CORE JAVA

Uday Sharma

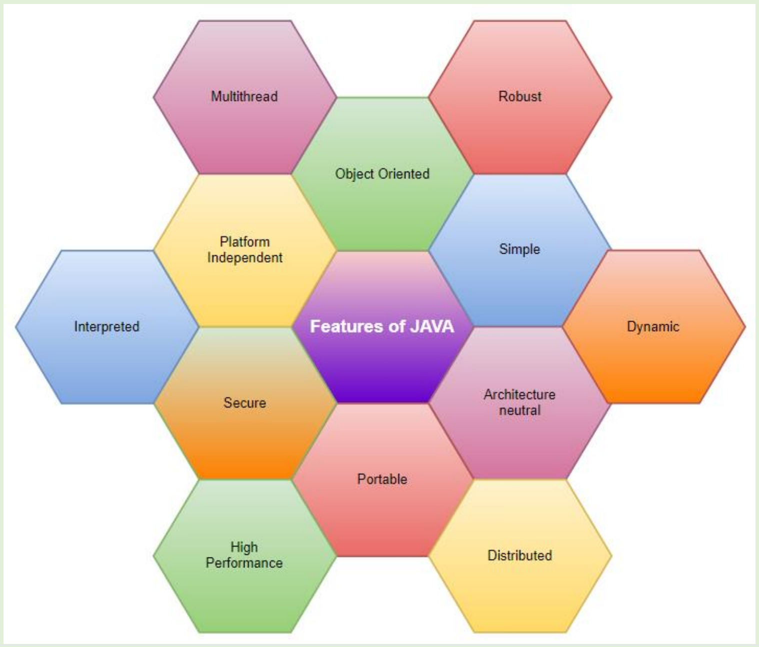
mrudaysharma4600@gmail.com

CORE JAVA :-

JAVA:-

Java is an open-source, class-based, high-level, object-oriented programming language. Java is platform independent as the java programs are compiled into byte code that are platform independent.

- ***Features of Java:-***
- Object Oriented: In Object Oriented programming everything is an object rather that function and logic.
- Simple: Java is simple to understand, easy to learn and implement.
- Secured: It is possible to design secured software systems using Java.
- Platform Independent: Java is written once and run anywhere language, meaning once the code is written, it can be executed on any software and hardware systems.
- Portable: Java is not necessarily fixated to a single hardware machine. Once created, java code can be used on any platform.
- Architecture Neutral: Java is architecture neutral meaning the size of primitive type is fixed and does not vary depending upon the type of architecture.
- Robust: Java emphasizes a lot on error handling, type checking, memory management, etc. This makes it a robust language.
- Interpreted: Java converts high-level program statement into Assembly Level Language, thus making it interpreted.
- Distributed: Java lets us create distributed applications that can run on multiple computers simultaneously.
- Dynamic: Java is designed to adapt to ever evolving systems thus making it dynamic.
- Multi-thread: multi-threading is an important feature provided by java for creating web applications.
- High-performance: Java uses Just-In-Time compiler thus giving us a high performance.



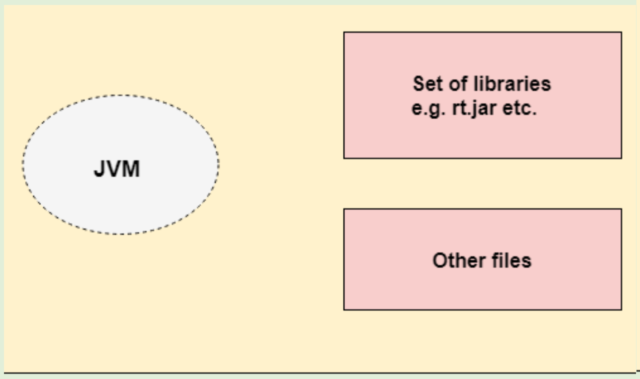
- ***JVM:-***

JVM (Java Virtual Machine) is an abstract machine. It is called a virtual machine because it doesn't physically exist. It is a specification that provides a runtime environment in which Java bytecode can be executed. It can also run those programs which are written in other languages and compiled to Java bytecode.
JVMs are available for many hardware and software platforms. JVM, JRE, and JDK are platform dependent because the configuration of each OS is different from each other. However, Java is platform independent. There are three notions of the JVM: *specification*, *implementation*, and *instance*.
The JVM performs the following main tasks:

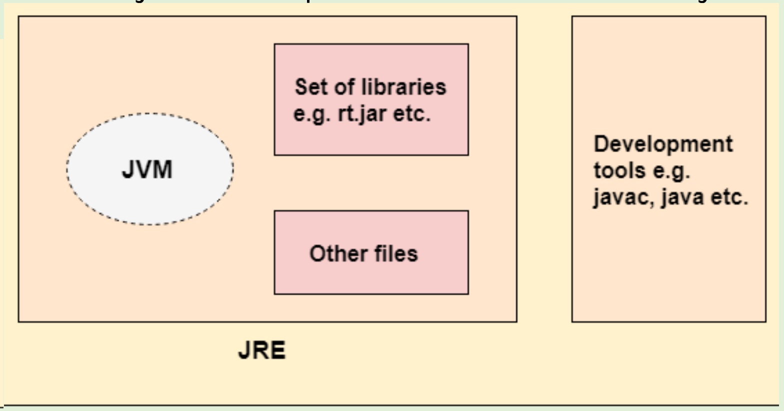
- Loads code
- Verifies code
- Executes code
- Provides runtime environment

- ***JRE:-***

JRE is an acronym for Java Runtime Environment. It is also written as Java RTE. The Java Runtime Environment is a set of software tools which are used for developing Java applications. It is used to provide the runtime environment. It is the implementation of JVM. It physically exists. It contains a set of libraries + other files that JVM uses at runtime.
The implementation of JVM is also actively released by other companies besides Sun Micro Systems.



JRE


JVM
Set of libraries e.g. rt.jar etc.
Other files
Development tools e.g. javac, java etc.
JRE

JDK

- ***JDK:-***

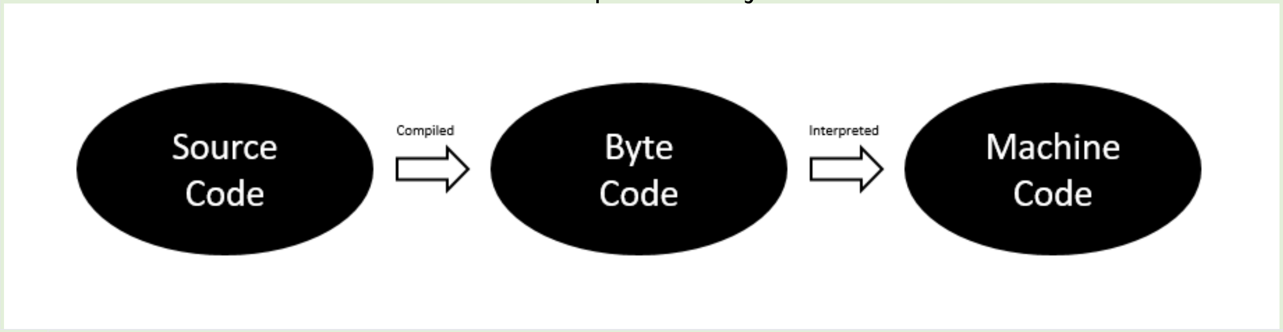
JDK is an acronym for Java Development Kit. The Java Development Kit (JDK) is a software development environment which is used to develop Java applications and applets. It physically exists. It contains JRE + development tools.
JDK is an implementation of any one of the below given Java Platforms released by Oracle Corporation:

- Standard Edition Java Platform
- Enterprise Edition Java Platform
- Micro Edition Java Platform

The JDK contains a private Java Virtual Machine (JVM) and a few other resources such as an interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc), etc. to complete the development of a Java Application.

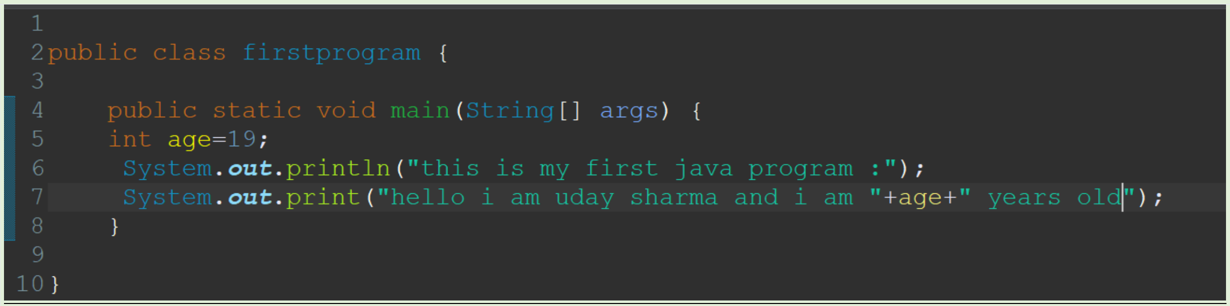
- ***Basic Structure of a Java Program:-***
- Java is one of the most popular programming languages because it is used in various tech fields like app development, web development, client-server applications, etc.
- Java is an object-oriented programming language developed by Sun Microsystems of the USA in 1991.
- It was originally called Oak by James Goslin. He was one of the inventors of Java.
- Java = Purely Object-Oriented.

How Java Works?
The source code in Java is first compiled into the bytecode.
Then the Java Virtual Machine(JVM) compiles the bytecode to the machine code.

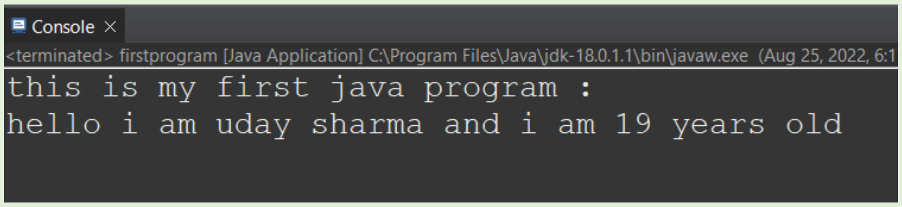


- ***In this the file name is the same as the class name in the java:-***

1. **public static void main(String[]args){..} :-**
   - This is the main() method of our Java program.
   - Every Java program must contain the main() method.
2. **System.out.println("Hello World"):-**
   - The above code is used to display the output on the screen.
     Anything passed inside the inverted commas is printed on the screen as plain text.



```java

public class firstprogram {

    public static void main(String[] args) {
    int age=19;
     System.out.println("this is my first java program :");
     System.out.print("hello i am uday sharma and i am "+age+" years old");
    }

}
```



```
<terminated> firstprogram [Java Application] C:\Program Files\Java\jdk-18.0.1.1\bin\javaw.exe (Aug 25, 2022, 6:1
this is my first java program :
hello i am uday sharma and i am 19 years old
```

- **Naming Conventions:-**
- For classes, we use Pascal Convention. The first and Subsequent characters from a word are capital letters (uppercase).
  Example: Main, MyScanner, MyEmployee, codewithme
- For functions and variables, we use camelCaseConvention. Here the first character is lowercase, and the subsequent characters are uppercase like myScanner, myMarks.

- ***Java Comments:-***

Comments in any programming language are ignored by the compiler or the interpreter. A comment is a part of the coding file that the programmer does not want to execute, rather the programmer uses it to either explain a block of code or to avoid the execution of a specific part of code while testing.

