Website
https://www.curiousinfotechsolutions.com

Java Programming

Day-4

Read Input Value in Java	https://youtu.be/VdIa-m7H-uU
Branching Statements in Java	https://youtu.be/AZZkRlUeiR4
Looping Statements in Java	https://youtu.be/nHlsf8U8nMQ

Email id:
info@curiousinfotechsolutions.com

Syllabus Day-4

Part-1

- Input in Java

Part-2

- Java Control Statements
  - Branching Statements
  - Looping Statements
  - Jumping Statements

Inputs integer number in Java


```
import java.util.Scanner;
public class Main
{
 public static void main(String[] args) {
        Scanner input= new Scanner(System.in);
        System.out.print("Enter Value of A:");
        int A = input.nextInt();//2,-10,0
        System.out.print("A="+A);//A=2
         }
}
```

Inputs float number in Java


```
import java.util.Scanner;
public class Main
{
 public static void main(String[] args)
    {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter a float: ");
        float number = input.nextFloat();//3.14, -3.14, 0.0
        System.out.println("You entered " + number);You entered 3.14
        // closing the scanner object
        input.close();
    }
}
```

Inputs string in Java


```
import java.util.Scanner;
public class Main
{
 public static void main(String[] args)
    {
       Scanner input = new Scanner(System.in);
       System.out.print("Enter Name: ");
       String myString = input.next();//Vivek, V12345,
       System.out.println("Name entered = " + myString);//Name enterd Vivek
       // closing the scanner object
       input.close();
    }
}
```

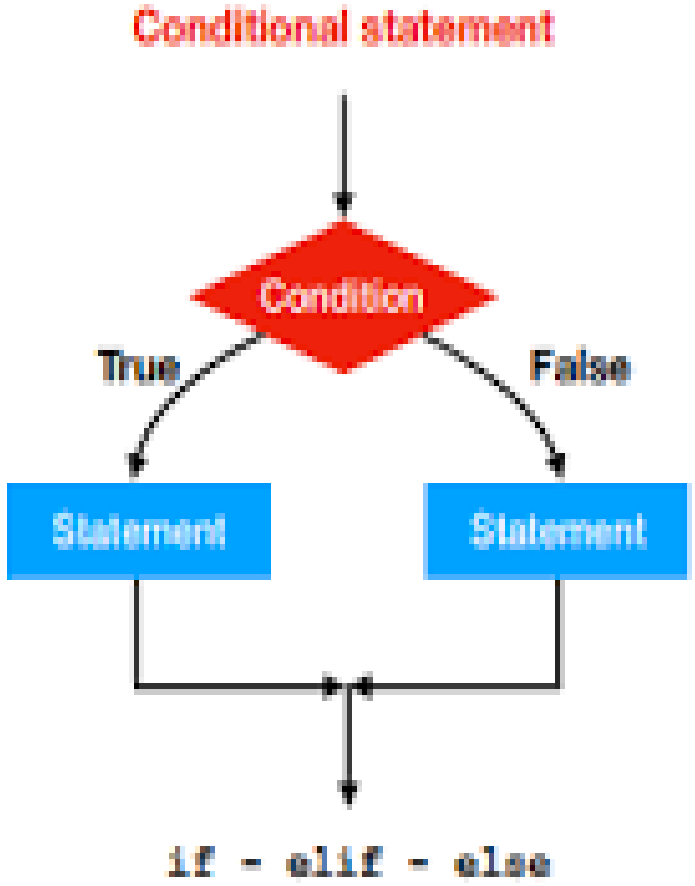
Java Control Statements

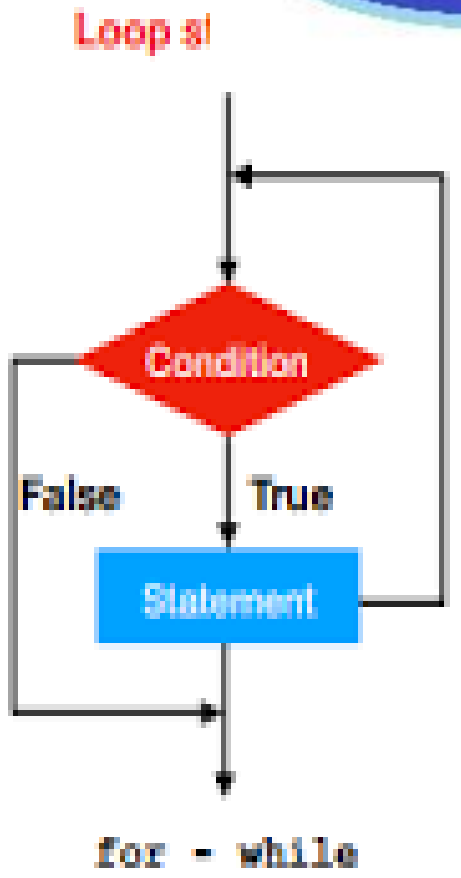
1. Decision Making statements
    1. if statements
    2. If-else statements
    3. switch statement
2. Loop statements
    1. do while loop
    2. while loop
    3. for loop
3. Jump statements
    1. break statement
    2. continue statement

