while loop

Example:

```
// Program to display numbers from 1 to 20.

class WhileLoop
{
public static void main (String args[])
{
    int i = 1;
    while (i <= 20)
    {
    System.out.println(i);
      i++;
}
}
}
```

// Program to find sum of natural numbers from 0 to 100

```
class NaturalNumbers
{
  public static void main (String args [])
  {
     int i = 0;
     int sum = 0;
    while (i <= 100)
    {
       sum += i;        // sum = sum + i;
       i++;
    }
    System.out.println("The sum is" + sum);
  }
}
```

(b) do-- while loop

↳ Looping statement that executes program statement repeatedly until given condition is true.

↳ Also called post-test or exit controlled loop

↳ Statement(s) are executed once at first even condition is ~~true~~ false.

Syntax :

```
initialization
do
  {
    Statement(s);
```

increment/decrement;
}
while (condition);

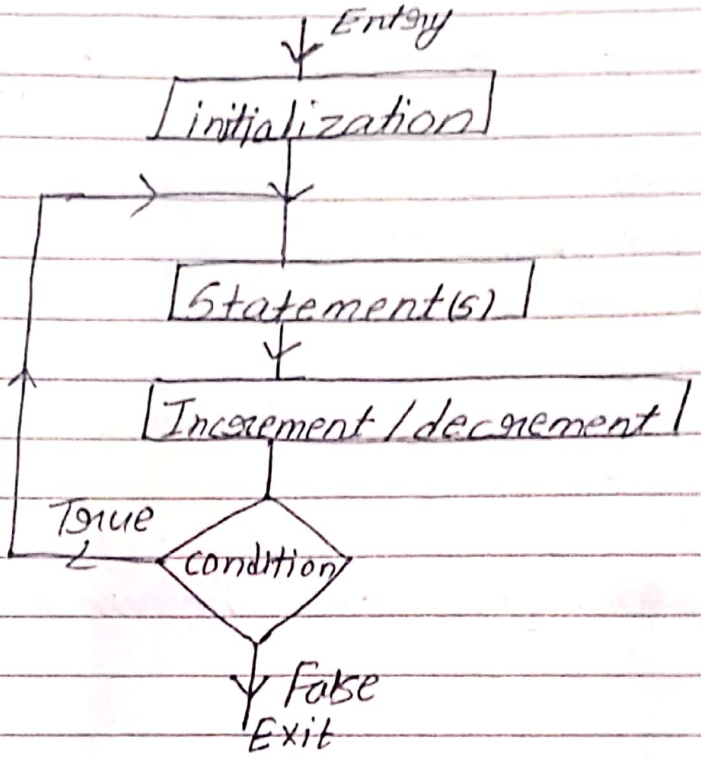


fig:- flowchart for do-while loop.

Example:

//Program to display series 5, 9, 13, upto 10th term

```
class DoWhileLoop
{
    public static void main (String args[])
    {
        int i = 10, n = 5;
        do
        {
            System.out.println(n);
            n = n+4;
```

i++;
} while (i>=i<= 10);
}
}

② For Loop

↳ Executes statement(s) repeatedly until given condition is true.

↳ Also pre-test looping statement.

↳ Consists of 3 parts: initization, condition and increment or decrement.

Syntax:

```
For(initization; condition; increment/decrement)
{
    statement(s);
}
```

Example

```
For(i=1; i<=10; i++)
{
    System.out.println(i);
}
```

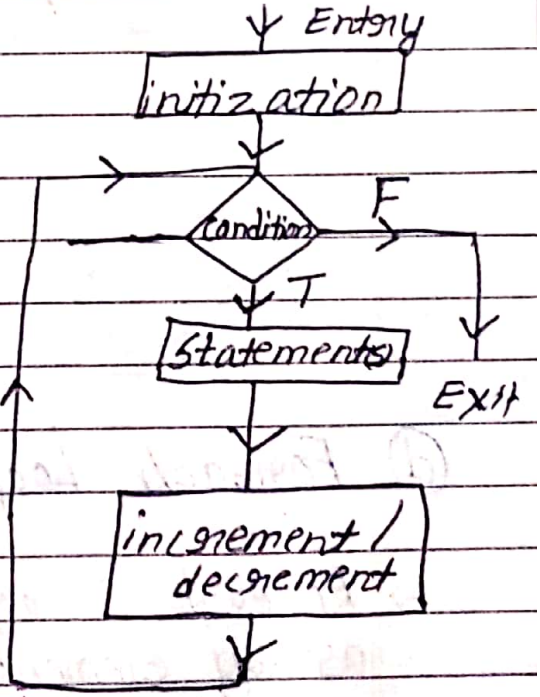


fig:- flowchart for for loop

// To display fibonacci series 0, 1, 1, 2, 3, 5, 8... nth terms.

```
import java.util.*;
class FibonacciSeries
{
    public static void main(String args[])
    {
        int i, n;
        int t1=0, t2=1;
        int nextTerm = t1+t2;
        Scanner sc = new Scanner(System.in)
        System.out.println("Enter how many terms:");
        n = sc.nextInt();
        System.out.println(t1+" "+t2); // Display first & second term
        for(i=3; i<=n; i++)
        {
            System.out.println(" "+nextTerm);
            t1 = t2;
            t2 = nextTerm
            nextTerm = t1+t2;
        }
    }
}
```