There are two types of comments:

1. Single-line comment
2. Multi-line comment

1). Single Line Comments:-

To write a single-line comment just add a '//' at the start of the line.
Example:-

```
public static void main(String[] args) {
     //This is a single line comment
     System.out.println("Hello World!!!");
```

2).Multi-Line Comments:-To write a multi-line comment just add a '/*.......*/' at the start of the line.
Example:

```
public static void main(String[] args) {
     /*This
      * is
      * a
      * Multiline
      * Comment
      */
     System.out.println("Hello World!!!");
```

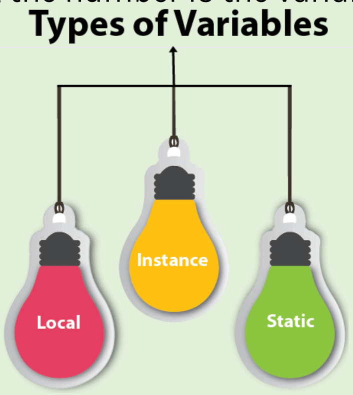
- ***Variables:-***

1. A variable is a container that stores a value.
2. This value can be changed during the execution of the program.
3. Example: int number = 8; (Here, int is a data type, the number is the variable name, and 8 is the value it contains/stores).

- ***Types of Variables***

There are three types of variables in Java:

- local variable
- instance variable
- static variable



1) Local Variable:-

A variable declared inside the body of the method is called local variable. You can use this variable only within that method and the other methods in the class aren't even aware that the variable exists.

A local variable cannot be defined with "static" keyword.

2) Instance Variable:-

A variable declared inside the class but outside the body of the method, is called an instance variable. It is not declared as static.

It is called an instance variable because its value is instance-specific and is not shared among instances.

3) Static variable:-

A variable that is declared as static is called a static variable. It cannot be local. You can create a single copy of the static variable and share it among all the instances of the class. Memory allocation for static variables happens only once when the class is loaded in the memory.

Rules for declaring a variable name:-

We can choose a name while declaring a Java variable if the following rules are followed:

- Must not begin with a digit. (E.g., 1arry is an invalid variable)
- Name is case sensitive. (Harry and harry are different)
- Should not be a keyword (like Void).
- White space is not allowed. (int Code With Harry is invalid)
- Can contain alphabets, $character, _character, and digits if the other conditions are met.

Data Types:-

Data types in Java fall under the following categories:-

1. Primitive Data Types (Intrinsic).
2. Non-Primitive Data Types (Derived).

- ***Primitive Data Types:-***

Java is statically typed, i.e., variables must be declared before use. Java supports 8 primitive data types:

boolean , byte ,char ,short ,int ,long ,float ,double data type

Data Type	Size	Value Range
1. Byte	1 byte	-128 to 127
2. short	1 byte	-32,768 to 32,767
3. int	2 byte	-2,147,483,648 to 2,147,483,647
4. float	4 byte	$3.40282347 \times 10^{38}$ to $1.40239846 \times 10^{-45}$
5. long	8 byte	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
6. double	8 byte	$1.7976931348623157 \times 10^{308}$, $4.9406564584124654 \times 10^{-324}$
7. char	2 byte	0 to 65,535
8. boolean	Depends on JVM	True or False

- ***Non-Primitive Data Types:-***

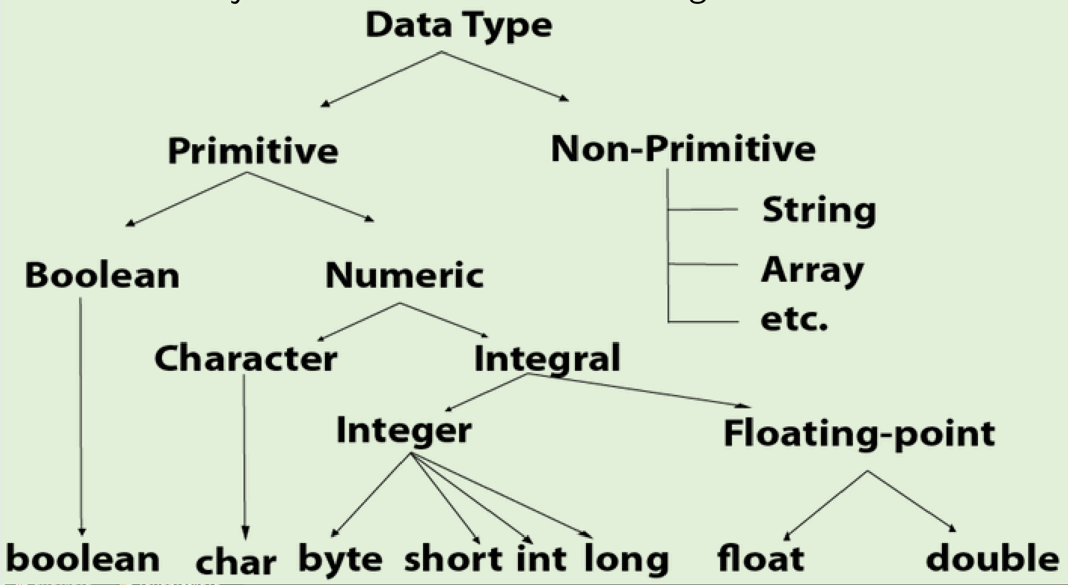
  These are of variable size & are usually declared with a ‘new’ keyword.

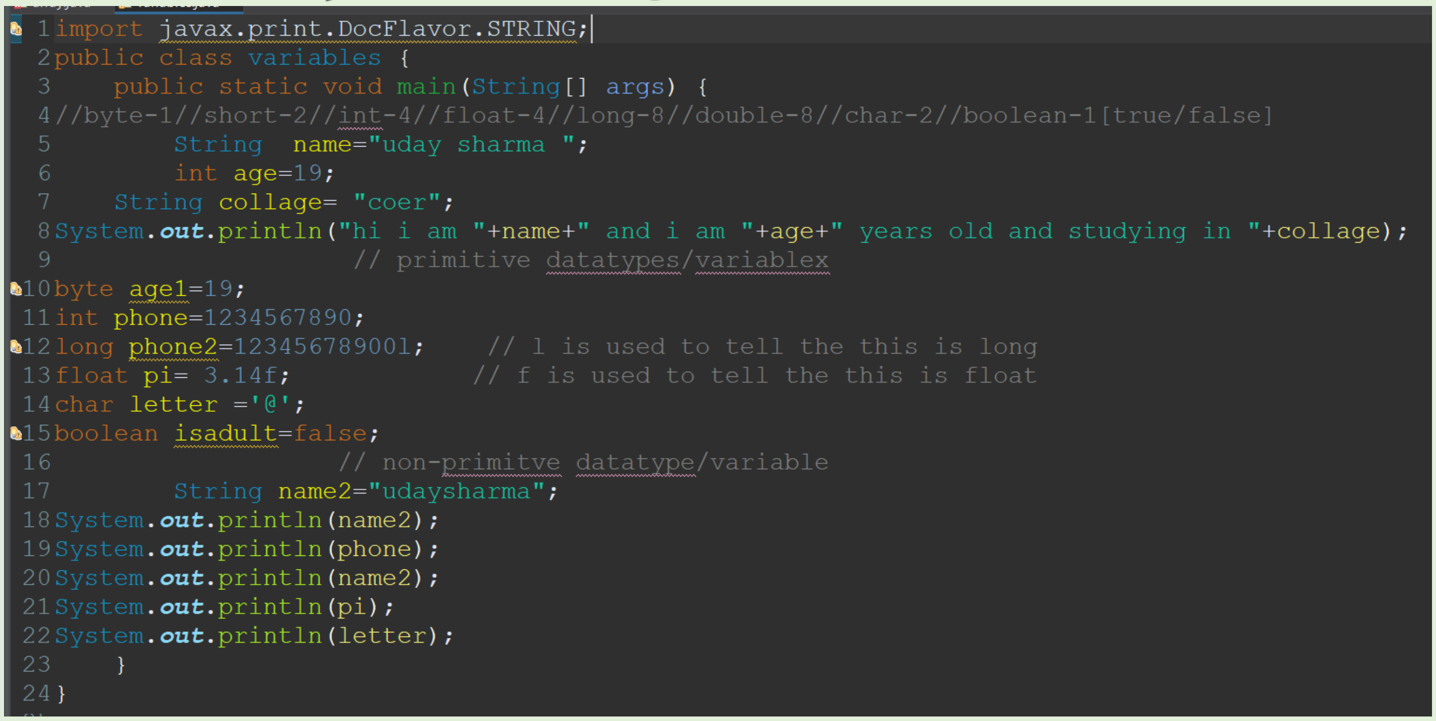
  Eg : String, Arrays

```
String name = new String("uday sharma");
int[] marks = new int[3];
marks[0] = 97;
marks[1] = 98;
marks[2] = 95;
```

- ***How to choose data types for our variables: -***

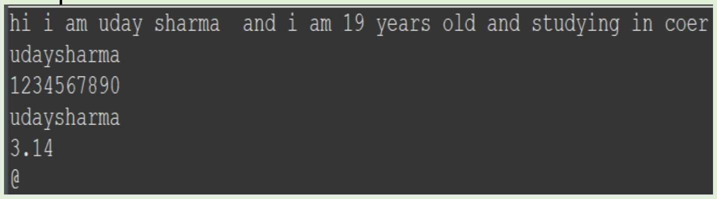
In order to choose the data type, we first need to find the type of data we want to store. After that, we need to analyze the min & max value we might use.





```java
import javax.print.DocFlavor.STRING;
public class variables {
    public static void main(String[] args) {
//byte-1//short-2//int-4//float-4//long-8//double-8//char-2//boolean-1[true/false]
        String  name="uday sharma ";
        int age=19;
    String collage= "coer";
System.out.println("hi i am "+name+" and i am "+age+" years old and studying in "+collage);
                    // primitive datatypes/variablex
byte age1=19;
int phone=1234567890;
long phone2=12345678900l;     // l is used to tell the this is long
float pi= 3.14f;             // f is used to tell the this is float
char letter ='@';
boolean isadult=false;
                   // non-primitve datatype/variable
        String name2="udaysharma";
System.out.println(name2);
System.out.println(phone);
System.out.println(name2);
System.out.println(pi);
System.out.println(letter);
    }
}
```

Output:-



```
hi i am uday sharma  and i am 19 years old and studying in coer
udaysharma
1234567890
udaysharma
3.14
@
```

- ***Literals :-***

A constant value that can be assigned to the variable is called a literal.