Nested if
Nested switch
Nested Loops

https://www.curiousinfotechsolutions.com

Types of Control Structures



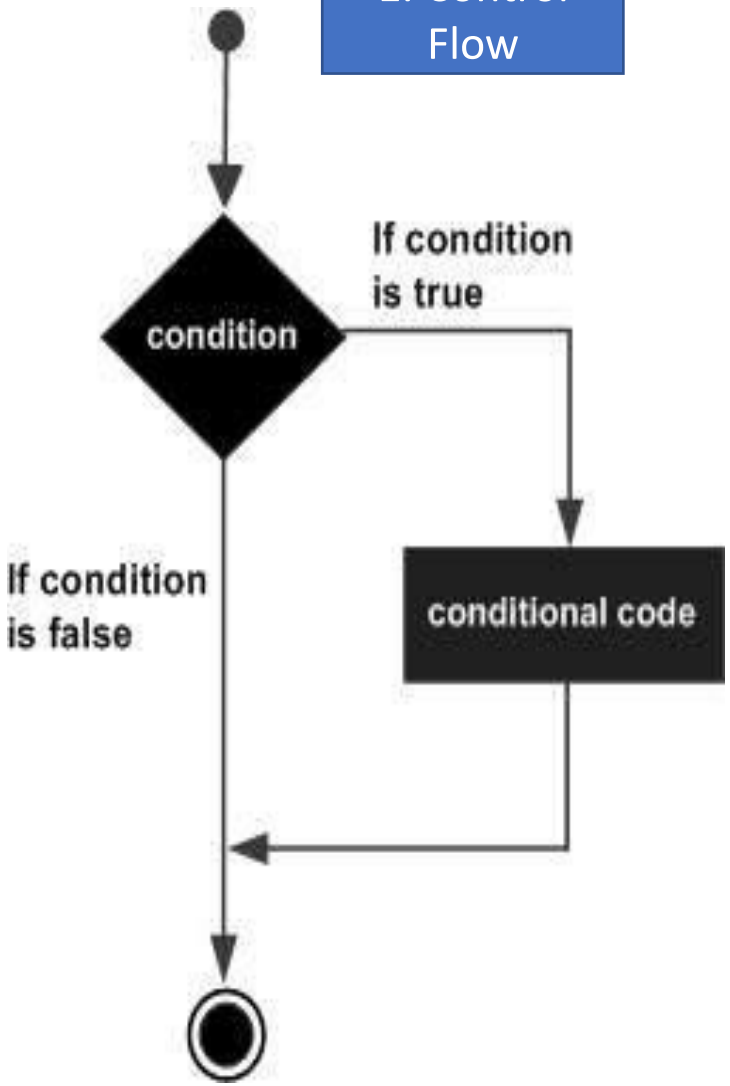


Decision-Making if statements

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

Simple if statement

1. Control Flow



2. Syntax

Syntax of if statement

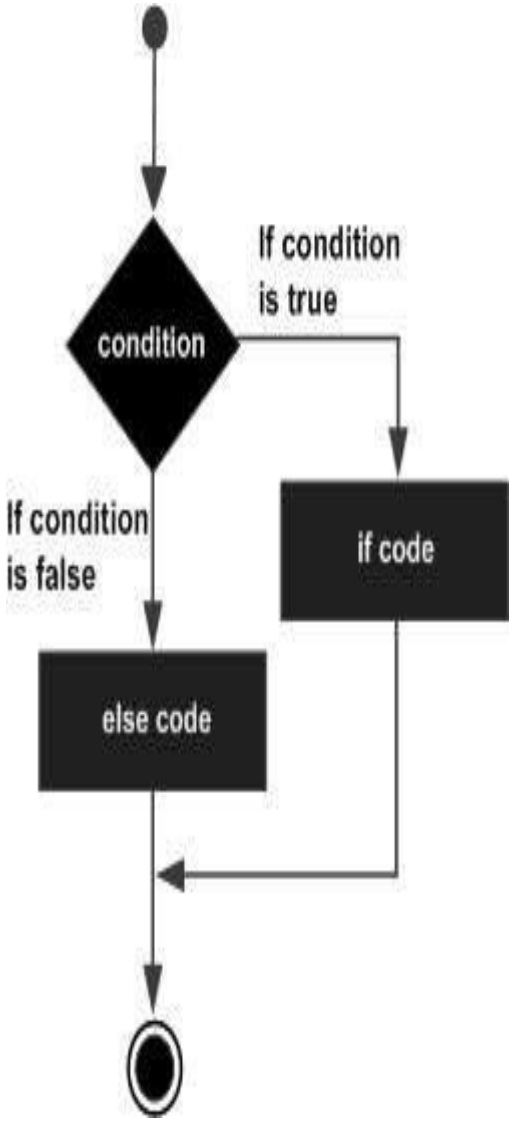
```
1. if (condition)
2. {
3.         statement 1; //executes when condition is true
   }
```


3. Example

Student.java

```
1. public class Student
2. {
    1. public static void main(String[] args)
    2. {
    3. Scanner sc= new Scanner(System.in);
    4. System.out.print("Enter N: ");
    5. int x= sc.nextInt();//x=20
    6. int y= sc.nextInt();//y=3
    7. If (x+y > 20)
        1. System.out.println("x + y is greater than 20")
    8. }
```

if-else statement



Syntax:

1.if(condition)

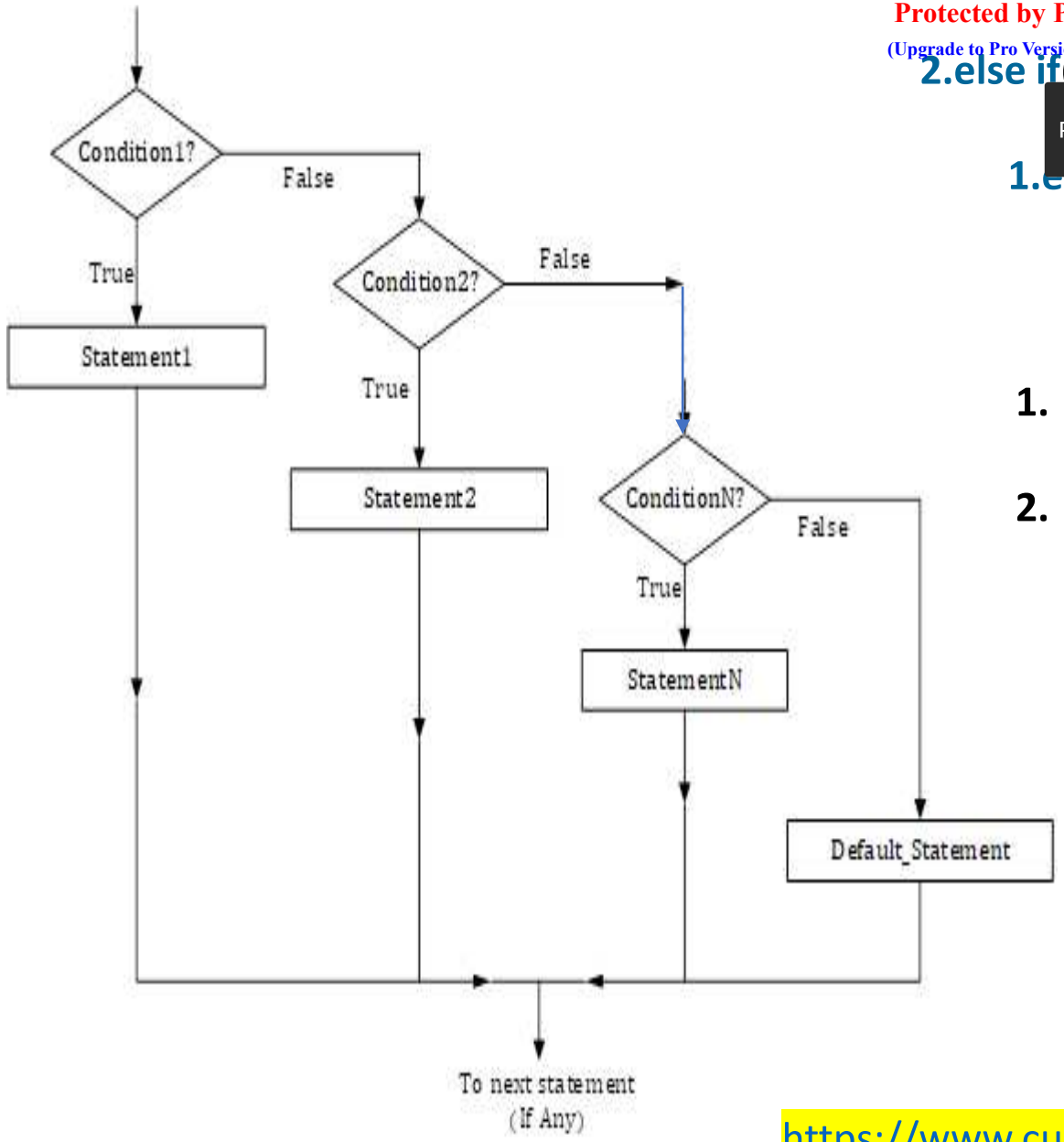
1.statement 1; //executes when condition is true

2.else

1.statement 2; //executes when condition is false

```
1. public class Student
2. {
    1. public static void main(String[] args)
    2. {
    3.  Scanner sc= new Scanner(System.in);
    4.  System.out.print("Enter N: ");
    5.  int x= sc.nextInt();//x=2=3
    6.  if( x%2 == 0)5%2 -> 1 == 0
        1. System.out.println("x is Even Number"); //2,4,6
    7.  else
        1. System.out.println("x is odd number"); //1,3,5
    8. }
3. }
```

if-else-if ladder:



1.if(condition 1)

1.Tstatement 1; //executes when condition 1 is true

2.else if(condition 2)

1.Tstatement 2; //executes when condition 2 is true

1.else

1.Fstatement ; //executes when all the conditions are false

1. if (N < 0) -10 < 0 T
    1. System.out.println("N is Lesser than Zero");
2. else if(N == 0)
    1. System.out.println("N is Zero");
1. else
    1. System.out.println("N is Greater than Zero");

CURIOUS INFOTECH SOLUTIONS LLP

https://www.curiousinfotechsolutions.com

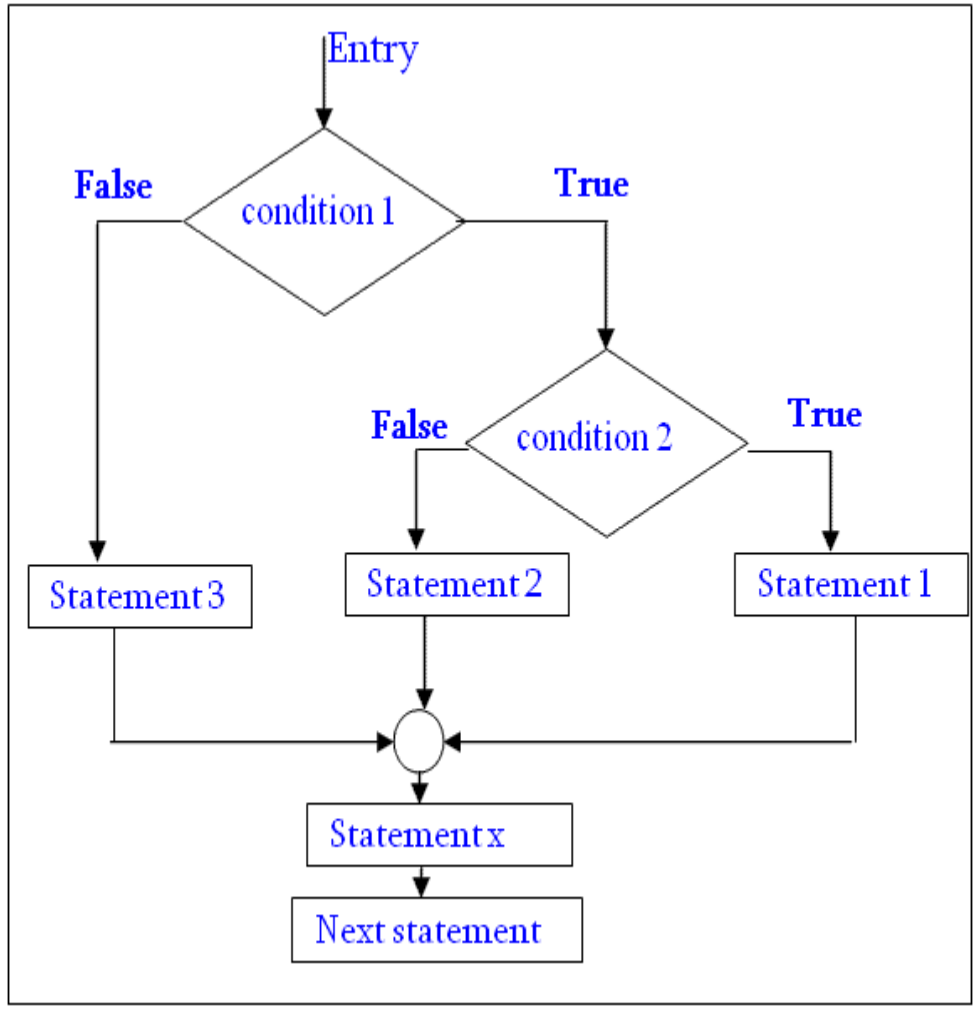
if-else-if ladder Example

```
Output

Enter N : 0
N is Zero

Enter N : 10
N is Greater than Zero

Enter N : -10
N is Lesser than Zero
```



```
import java.util.*;
public class Main
{
    public static void main(String[]args)
    {
        Scanner input= new Scanner(System.in);
        System.out.print("Enter N: ");
        int N = input.nextInt();
        if (N < 0)
            System.out.println("N is Lesser than Zero");
        else if(N == 0)
              System.out.println("N is Zero");
            else
             System.out.println("N is Greater than Zero");
    }
}
```

Nested if-statement



1.if(condition 1)
    1.if(condition 2)
        1.statement 1; //executes when condition 2 is true
    2.else
        1.statement 2; //executes when condition 2 is false
2.else
    1.Statement3; //executes when condition 1 is false

0, 10, -10

```
if (N >= 0)
    if(N == 0)
       System.out.println("N is Zero");
    else
       System.out.println("N is Greater than Zero");
 else
    System.out.println("N is Lesser than Zero");
```

Nested if-statement Example

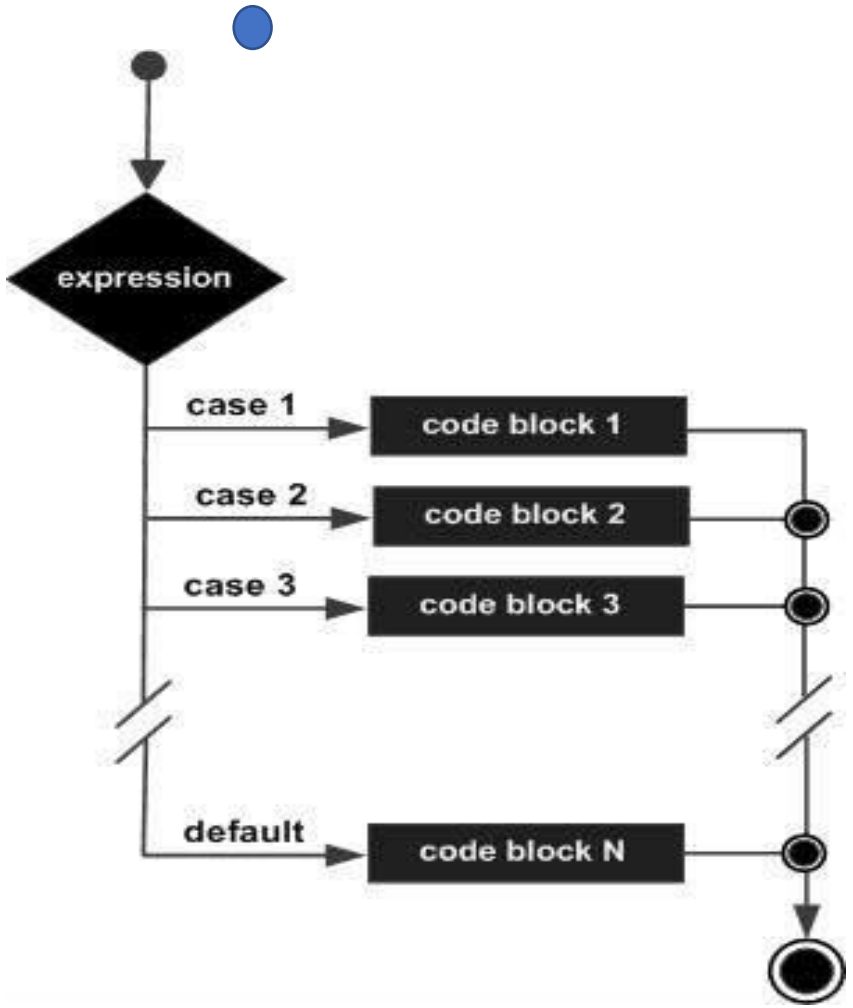
```
Output

Enter N : 0
N is Zero

Enter N : 10
N is Greater than Zero

Enter N : -10
N is Lesser than Zero
```



```
import java.util.Scanner;
public class Main
{
    public static void main(String[]args)
    {
      Scanner sc= new Scanner(System.in);
       System.out.print("Enter N: ");
       int N = sc.nextInt();
       if (N >= 0)
          if(N == 0)
             System.out.println("N is Zero");
          else
                    System.out.println("N is Greater than Zero");
       else
          System.out.println("N is Lesser than Zero");
    }
}
```