(d) Foreach Loop

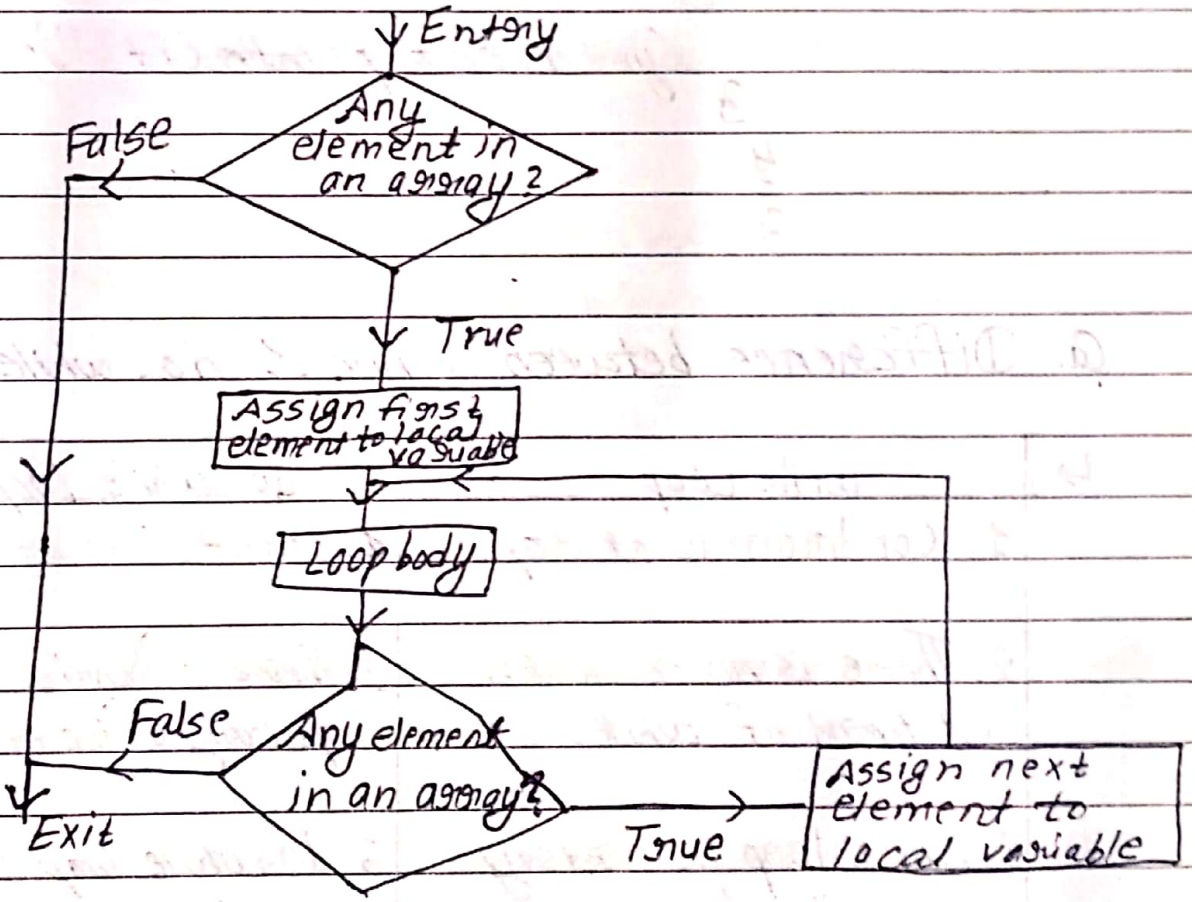
↳ Foreach is looping statement used to access array elements in a array.

↳ It travels the arrays or collection until the last element.

↳ For each element, it stores element in the variable and executes the body of the foreach loop.