- 101 – Integer literal
- 10.1f – float literal

- 10.1 – double literal (default type for decimals)
- 'A' – character literal
- true – Boolean literal
- "Harry" – String literal
- ***Keywords :-***

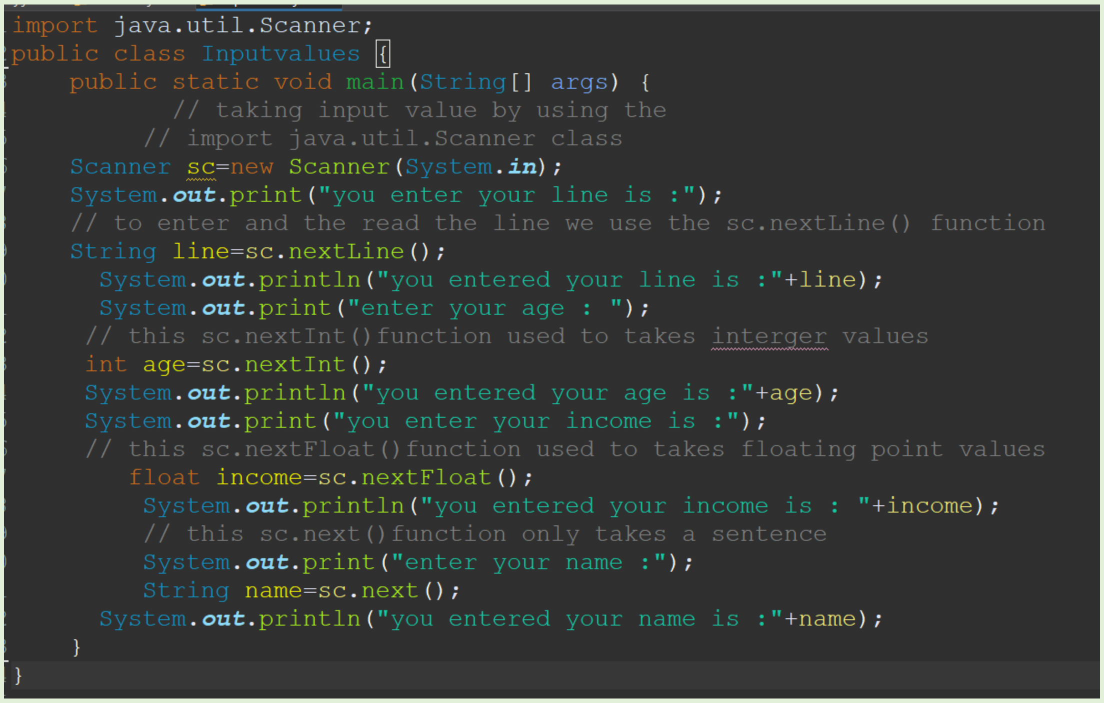
Words that are reserved and used by the Java compiler. They cannot be used as an Identifier. Java keywords are also known as reserved words. Keywords are particular words that act as a key to a code. These are predefined words by Java so they cannot be used as a variable or object name or class name.
Ex:-Abstract ,Boolean ,void ,public ,break ,byte ,case ,char ,class ,continue ,default , interface ,throw, throws,final ,finally ,finalise ,do ,for ,while ,float ,int, String ,if , else ,else if , long, new ,null ,protected ,private ,short ,static , super ,this ,try ,catch etc...

Taking Input:-

We take input using the Scanner class and input various types of data using it.
To import the Scanner class - `import java.util.Scanner;`
Example :

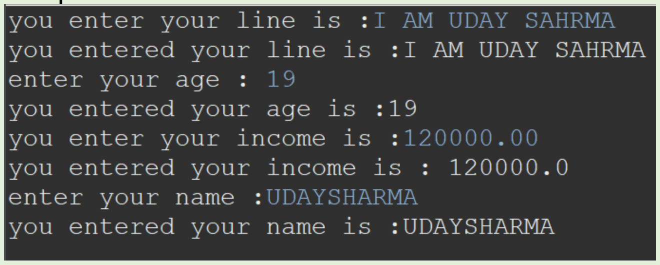
```
Scanner sc = new Scanner(System.in);
int n = sc.nextInt();
float a = sc.nextFloat();
String name = sc.next();
String line = sc.nextLine();
```

Code:-



```
import java.util.Scanner;
public class Inputvalues {
    public static void main(String[] args) {
           // taking input value by using the
         // import java.util.Scanner class
    Scanner sc=new Scanner(System.in);
    System.out.print("you enter your line is :");
    // to enter and the read the line we use the sc.nextLine() function
    String line=sc.nextLine();
      System.out.println("you entered your line is :"+line);
      System.out.print("enter your age : ");
     // this sc.nextInt()function used to takes interger values
     int age=sc.nextInt();
     System.out.println("you entered your age is :"+age);
     System.out.print("you enter your income is :");
     // this sc.nextFloat()function used to takes floating point values
        float income=sc.nextFloat();
         System.out.println("you entered your income is : "+income);
         // this sc.next()function only takes a sentence
         System.out.print("enter your name :");
         String name=sc.next();
      System.out.println("you entered your name is :"+name);
    }
}
```

Output:-



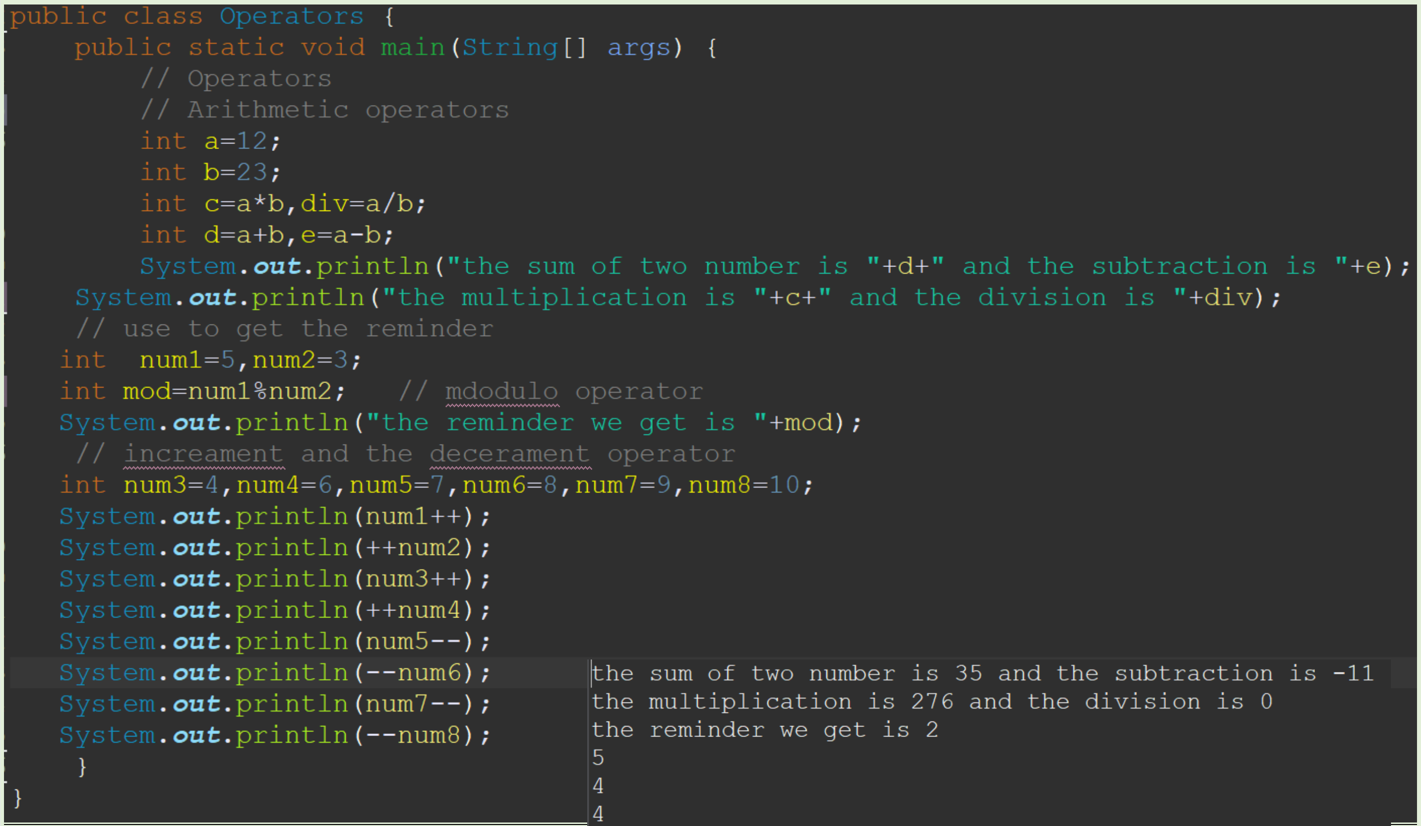
```
you enter your line is :I AM UDAY SAHRMA
you entered your line is :I AM UDAY SAHRMA
enter your age : 19
you entered your age is :19
you enter your income is :120000.00
you entered your income is : 120000.0
enter your name :UDAYSHARMA
you entered your name is :UDAYSHARMA
```

- **Types of operators :-**

1. **Arithmetic Operators :-**

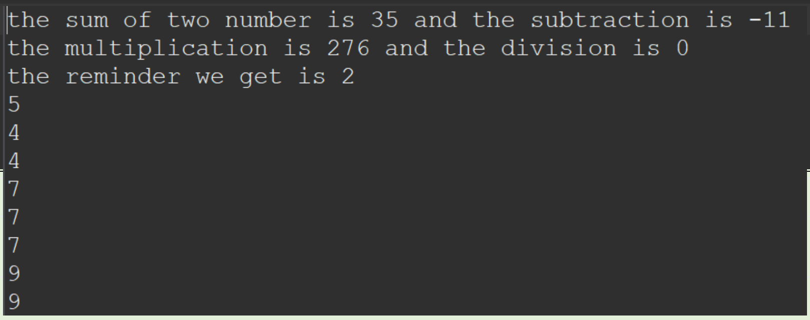
Operator	Description	Example
+ (Addition)	Used to add two numbers	x + y = 9
- (Subtraction)	Used to subtract the right-hand side value from the left-hand side value	x - y = 5
* (Multiplication)	Used to multiply two values.	x * y = 14
/ (Division)	Used to divide left-hand Value by right-hand value.	x / y = 3
% (Modulus)	Used to print the remainder after dividing the left-hand side value from the right-hand side value.	x % y = 1
++ (Increment)	Increases the value of operand by 1.	x++ = 8
-- (Decrement)	Decreases the value of operand by 1.	y-- = 1

Code:-



```java
public class Operators {
    public static void main(String[] args) {
        // Operators
        // Arithmetic operators
        int a=12;
        int b=23;
        int c=a*b,div=a/b;
        int d=a+b,e=a-b;
        System.out.println("the sum of two number is "+d+" and the subtraction is "+e);
    System.out.println("the multiplication is "+c+" and the division is "+div);
    // use to get the reminder
   int  num1=5,num2=3;
   int mod=num1%num2;   // mdodulo operator
   System.out.println("the reminder we get is "+mod);
    // increament and the decerament operator
   int num3=4,num4=6,num5=7,num6=8,num7=9,num8=10;
   System.out.println(num1++);
   System.out.println(++num2);
   System.out.println(num3++);
   System.out.println(++num4);
   System.out.println(num5--);
   System.out.println(--num6);
   System.out.println(num7--);
   System.out.println(--num8);
    }
}
```

Output:-



```
the sum of two number is 35 and the subtraction is -11
the multiplication is 276 and the division is 0
the reminder we get is 2
5
4
4
7
7
7
9
9
```

2. ***Comparison Operators :-***

- Let x=7 and y=2.

Operator	Description	Example
== (Equal to)	Checks if two operands are equal. Returns a boolean value.	x == y --> False
!= (Not equal	Checks if two operands are not equal. Returns a boolean value.	x != y --> True

> (Greater than)	Checks if the left-hand side value is greater than the right-hand side value. Returns a boolean value.	x > y --> True
< (Less than)	Checks if the left-hand side value is smaller than the right-hand side value. Returns a boolean value.	x < y --> False
>=(Greater than or equal to)	Checks if the left-hand side value is greater than or equal to the right-hand side value. Returns a boolean value.	x >= y --> True
<= (Less than or equal to)	Checks if the left-hand side value is less than or equal to the right-hand side value. Returns a boolean value.	x <= y -->False

3. ***Logical Operators :-***

- Let x = 8 and y =2

&& (logical and)	Returns true if both operands are true.	x<y && x!=y --> True
|| (logical or)	Returns true if any of the operand is true.	x<y && x==y --> True
! (logical not)	Returns true if the result of the expression is false and vice-versa	!(x<y && x==y) --> False

4. **Bitwise Operators :-**

- These operators perform the operations on every bit of a number.
- Let x =2 and y=3. So 2 in binary is 100, and 3 is 011.