Switch Statement


```
1.switch (expression)
2.{
3.    case value1:
4.     statement1;
5.     break;
6.    .
7.    .
8.    .
9.    case valueN:
10.     statementN;
11.     break;
12..
13.    default:
14.     default statement;
15.}
```



switch Example

Output

Enter N:0
Zero
End of Program

Enter N:9
Nine
End of Program

Enter N:10
Invalid Input
End of Program


```
import java.util.Scanner;
public class Main
{
    public static void main(String[] args) {
        Scanner Input = new Scanner(System.in);
        System.out.print("Enter N:");
        int N = Input.nextInt();
        switch(N) N=0
        {
            case 0: System.out.print("Zero");break;
            case 1: System.out.print("One");break;
            case 2: System.out.print("Two");break;
            case 3: System.out.print("Three");break;
            case 4: System.out.print("Four");break;
            case 5: System.out.print("Five");break;
            case 6: System.out.print("Six");break;
            case 7: System.out.print("Seven");break;
            case 8: System.out.print("Eight");break;
            case 9: System.out.print("Nine");break;
            default: System.out.print("Invalid Input");
        }
        System.out.print("\nEnd of Program");
    }
}
```

Need of Loop


```
WAP to display Hello World 5000 times

public class Main
{
 public static void main(String[] args)
 {
  System.out.println("Hello World");
  System.out.println("Hello World");
  System.out.println("Hello World");
  System.out.println("Hello World");
  System.out.println("Hello World");
 }
}
```

```
WAP to read and display 300 Numbers.
import java.util.Scanner;
public class Main
{
  public static void main(String[] args) {
      Scanner input = new Scanner(System.in);
      System.out.print("Enter Value of A:");
      int A = input.nextInt();
      System.out.println("A="+A);
      System.out.print("Enter Value of B:");
      int B = input.nextInt();
      System.out.println("B="+B);
      System.out.print("Enter Value of C:");
      int C = input.nextInt();
      System.out.println("C="+C);
      // closing the scanner object
      input.close();
   }
}
```

Concept of Looping

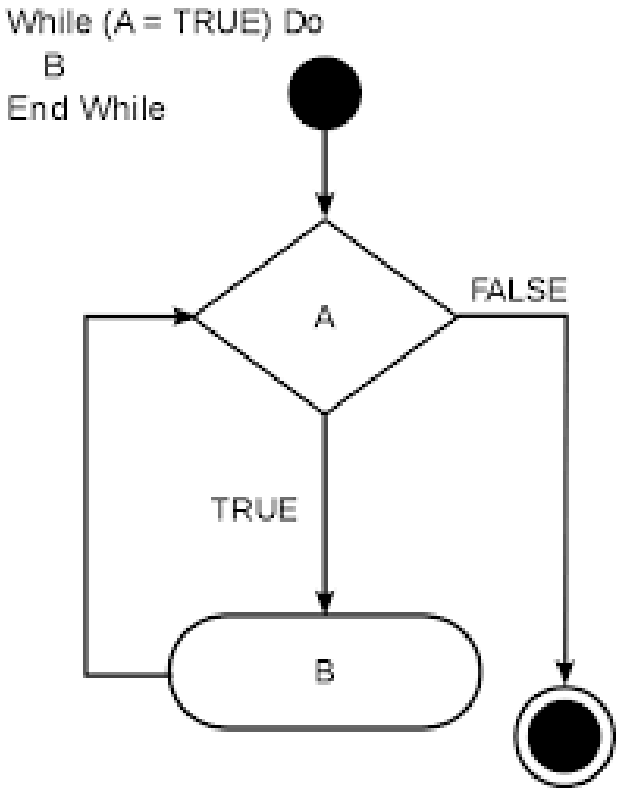
- A loop in a computer program is **an instruction that repeats until a specified condition is reached.**
- There are 3 type of loops in Java language
  1.while loop
  2.for loop
  3.do-while loop

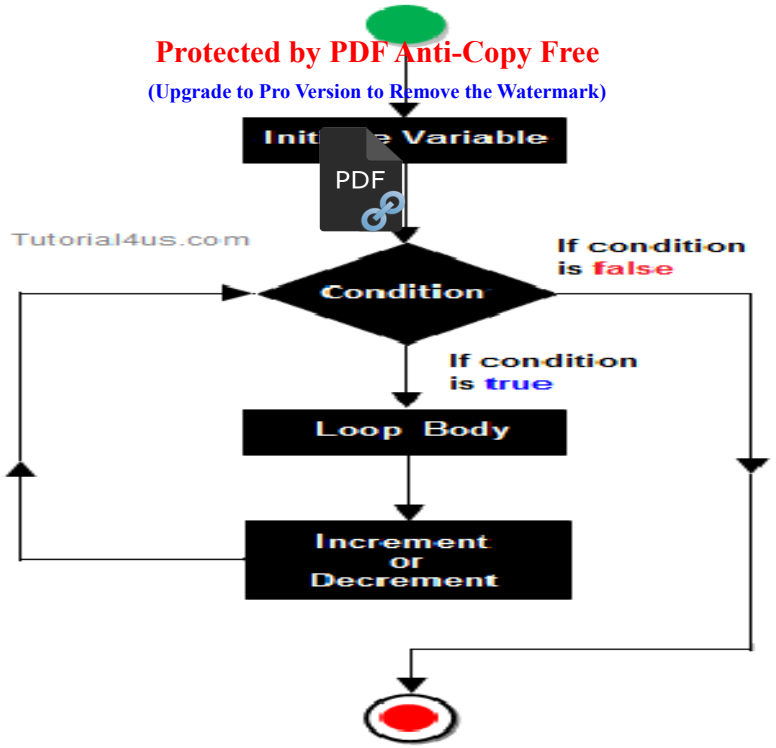
1.Entry Controlled Loops: In this type of loops the test condition is tested before entering the loop body. **For Loop** and **While Loop** are entry controlled loops.