Syntax:

```
for (type var : array)
{
    statements;
}
```



∴ fig:- flowchart for foreach loop

// Program to illustrate foreach loop:

```
class Program
{
    public static void main(string [] args)
    {
        int [] num = {2, 4, 6, 8, 10}; //array
                                declaration & initialization
        for (int i : num)
        {
            System.out.println(i + " ");
        }
    }
}
```

Q. Difference between while & do-while loop:-

↳	while Loop	do-while loop
	1. Condition is at top.	1. Condition is at the bottom.
	2. There is no semicolon at the end of while.	2. There is semicolon is compulsory at the end do while
	3. While loop is entry controlled loop.	3. do while loop is exit controlled loop.
	4. ~~Syntax~~: Syntax: while (condition) { Statement(s); }	4. Syntax: do {Statement(s);} while (condition);

Infinite loop

↳ A loop never terminates is called infinite loop.

↳ Executes statement(s) repeatedly which does not meet any ending point.

Example:

```
public class Test
{
    public static void main(String[] args)
    {
        int i;
        for(i=1; i>=0; i++)
            System.out.println("Value of i = "+i);
    }
}
```

④ Unconditional (Jumping) statements:

↳ Used to jump execution of programs statements from one place to another.

↳ Executes some program statements repeatedly or skip some program statements.

↳ 3 types of jumping statements:

a) break
b) continue
c) ~~goto~~ return

a) break statement:

↳ Used to break, the normal flow of program statement execution in loop and switch case statements.

↳ Allows to exit from loop or switch statement as soon as certain condition is satisfied.

↳ When break is encounted, remaining part of loop or switch statement is skipped and control passed to the next statement after loop.

Syntax:

```
break;
```

Example:

```
for (int i = 1; i<=5; i++)
{
    if (i == 4)
        break;
    System.out.println(i);
}
```

o/p 1 2 3

b) Continue statement:

↳ Used to continue the flow of execution in loop skipping the iteration in the loop as soon as.

↳ When Continue is encountered in the loop, then that particular iteration is skipped and loop will be continued with next iteration.

Syntax:

```
continue;
```

Example:

```
class program
{
  public static void main(String [] args)
{
  for (int i = 1; i <= 5; i++)
   {
     if (i == 4)
      continue;
    System.out.println(i);
   }
}
}
```

O/p
1
2
3
5

c) return statement:

↳ It returns the value of expression from called method to calling location.

Syntax:

```
return expression;
```

// Program to display sum of two number

```
import java.util.*;
Class Sum
{
  int a, b;
  Void setData(int x, int y)
  {
      a = x;
      b = y;
  }
  int calculate()
  {
    int result;
    result = a+b;
    return result;
  }
}
  Class ReturnStatement{
    public static void main(String [] args)
    {
      int num1, num2;
      Sum obj = new Sum();
      Scanner sc = new Scanner (System.in);
      System.out.println("Enter two numbers:");
          num1 = sc.nextInt();
          num2 = sc.nextInt();
        Object.setData(num1, num2);
    System.out.println("Area =" + obj.@Override Area());
    }
}
```

/* program to read a number and check whether whether it is palindrome or not. (Palindrome no is is equal to reversing its digits eg. 121, 11, 252 etc) */

```
import java.util.*;
class Palindrome
{
    public static void main (String [] args)
    {
        int num, rev=0; digit, temp;
        Scanner sc = new Scanner (System.in)
        System.out.println ("Enter a number :");
        num = sc.nextInt ();
        temp = num;
        while (num != 0)
        {
            digit = num%10;
            rev = rev *10 + digit;
            num = num/10;
        }
        if (temp == rev)
            Console.WriteLine (temp + "is palindrome
            System.out.println number");
        else
            Console.WriteLine (temp +" is not palindrome
            system.out.println number");
    }
}
```

Reversing code

/* Write Java Program to display multiplication table of 'n' number */

```
import java.util.*;
class Multiplication
{
    public static void main (String [] args)
    {
        int n, i;
    Scanner sc = new Scanner (System.in);
    System.out.println ("Enter number:");
      n = sc.nextInt();
    for ( i = 1; i<=10; i++)
    {
      System.out.println (n+"*"+i+"="+n*i);
    }
  }
}
```

/* Program to check the entered number is armstrong number or not */

```
import java.util.*;
  class ArmStrong
{
    public static void main(String [] args)
{
    int n, arm = 0, rem, a;
  Scanner sc = new Scanner (System.in);
  System.out.println ("Enter n number");
    n = sc.nextInt ();
```

```
a = n;
while (n > 0)
{
    rem = n % 10;
    arm = (rem * rem * rem) + arm;
    n = n / 10;
}
if (a == arm)
    System.out.println(temp + " is a palindrome number");
else
    System.out.println(temp + " is not a palindrome number");
```

System.out.println("a+" is a arm strange number");

else

System.out.println("a+" is a not arm strange number");

}

}