Operator	**Description**	**Example**
& (bitwise and)	1&1 =1, 0&1=0,1&0=0,1&1=1, 0&0 =0	(A & B) = (100 & 011) = 000
| (bitwise or)	1&0 =1, 0&1=1,1&1=1, 0&0=0	(A | B) = (100 | 011) = 111
^ (bitwise XOR)	1&0 =1, 0&1=1,1&1=0, 0&0=0	(A ^ B) = (100 ^ 011) = 111
<< (left shift)	This operator moves the value left by the number of bits specified.	13<<2 = 52(decimal)
>> (right shift)	This operator moves the value left by the number of bits specified.	13>>2 = 3(decimal)

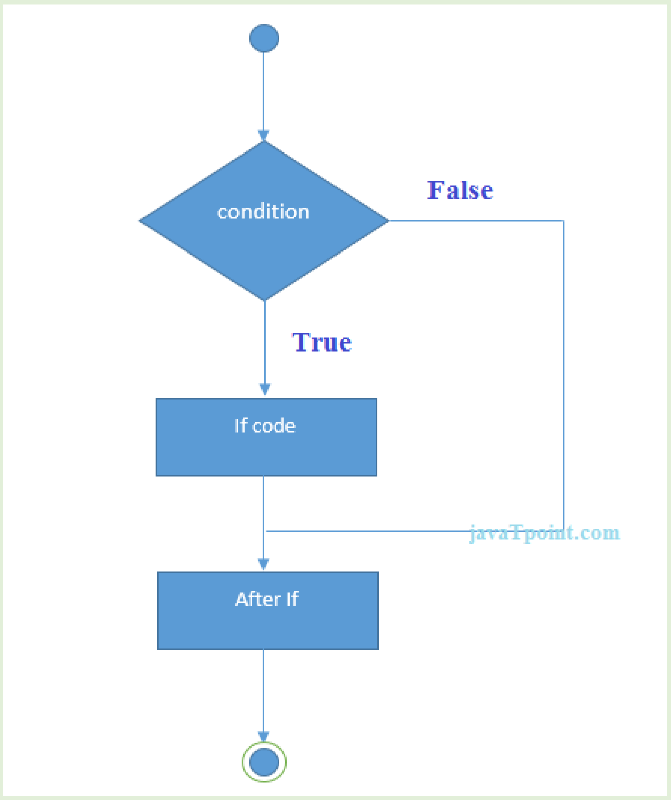
- ***Conditional Statements 'if-else':-***
- The if block is used to specify the code to be executed if the condition specified in if is true, the else block is executed otherwise.
- The Java *if statement* is used to test the condition. It checks boolean condition: *true* or *false*. There are various types of if statement in Java.
  - if statement
  - if-else statement
  - if-else-if ladder
  - nested if statement

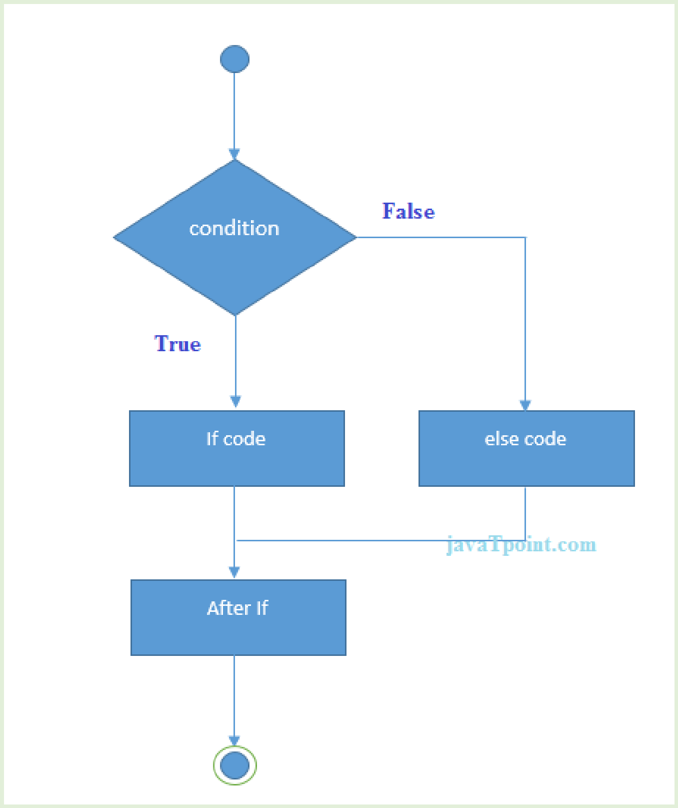
#*if Statement:-*

The Java if statement tests the condition. It executes the *if block* if condition is true.

Syntax:

```
if(condition){
//code to be executed
}
```





If condition if-else condition

# if-else Statement:- The Java if-else statement also tests the condition. It executes the *if block* if condition is true otherwise *else block* is executed.

Syntax:-

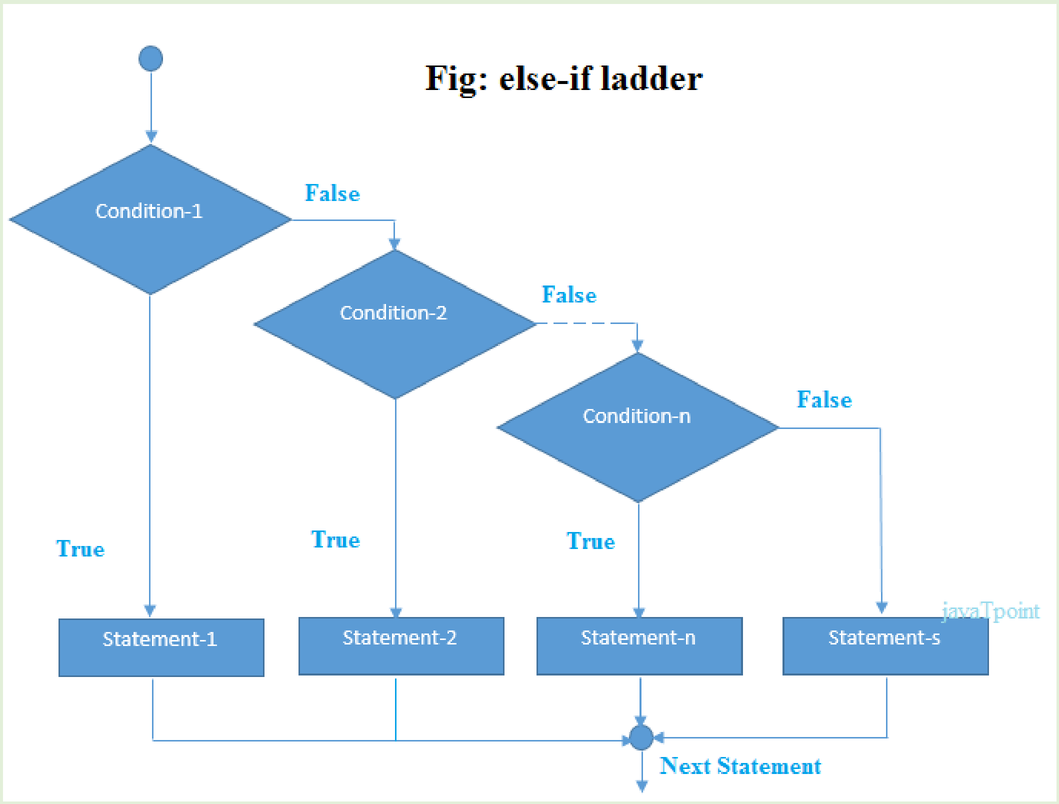
```
if (condition-to-be-checked) {
 //statements-if-condition-true;
 }
else {
//statements-if-condition-false;
  }
```

# if-else ladder syntax:-

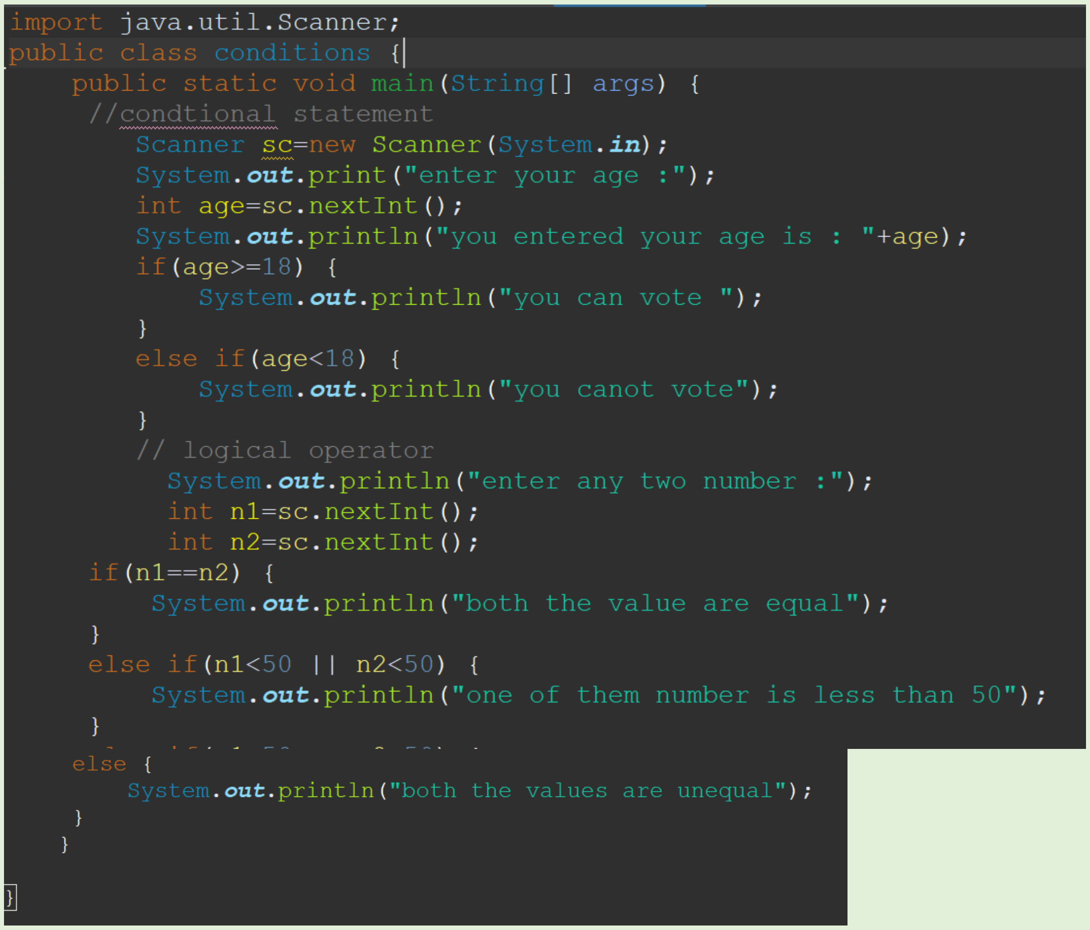
The if-else-if ladder statement executes one condition from multiple statements.