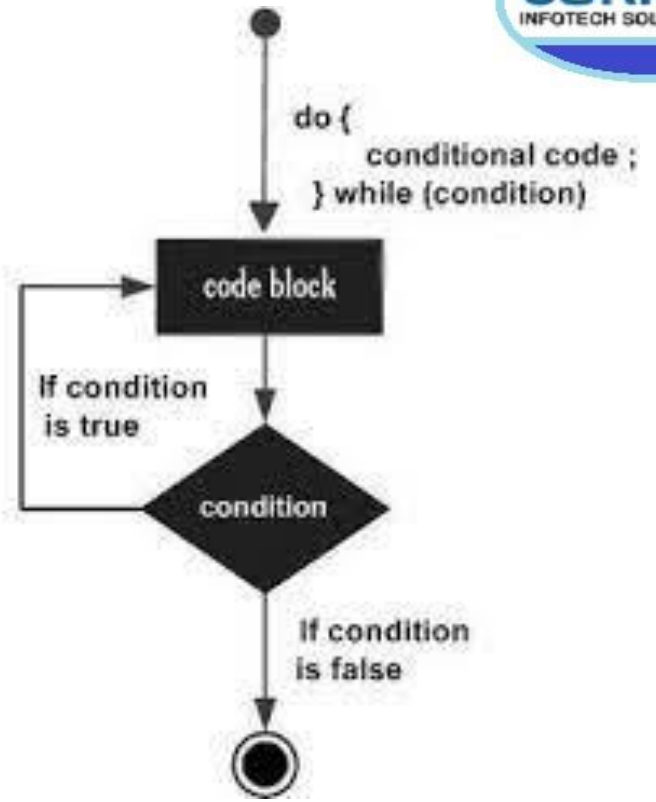
2.Exit Controlled Loops: In this type of loops the test condition is tested or evaluated at the end of loop body. **do-while Loop** is exit controlled loops

- Let to Display Natural Number Series
  - 1,2,3,4,5,6,7,8,9,10

- Observe
  - First Number – 1
  - Last Number – 10
  - Difference between 1st No and 2nd No – 1