Syntax:-

```
if (condition1) {
        //Statements;
  }
else if (condition2){
        //Statements;
}
else {
//statements
  }
```

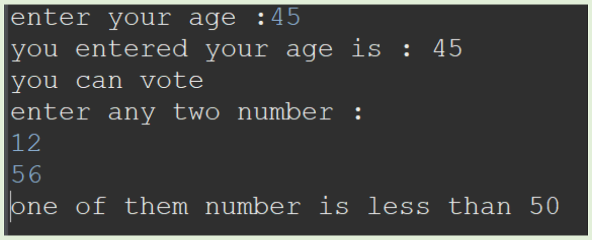


Code:- this code will demonstrate both if-else and if-else ladder the statements:-



```
import java.util.Scanner;
public class conditions {
    public static void main(String[] args) {
     //condtional statement
         Scanner sc=new Scanner(System.in);
         System.out.print("enter your age :");
         int age=sc.nextInt();
         System.out.println("you entered your age is : "+age);
         if(age>=18) {
             System.out.println("you can vote ");
         }
         else if(age<18) {
             System.out.println("you canot vote");
         }
         // logical operator
           System.out.println("enter any two number :");
           int n1=sc.nextInt();
           int n2=sc.nextInt();
     if(n1==n2) {
         System.out.println("both the value are equal");
     }
     else if(n1<50 || n2<50) {
         System.out.println("one of them number is less than 50");
     }
    else {
        System.out.println("both the values are unequal");
    }
   }

}
```

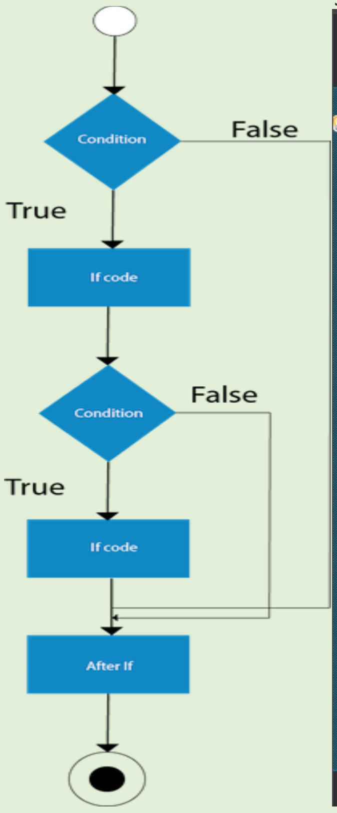
Output:-

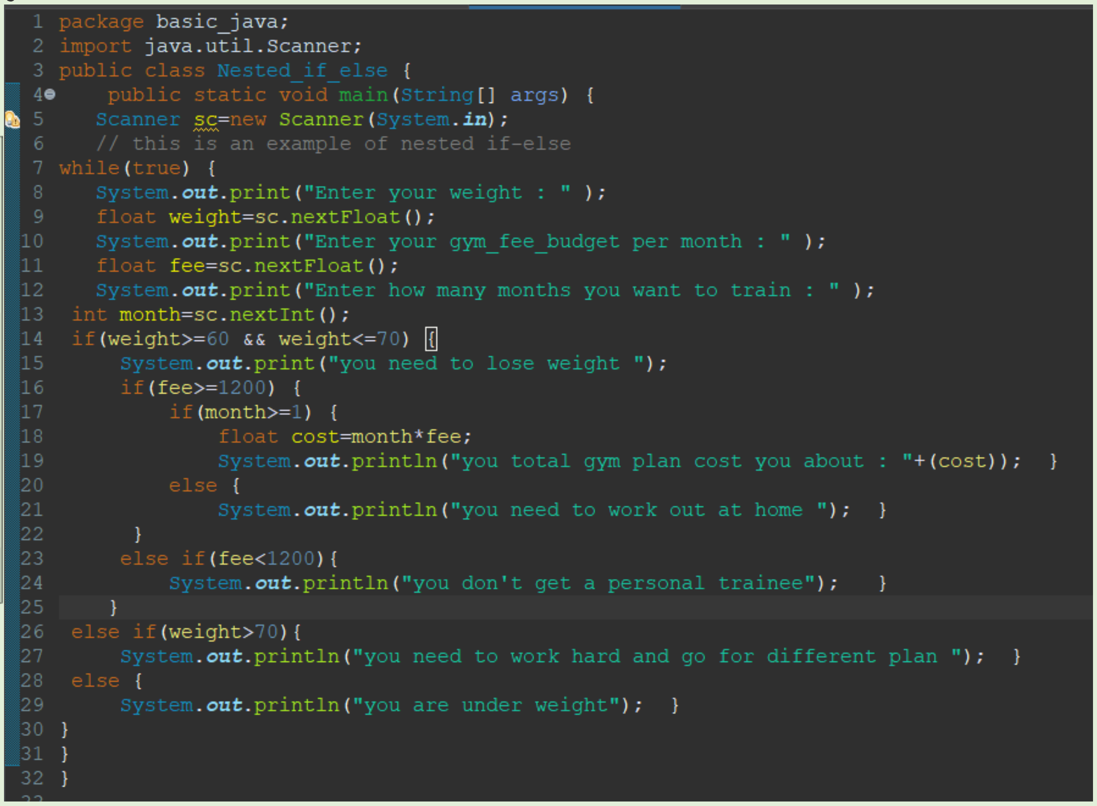


```
enter your age :45
you entered your age is : 45
you can vote
enter any two number :
12
56
one of them number is less than 50
```

#Java Nested if statement:-

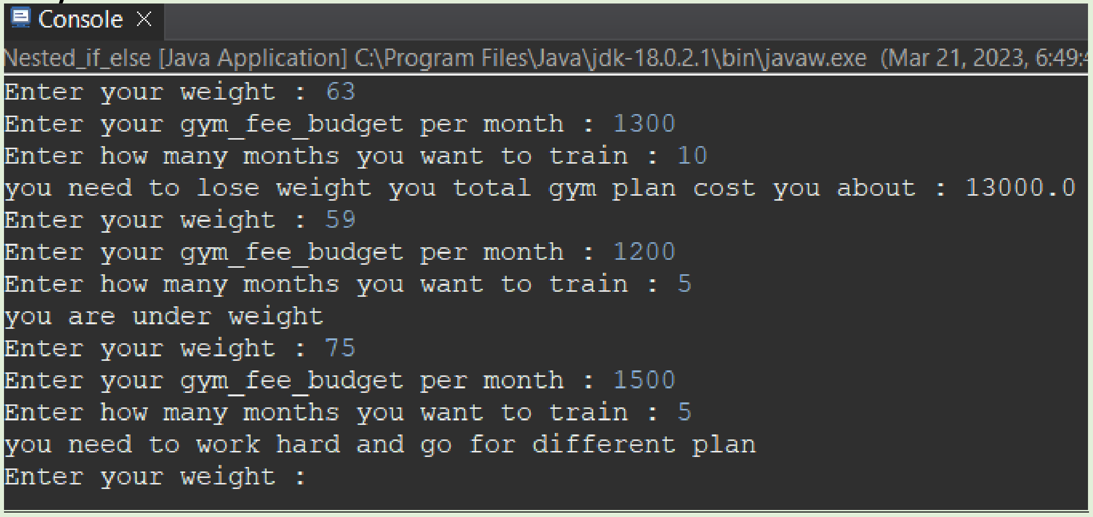
The nested if statement represents the *if block within another if block*. Here, the inner if block condition executes only when outer if block condition is true.





```
package basic_java;
import java.util.Scanner;
public class Nested_if_else {
     public static void main(String[] args) {
    Scanner sc=new Scanner(System.in);
    // this is an example of nested if-else
while(true) {
    System.out.print("Enter your weight : " );
    float weight=sc.nextFloat();
    System.out.print("Enter your gym_fee_budget per month : " );
    float fee=sc.nextFloat();
    System.out.print("Enter how many months you want to train : " );
  int month=sc.nextInt();
  if(weight>=60 && weight<=70) {
      System.out.print("you need to lose weight ");
      if(fee>=1200) {
          if(month>=1) {
              float cost=month*fee;
              System.out.println("you total gym plan cost you about : "+(cost));  }
          else {
              System.out.println("you need to work out at home ");  }
       }
      else if(fee<1200){
          System.out.println("you don't get a personal trainee");   }
    }
  else if(weight>70){
      System.out.println("you need to work hard and go for different plan ");  }
  else {
      System.out.println("you are under weight");  }
}
}
}
```

Output:-



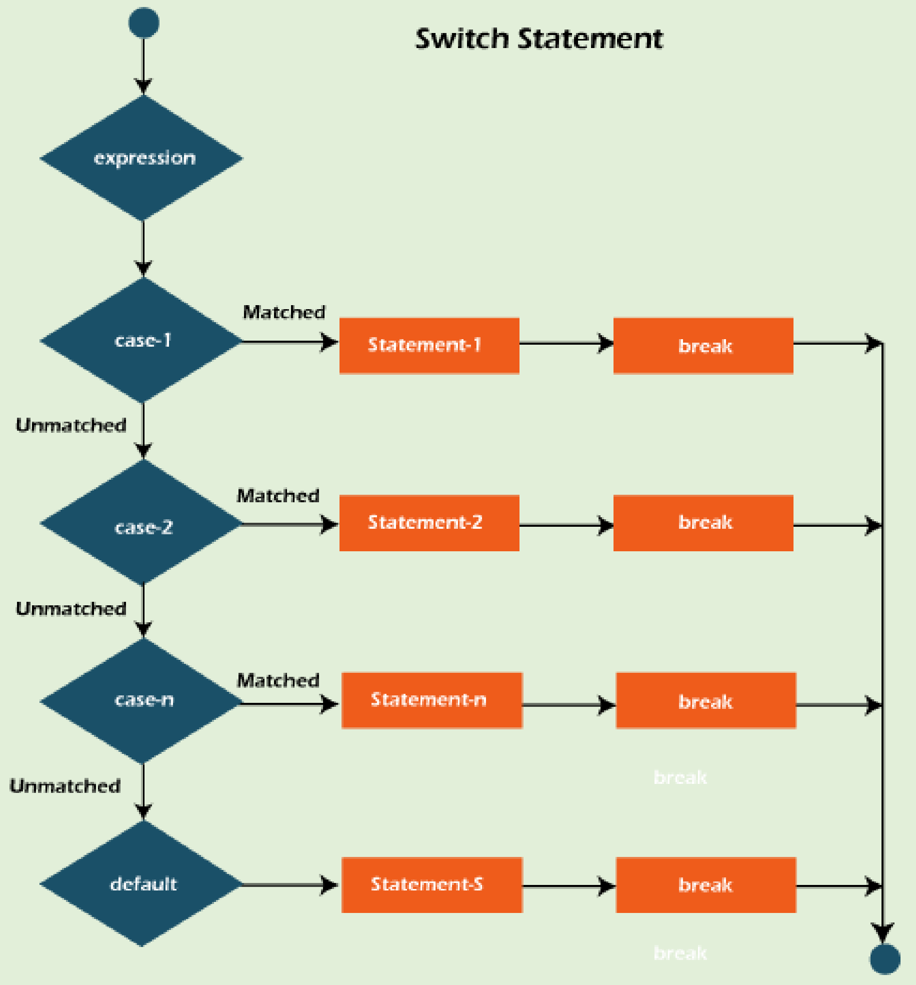
```
Console ×
Nested_if_else [Java Application] C:\Program Files\Java\jdk-18.0.2.1\bin\javaw.exe  (Mar 21, 2023, 6:49:
Enter your weight : 63
Enter your gym_fee_budget per month : 1300
Enter how many months you want to train : 10
you need to lose weight you total gym plan cost you about : 13000.0
Enter your weight : 59
Enter your gym_fee_budget per month : 1200
Enter how many months you want to train : 5
you are under weight
Enter your weight : 75
Enter your gym_fee_budget per month : 1500
Enter how many months you want to train : 5
you need to work hard and go for different plan
Enter your weight :
```

- ***Conditional Statements 'switch':-***

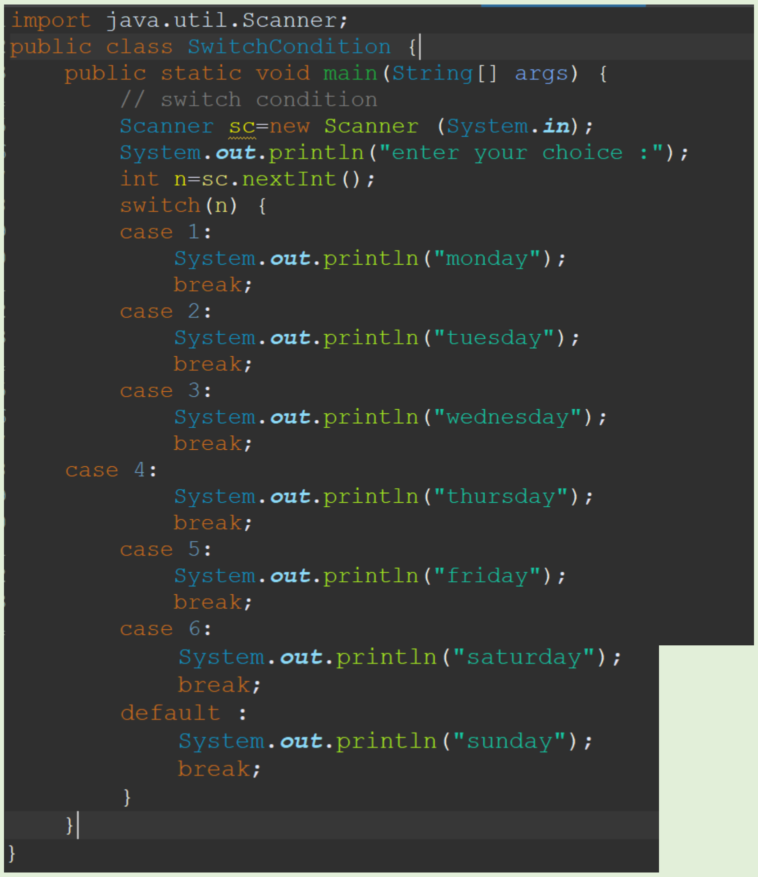
- The Java *switch statement* executes one statement from multiple conditions. It is like if-else-if ladder statement. The switch statement works with byte, short, int, long, enum types, String and some wrapper types like Byte, Short, Int, and Long. Since Java 7, you can use strings in the switch statement.
- In other words, the switch statement tests the equality of a variable against multiple values.
- Switch case statements are a substitute for long if statements that compare a variable to multiple values. After a match is found, it executes the corresponding code of that value case.

Syntax:-

```
Switch(var) {
Case C1:
        //Code;
        break;
Case C2:
        //Code;
        break;
Case C3:
        //Code
        break;
default:
        //Code   }
```

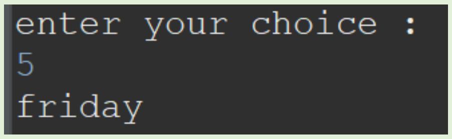


Code:-



```
import java.util.Scanner;
public class SwitchCondition {
    public static void main(String[] args) {
        // switch condition
        Scanner sc=new Scanner (System.in);
        System.out.println("enter your choice :");
        int n=sc.nextInt();
        switch(n) {
        case 1:
            System.out.println("monday");
            break;
        case 2:
            System.out.println("tuesday");
            break;
        case 3:
            System.out.println("wednesday");
            break;
    case 4:
            System.out.println("thursday");
            break;
        case 5:
            System.out.println("friday");
            break;
        case 6:
            System.out.println("saturday");
            break;
        default :
            System.out.println("sunday");
            break;
        }
    }
}
```

Output:-

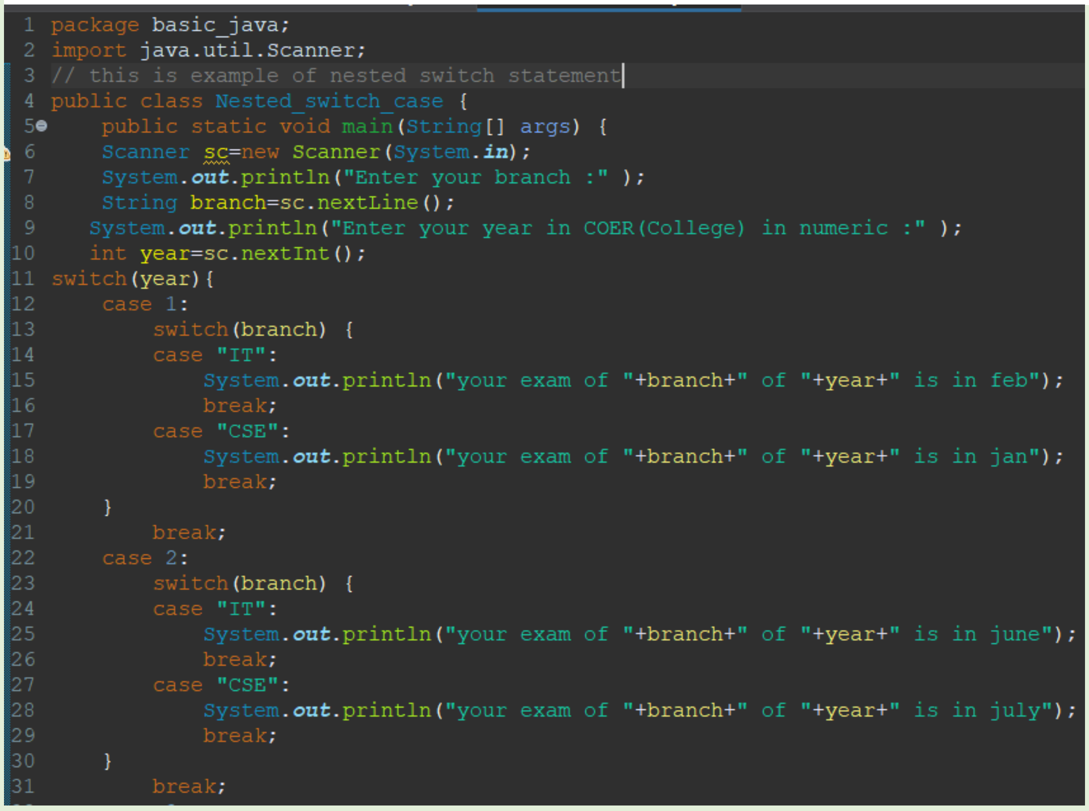


```
enter your choice :
5
friday
```

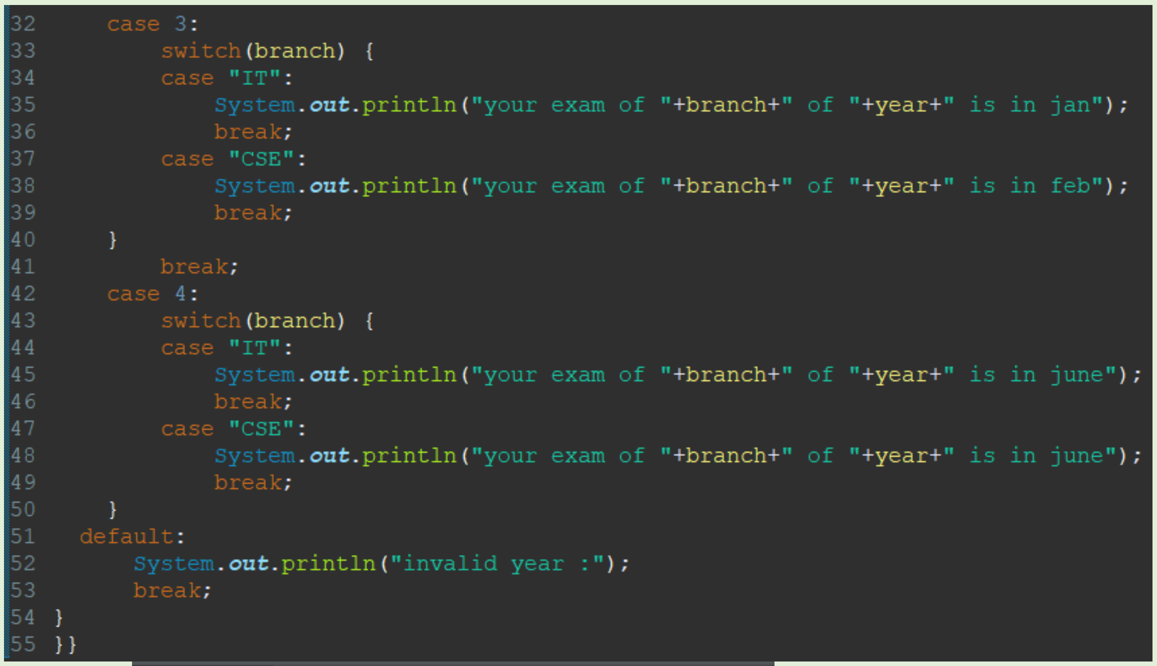
Note:-we can Nested Switch case like we use the nested if-else Statement:-

- ***Java Nested-switch Statement:-***

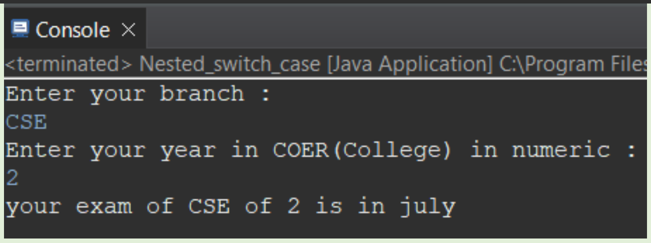
We can use switch statement inside other switch statement in Java. It is known as nested switch statement.



```
package basic_java;
import java.util.Scanner;
// this is example of nested switch statement
public class Nested_switch_case {
    public static void main(String[] args) {
    Scanner sc=new Scanner(System.in);
    System.out.println("Enter your branch :" );
    String branch=sc.nextLine();
   System.out.println("Enter your year in COER(College) in numeric :" );
   int year=sc.nextInt();
switch(year){
    case 1:
        switch(branch) {
        case "IT":
            System.out.println("your exam of "+branch+" of "+year+" is in feb");
            break;
        case "CSE":
            System.out.println("your exam of "+branch+" of "+year+" is in jan");
            break;
    }
        break;
    case 2:
        switch(branch) {
        case "IT":
            System.out.println("your exam of "+branch+" of "+year+" is in june");
            break;
        case "CSE":
            System.out.println("your exam of "+branch+" of "+year+" is in july");
            break;
    }
        break;
```



```
    case 3:
        switch(branch) {
        case "IT":
            System.out.println("your exam of "+branch+" of "+year+" is in jan");
            break;
        case "CSE":
            System.out.println("your exam of "+branch+" of "+year+" is in feb");
            break;
    }
        break;
    case 4:
        switch(branch) {
        case "IT":
            System.out.println("your exam of "+branch+" of "+year+" is in june");
            break;
        case "CSE":
            System.out.println("your exam of "+branch+" of "+year+" is in june");
            break;
    }
  default:
      System.out.println("invalid year :");
      break;
 }
 }}
```