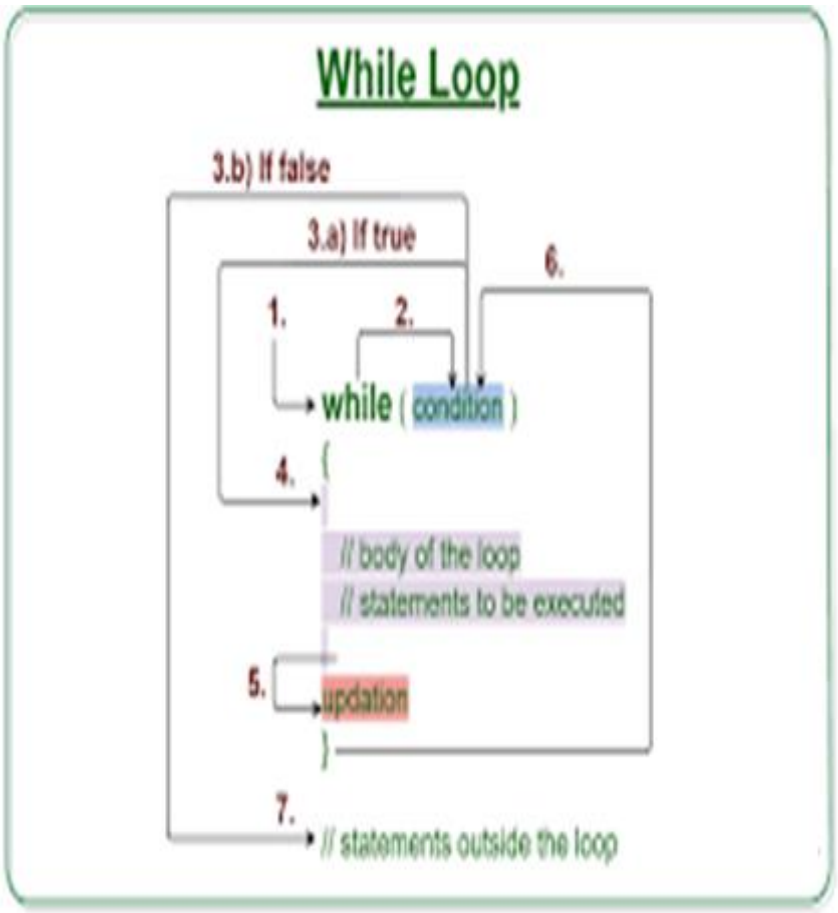


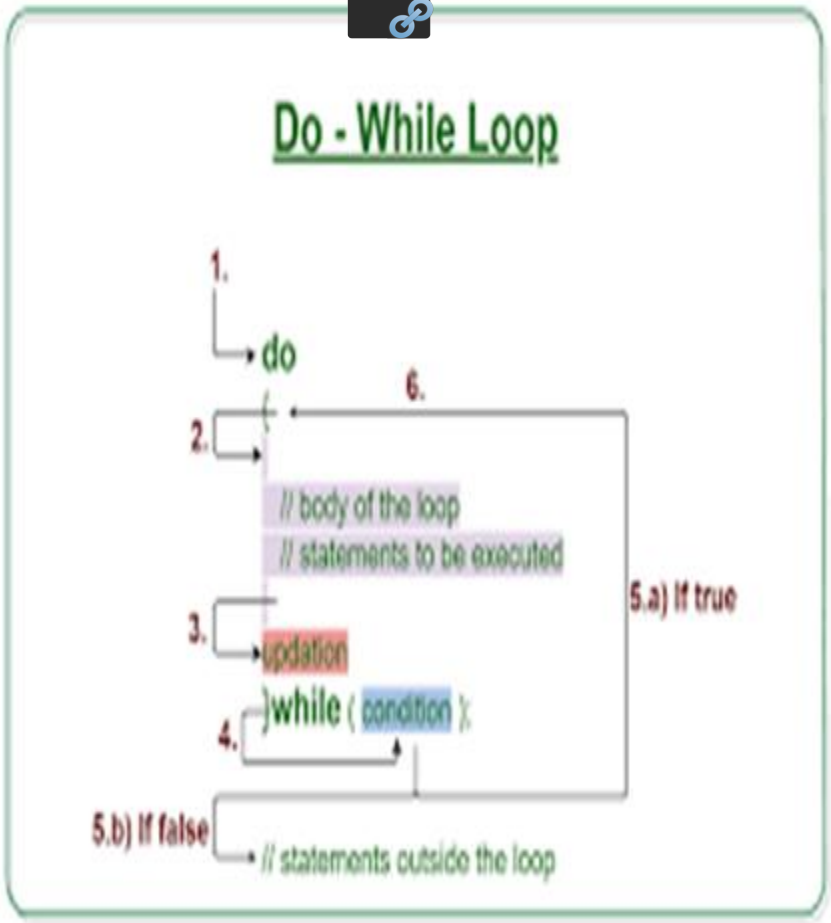


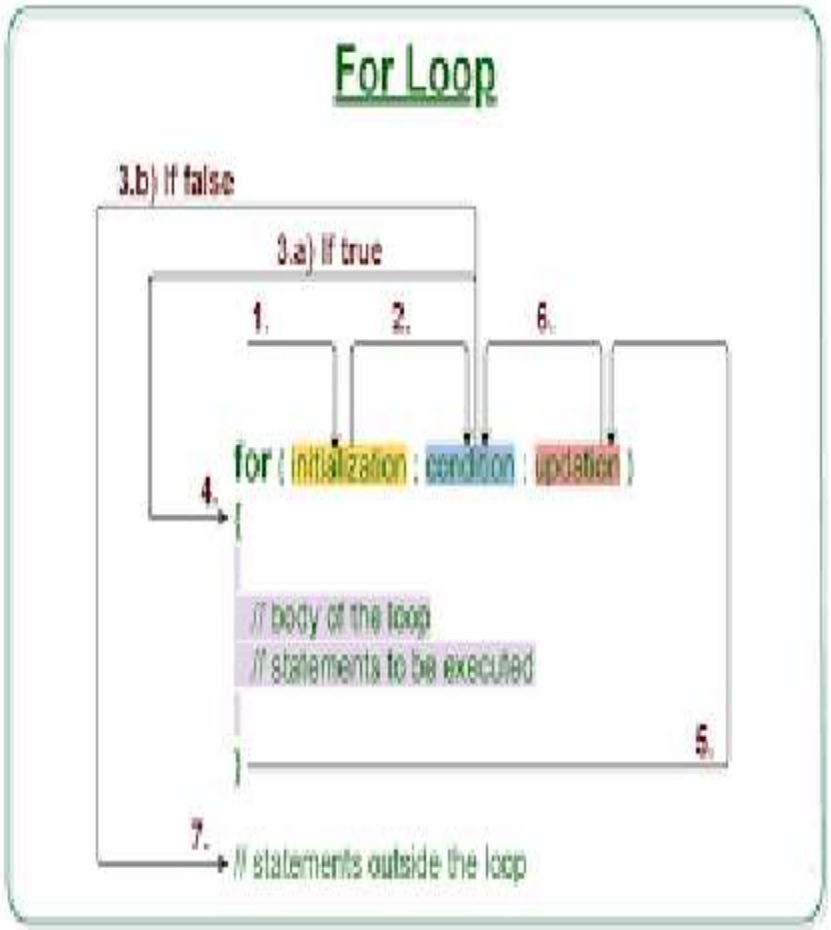
Flow Control of Looping Statements

https://www.curiousinfotechsolutions.com

Syntax of Looping Statements







Example

WAP to display number from 1 to 10
1,2,3,4,5,6,7,8,9,10

```
public class Main
{
    public static void main(String[] args) {
    int FN=1, LN=10;
    while (FN <= LN)
        {
            System.out.println(FN);
            FN = FN + 1;
        }
    }
}
```

```
public class Main
{
    public static void main(String[] args) {
    int FN=1, LN=10;
    do
        {
            System.out.println(FN);
            FN = FN + 1;
        }while (FN <= LN);
    }
}
```

```
public class Main
{
    public static void main(String[] args) {
    int FN, LN=10;
        for(FN=1;FN<=LN;FN=FN+1)
        {
            System.out.println(FN);
        }

    }
}
```

Exp-1 Using while loop display sum of series 1,2,3,4,5,6,7,8,9,10 = 55

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

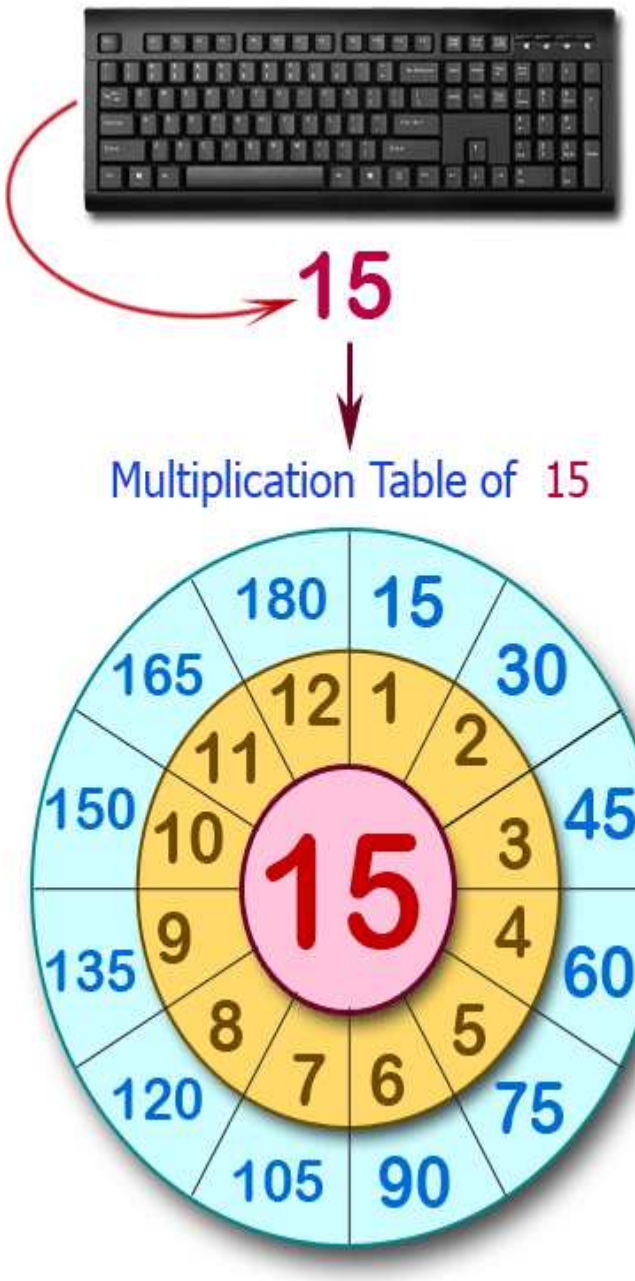
```
public class Main
{
    public static void main(String[] args) {
        int FN, LN=1, Sum = 0;
        for(FN=10;FN>=LN;FN=FN-1)
            {
                    Sum = Sum + FN;
                    System.out.println(FN);
            }
        System.out.println("Sum="+Sum);
    }
}
```

Exp-2 WAJP to sum of positive numbers if the user enters a negative number, the loop ends the negative number entered is not added to the sum. For Example: 1,2,3,-4 = 6


```
import java.util.Scanner;
public class Main
{
    public static void main(String[] args) {
    Scanner Input = new Scanner(System.in);
    int sum = 0;

    // take input from the user
    System.out.print("Enter a number: ");
    int number = Input.nextInt();//1
while (number >= 0) -4>=0 f
{
        // add all positive numbers
        sum += number; 3+3=6
        // take input again if the number is positive
        System.out.print("Enter a number: ");
        number = Input.nextInt();//-4
    }
    // display the sum
    System.out.print("\nThe sum is "+sum);//6
    }
}
```

Exp-3 WAP to read a = 1234 and display sum of digits i.e. 1+2+3+4 = 10


```
public class Main
{
   public static void main(String[] args) {
   int n, Sum = 0, m;
   n = 1234;
   while( n > 0 )   123>0 T
      {
         m = n % 10; 3 = 123%10
         Sum = Sum + m; 7 = 4 + 3
         n = n / 10;  12=123/10
      }
      System.out.print("Sum is = "+ Sum);
   }
}
```




```
public class Main
{
   public static void main(String[] args) {
   int j,n=15;

   System.out.println("Multiplication Table of " + n + " is:");
   for(j=1;j<=12;j++)
      {
         System.out.println( n*j );//15*12=180
      }
   }
}
```

```
15
30
45
```

Break and Continue

Break:

We have already seen the `break` statement used in an earlier Class in switch statement.

It was used to "jump out" of a `switch` statement.

The `break` statement can also be used to jump out of a **loop**.