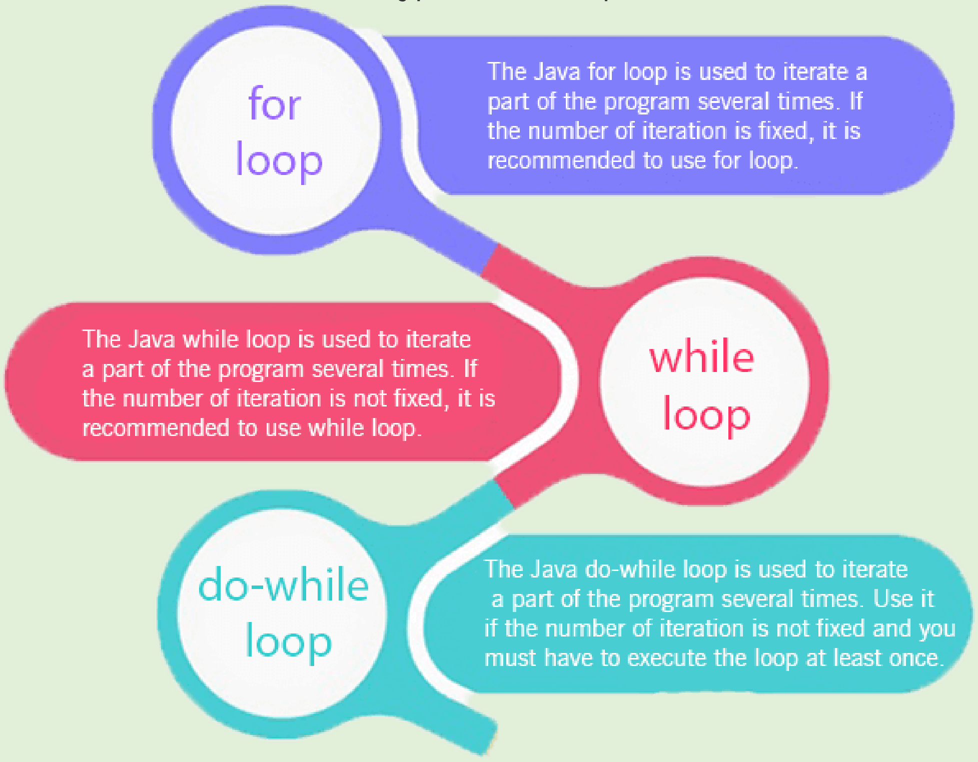


```
<terminated> Nested_switch_case [Java Application] C:\Program Files
Enter your branch :
CSE
Enter your year in COER(College) in numeric :
2
your exam of CSE of 2 is in july
```

Output:-

- ***Loops:-***

A loop is used for executing a block of statements repeatedly until a particular condition is satisfied. A loop consists of an initialization statement, a test condition and an increment statement. There are three types of for loops in Java:-

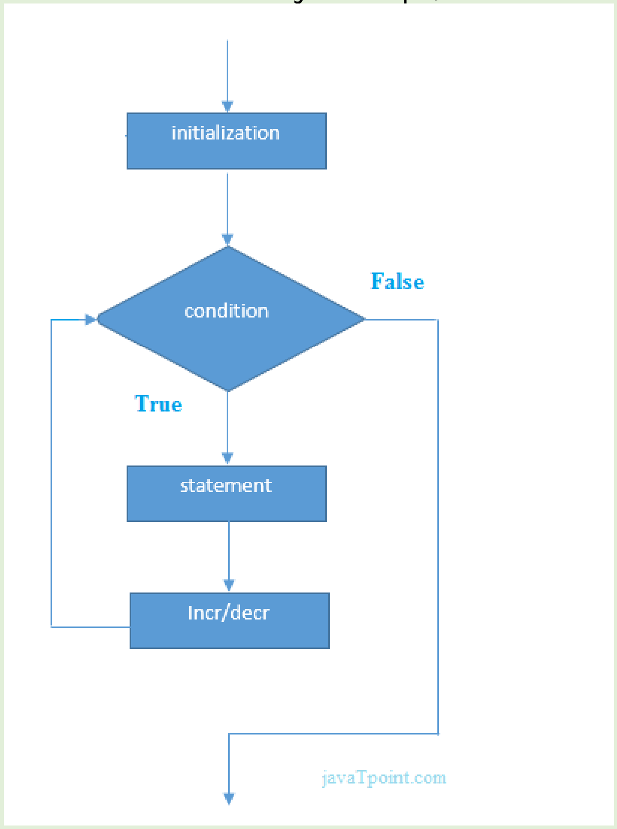

for loop
The Java for loop is used to iterate a part of the program several times. If the number of iteration is fixed, it is recommended to use for loop.
while loop
The Java while loop is used to iterate a part of the program several times. If the number of iteration is not fixed, it is recommended to use while loop.
do-while loop
The Java do-while loop is used to iterate a part of the program several times. Use it if the number of iteration is not fixed and you must have to execute the loop at least once.

1. ***For Loop:-***

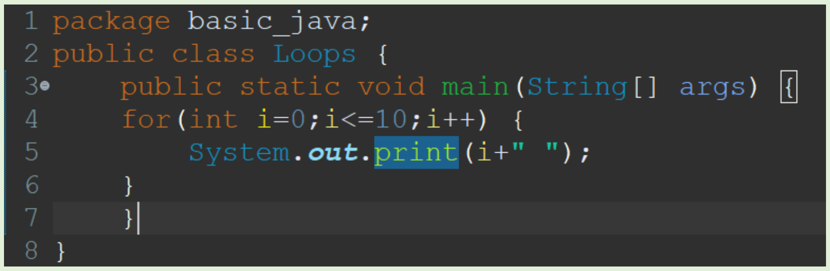
The syntax of the for loop is :-

for (initialization; condition; update) {

// body of-loop }

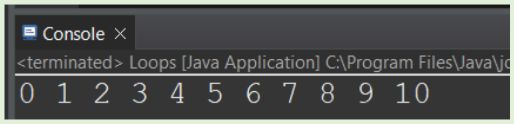


Code:-



```
package basic_java;
public class Loops {
    public static void main(String[] args) {
    for(int i=0;i<=10;i++)  {
        System.out.print(i+" ");
    }
    }
}
```

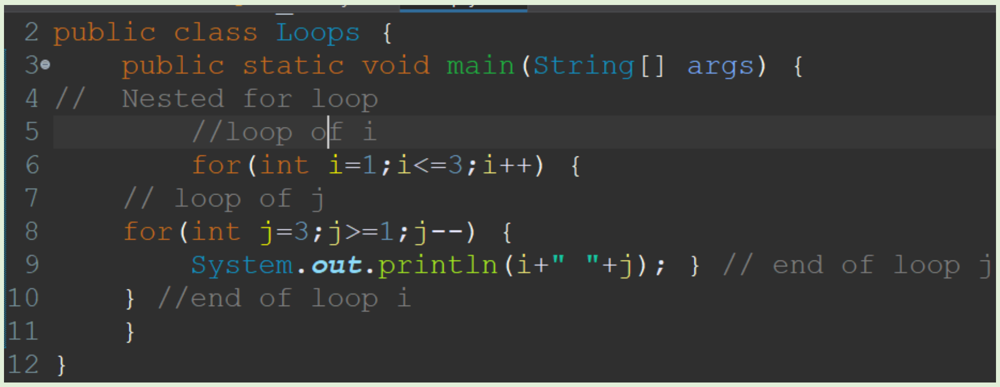
Output:-



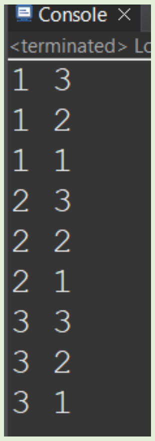
```
<terminated> Loops [Java Application] C:\Program Files\Java\j
0 1 2 3 4 5 6 7 8 9 10
```

Nested for loop:-

If we have a for loop inside the another loop, it is known as nested for loop. The inner loop executes completely whenever outer loop executes.

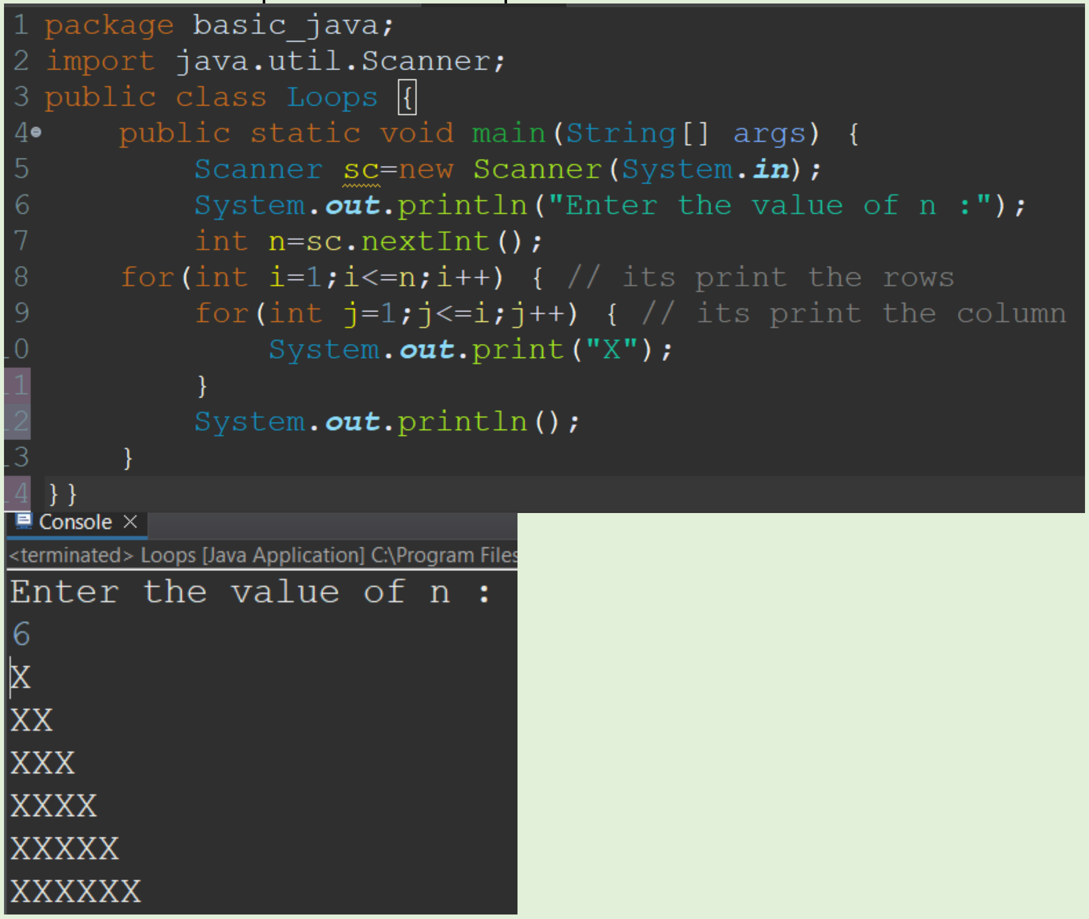


```
public class Loops {
    public static void main(String[] args) {
//  Nested for loop
        //loop of i
        for(int i=1;i<=3;i++) {
    // loop of j
    for(int j=3;j>=1;j--) {
        System.out.println(i+" "+j); } // end of loop j
    } //end of loop i
    }
}
```



```
<terminated> Lo
1 3
1 2
1 1
2 3
2 2
2 1
3 3
3 2
3 1
```