Continue Example

The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

```
public class Main
{
   public static void main(String[] args) {
   for (int i = 1; i <= 10; i++)
      {
            if (i == 4) break;
               System.out.println( i );
      }
   }
}
```

Output
1
2
3

Continue Example


```
public class Main
{
    public static void main(String[] args) {
    for (int i = 1; i <= 10; i++)
        {
            if (i == 4) continue;
                System.out.println( i );
        }
    }
}
```

Output

1
2
3
5
6
7
8
9
10

WAP to display a Pattern


```
public class Main
{
    public static void main(String[] args) {
    for (int i = 1; i <= 5; i++)
     {
        System.out.println("*****");
      }
    }
}
```

Output

```
R1*****
R2*****
R3*****
R4*****
R5*****
```

Exercise

- WAP to display following pattern

```
R1* * * * *
R2  * *  * *
R3    * * *
R4      * *
R5        *
```

```
R1* * *  * *
R2* * * *
R3* * *
R4* *
R5*
```

```
Input : 5
Output :
        *
      * *
    * * *
  * * * *
* * * * *
* * * * *
  * * * *
    * * *
      * *
        *
```

Nested Loops in Java

Syntax for Nested For loop:

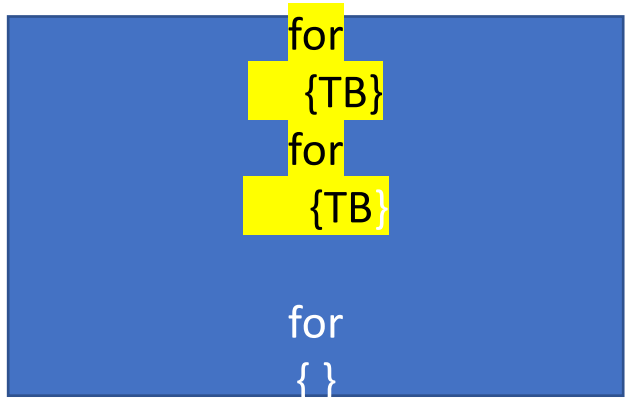
```
for3 ( initialization; condition; increment )
{
  for3 ( initialization; condition; increment )
    {
      // statement of inside loop 3*3 =9
    }
  // statement of outer loop
}
```

Syntax for Nested While loop:

```
while(condition)
{
  while(condition)
  {
    // statement of inside loop
   }
  // statement of outer loop
}
```

Syntax for Nested Do-While loop:

```
do
{
  do
    {
     // statement of inside loop
     } while(condition);
  // statement of outer loop
}while(condition);
```



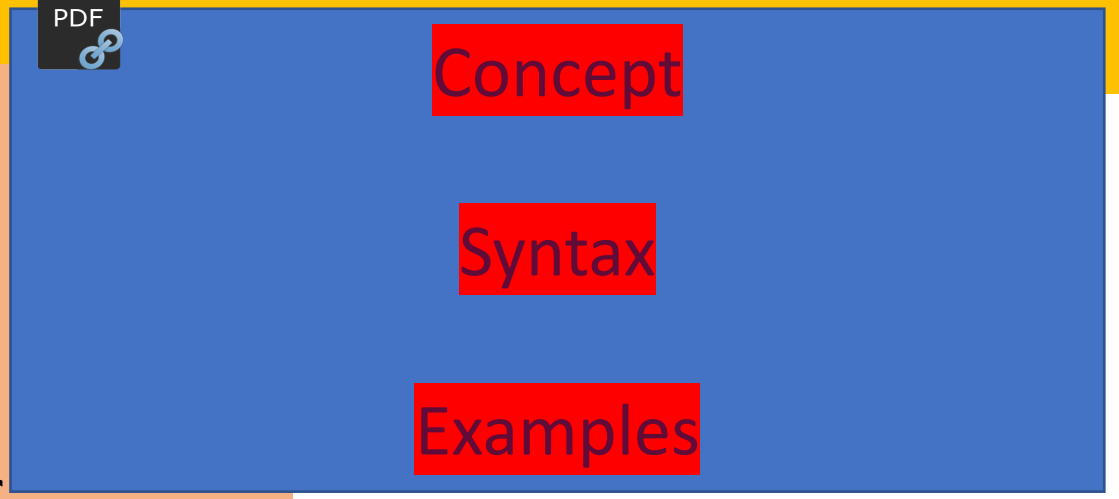
WAP to display following pattern.


```
public class Main
{
    public static void main(String[] args) {
    int i,j,n=15;

    for(i=1;i<=10;i++)
        {
         for (j=1;j<i;j++)
            System.out.print(n*j + " ");
        System.out.println("");
        }
    }
}
```

Output

```
15
15 30
15 30 45
15 30 45 60
15 30 45 60 75
15 30 45 60 75 90
15 30 45 60 75 90 105
15 30 45 60 75 90 105 120
15 30 45 60 75 90 105 120 135
```

Methods in Java



- Create a Method
  - Predefined Method
  - User-defined Method
- **Access Specifier**
- Java Method Parameters
- Multiple Parameters
- Return Values
- A Method with If...Else
- Java Method Overloading

CURIOUS INFOTECH SOLUTIONS LLP

https://www.curiousinfotechsolutions.com

Java Methods

- A **method** is a block of code which only runs when it is called.
- We can pass data, known as parameters, into a method.
- Methods are used to perform certain actions, and they are also known as **functions**.
- Why use methods?
  - ***To reuse code****: define the code once, and use it many times.*

Create a Method

- A method must be declared within a class.
- It is defined with the name of the method, followed by parentheses ().
- Types of Method:
  - There are two types of methods in Java:
    - Predefined Method
    - User-defined Method

Predefined Method

- In Java, predefined methods are the method that is already defined in the Java class libraries is known as predefined methods.
- It is also known as the standard library method or built-in method. We can directly use these methods just by calling them in the program at any point.
- Some pre-defined methods are length(), equals(), compareTo(), sqrt(), etc.
- When we call any of the predefined methods in our program, a series of codes related to the corresponding method runs in the background that is already stored in the library.

Example of Predefined Method

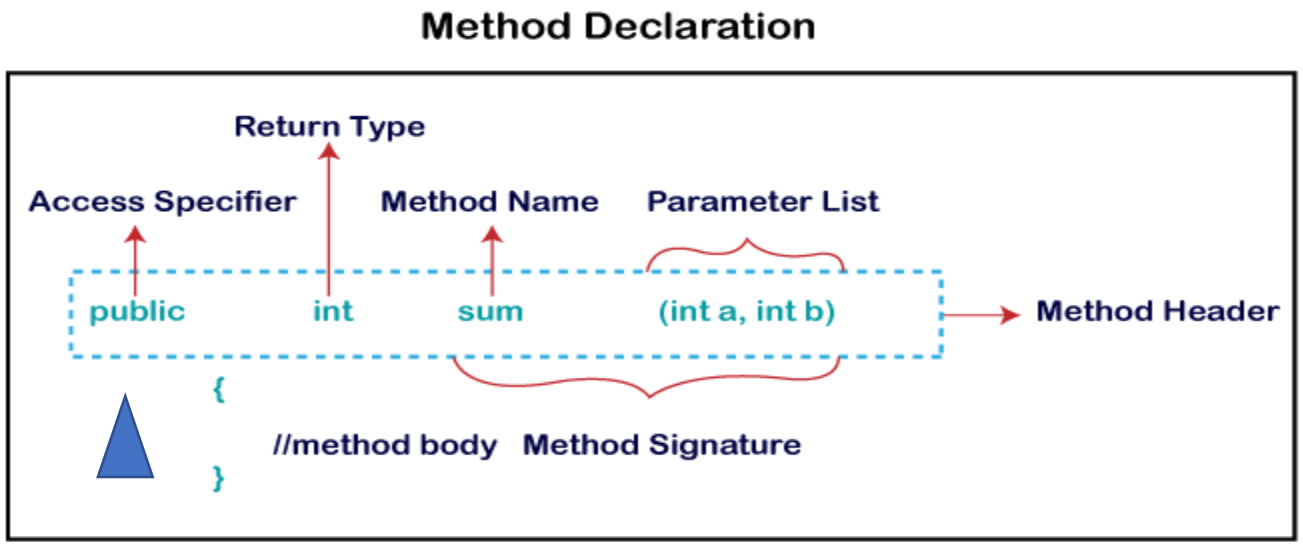

```
public class Main
{
    public static void main(String[] args)
    {
        // using the max() method of Math class
        System.out.print("The maximum number is: " + Math.max(Math.max(9,7),5));
    }
}
```

method Math.max(int,int)

User-defined Method

- The method written by the user or programmer is known as **a user-defined** method. These methods are modified according to the requirement.
- **Method Signature:** Every method has a method signature. It is a part of the method declaration. It includes the **method name** and **parameter list**.

```
C=sum(10,20);
```



Access Specifier

- Access specifier or modifier is the access type of the method. It specifies the visibility of the method. Java provides **four** types of access specifier:
- **Public:** The method is accessible by all classes when we use public specifier in our application.
- **Private:** When we use a private access specifier, the method is accessible only in the classes in which it is defined.
- **Protected:** When we use protected access specifier, the method is accessible within the same package or subclasses in a different package.
- **Default:** When we do not use any access specifier in the method declaration, Java uses default access specifier by default. It is visible only from the same package only.

Contd..

- **Return Type:** Return type is a data type that the method returns. It may have a primitive data type, object, collection, void, etc. If the method does not return anything, we use void keyword.
- **Method Name:** It is a unique name that is used to define the name of a method. It must be corresponding to the functionality of the method. Suppose, if we are creating a method for subtraction of two numbers, the method name must be **subtraction().** A method is invoked by its name.
- **Parameter List:** It is the list of parameters separated by a comma and enclosed in the pair of parentheses. It contains the data type and variable name. If the method has no parameter, left the parentheses blank.
- **Method Body:** It is a part of the method declaration. It contains all the actions to be performed. It is enclosed within the pair of curly braces.

Contd..

- Naming a Method
  - While defining a method, remember that the method name must be a **verb** and start with a **lowercase** letter. If the method name has more than two words, the first name must be a verb followed by adjective or noun. In the multi-word method name, the first letter of each word must be in **uppercase** except the first word. For example:
- **Single-word method name:** sum(), area()
- **Multi-word method name:** areaOfCircle(), stringComparision()

Exp.java
javac Exp.java Exp.class
java Exp

Exp.cpp
Exp.exe

Create a method inside Main


```
public class Main {
  static void myMethod() {
    System.out.println("This is my Method!");
  }

  public static void main(String[] args) {
    myMethod();
  }
}
```

User Defined Method

Calling Method Statement

myMethod() is the name of the method
static means that the method belongs to the Main class
void means that this method does not have a return value

https://www.curiousinfotechsolutions.com

A method can also be called multiple times

```
public class Main {
  static void myMethod() {
    System.out.println("This is my Method");
  }

  public static void main(String[] args) {
    myMethod();
    myMethod();
    myMethod();
  }
}
```

Output

This is my Method
This is my Method
This is my Method

https://www.curiousinfotechsolutions.com

Java Method Parameters

- Information can be passed to methods as parameter. Parameters act as variables inside the method.

- Parameters are specified after the method name, inside the parentheses. We can add as many parameters as we want, just separate them with a comma.

- The following example has a method that takes a String called fname as parameter. When the method is called, we pass along a first name, which is used inside the method to print the full name.

Example:


```
public class Main {
  static void myMethod(String fname) {
    System.out.println(fname + " Sharma");
  }

  public static void main(String[] args) {
    myMethod("Rama");
    myMethod("Raj");
    myMethod("Rajesh");
  }
}
```


When a parameter is passed to the method, it is called an argument. So, from the example above: fname is a parameter, while Rama, Raj and Rajesh are arguments.

Output
Rama Sharma
Raj Sharma
Rajesh Sharma

Multiple Parameters


```
public class Main {
  static void myMethod(String fname, int age) {
    System.out.println(fname + " is " + age + " Yr Old. " ) ;
  }

  public static void main(String[] args) {
    myMethod("Rama", 5);
    myMethod("Ram", 8);
    myMethod("Rajesh", 7);
  }
}
```

Output

Rama is 5 Yr Old.
Ram is 8 Yr Old.
Rajesh is 7 Yr Old.

Return Values


```
public class Main {
  static int Add(int x, int y) {
    return x + y;
  }

  public static void main(String[] args) {
    System.out.println(Add(5,3));
  }
}
```

Output
Outputs 8 (5 + 3)

A Method with If...Else


```
public class Main {
    //user defined method
        public static void findEvenOdd(int num)
        {
        //method body
        if(num%2==0)
        System.out.println(num+" is even");
        else
        System.out.println(num+" is odd");
        }
     public static void main(String[] args) {
      int X=20;
       findEvenOdd(X);
     }
    }
```

Output
20 is Even

Java Method Overloading

With **method overloading**, multiple methods can have the same name with different parameters

Example

- int Add(int x, int y, int z)
- int Add(int x, int y)
- double Add(double x, double y)

```
public class Main {
  static int Add(int x, int y) {
    return x + y;
  }

  static double Add(int x, double y) {
    return x + y;
  }

  public static void main(String[] args) {
    int C = Add(3,5);
    double D = Add(C, 10.0);
    System.out.println("int: " + C);
    System.out.println("double: " + D);
  }
}
```

Output
int: 8
double: 18.0

Java Recursion

- Recursion is the technique of making a function call itself.
- This technique provides a way to break complicated problems down into simple problems which are easier to solve.
- When the sum() function is called, it adds parameter k to the sum of all numbers smaller than k and returns the result. When k becomes 0, the function just returns 0. When running, the program follows these steps:

  - 10 + sum(9)
  - 10 + (9 + sum(8))
  - 10 + (9 + (8 + sum(7)))
  - ...
  - 10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 + sum(0)
  - 10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 + 0

Iteration for
Sum = Sum + N
induction

Example Java Recursion


```
public class Main {
  public static void main(String[] args) {
    int result = sum(10);
    System.out.println(result);
  }

 public static int sum(int k) {
    if (k > 0) { 0>0 F
      return k + sum(k - 1);
    } else { //False
      return 0;
    }
  }
}
```

	Return 1 + Sum(0)	Return 1 + 0
	Return 2 + Sum(1)	Return 2 + 1
	Return 3 + Sum(2)	Return 3 + 2 + 1 + 0
	Return 4+Sum(3)	Return 4 + 3 + 2 + 1 + 0
15	Retun 5+Sum(4)	Return 5+4+3+2+1
Result=sum(5)	Sum(5)	15
main()		

Example Static Method


```
public class Main
{
    public static void main(String[] args)
    {
        show();
    }
    static void show()
    {
        System.out.println("It is an example of static method.");
    }
}
```