Some other example of Nested loops:-

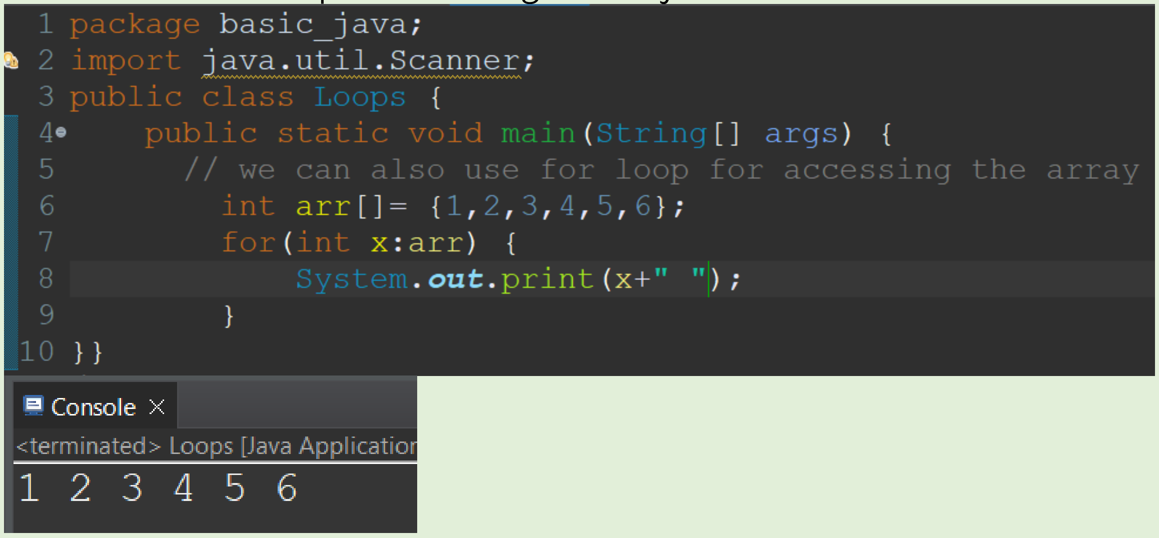


```
package basic_java;
import java.util.Scanner;
public class Loops {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the value of n :");
        int n=sc.nextInt();
    for(int i=1;i<=n;i++) { // its print the rows
        for(int j=1;j<=i;j++) { // its print the column
            System.out.print("X");
        }
        System.out.println();
    }
}}
```

Console

```
<terminated> Loops [Java Application] C:\Program Files
Enter the value of n :
6
X
XX
XXX
XXXX
XXXXX
XXXXXX
```

- We can use for loop for accessing the array like:-



```
package basic_java;
import java.util.Scanner;
public class Loops {
    public static void main(String[] args) {
      // we can also use for loop for accessing the array
        int arr[]= {1,2,3,4,5,6};
        for(int x:arr) {
            System.out.print(x+" ");
        }
}}
```

Console

```
<terminated> Loops [Java Application
1 2 3 4 5 6
```

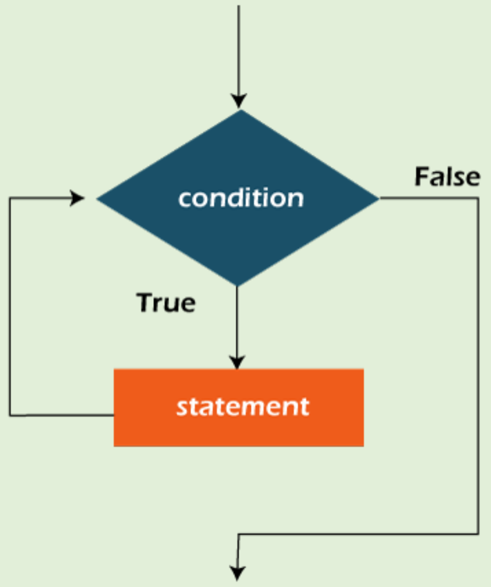
2. While Loop :-

- The Java *while loop* is used to iterate a part of the program repeatedly until the specified Boolean condition is true. As soon as the Boolean condition becomes false, the loop automatically stops.
- The while loop is considered as a repeating if statement. If the number of iteration is not fixed, it is recommended to use the while loop.

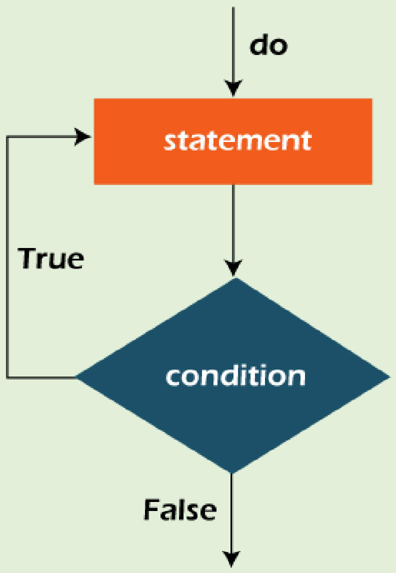
The ***syntax*** for while loop is :

```
    while(condition) {
            // body of the loop }
```

=>While loop:-



=>***Do-while loop:-***

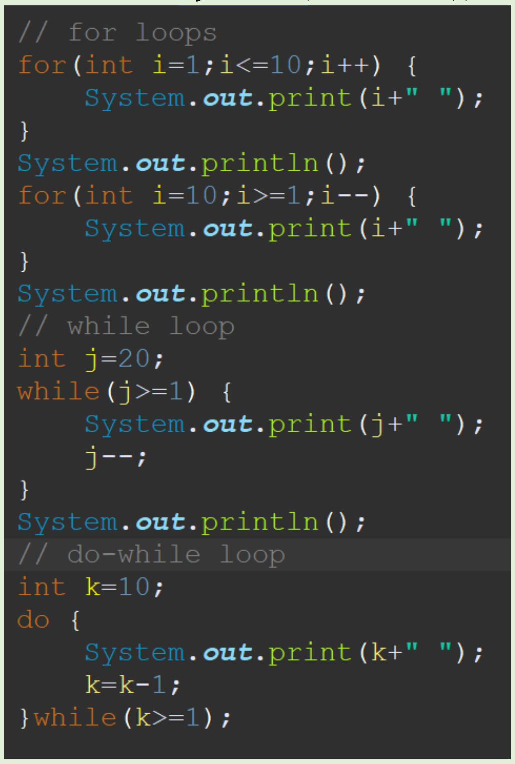


3. Do-While Loop :-

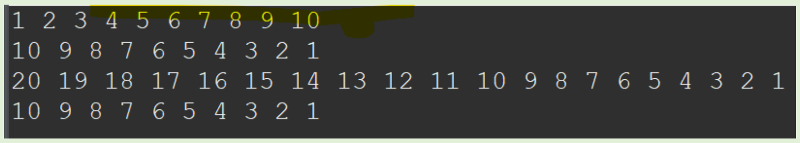
- The Java *do-while loop* is used to iterate a part of the program repeatedly, until the specified condition is true. If the number of iteration is not fixed and you must have to execute the loop at least once, it is recommended to use a do-while loop.
- Java do-while loop is called an **exit control loop**. Therefore, unlike while loop and for loop, the do-while check the condition at the end of loop body. The Java *do-while loop* is executed at least once because condition is checked after loop body.

The syntax for the do-while loop is :

```
do {
        // body of loop;
      } while (condition);
```



```
// for loops
for(int i=1;i<=10;i++) {
    System.out.print(i+" ");
}
System.out.println();
for(int i=10;i>=1;i--) {
    System.out.print(i+" ");
}
System.out.println();
// while loop
int j=20;
while(j>=1) {
    System.out.print(j+" ");
    j--;
}
System.out.println();
// do-while loop
int k=10;
do {
    System.out.print(k+" ");
    k=k-1;
}while(k>=1);
```

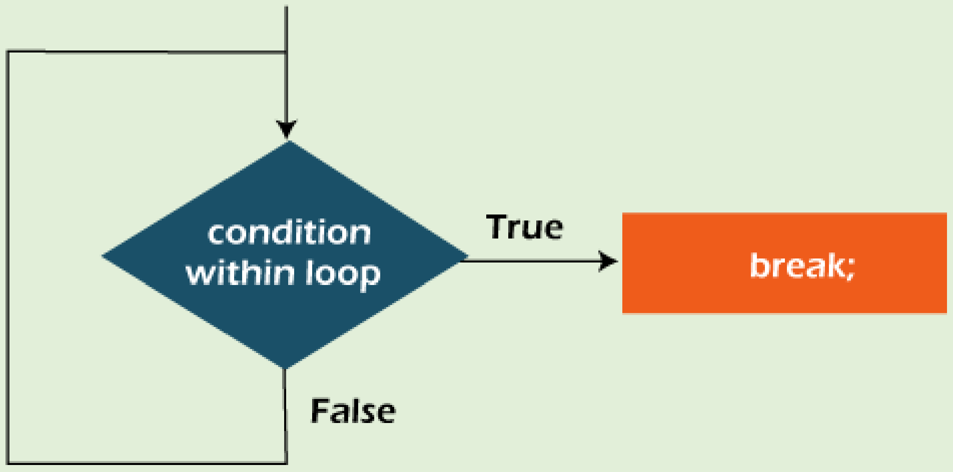


```
1 2 3 4 5 6 7 8 9 10
10 9 8 7 6 5 4 3 2 1
20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
10 9 8 7 6 5 4 3 2 1
```

- ***<u>Break & Continue:-</u>***

Jumps in loops are used to control the flow of loops. There are two statements used to implement jump in loops - Continue and Break. These statements are used when we need to change the flow of the loop when some specified condition is met.

Continue statement is used to skip to the next iteration of that loop. This means that it stops one iteration of the loop. All the statements present after the continue statement in that loop are not executed.



Flowchart of break statement

```
int i;
for (i=7; i>0; i--) { //  Output is :- 7 6 5 4 2 1
  if (i==3) {
     continue; // skip the part
  }
  System.out.println(i);
}
```

In this for loop, whenever i is a number divisible by 3, it will not be printed as the loop will skip to the next iteration due to the continue statement. Hence, all the numbers except those which are divisible by 3 will be printed.

- **Break statement** is used to terminate the current loop. As soon as the break statement is encountered in a loop, all further iterations of the loop are stopped and control is shifted to the first statement after the end of loop.

```
int i;
for (i=1; i<=20; i++) {
  if (i == 11) {
     break;
  }
  System.out.println(i); }
```

In this loop, when i becomes equal to 11, the for loop terminates due to break statement, Hence, the program will print numbers from 1 to 10 only.

- ***<u>Exception Handling (try-catch):-</u>***

Exception Handling in Java is a mechanism to handle the runtime errors so that normal flow of the application can be maintained.

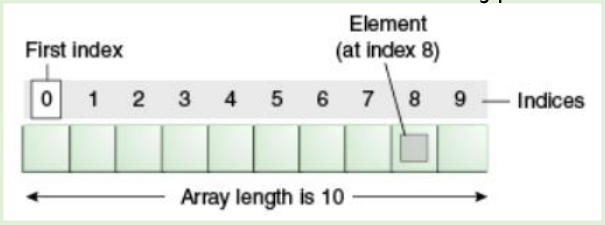
It is done using 2 keywords - '**<u>try</u>**' and '<u>catch</u>'.

Additional keywords like finally, throw and throws can also be used if we dive deep into this concept.

```
int[] marks = {98, 97, 95};
try {
  System.out.println(marks[4]);
} catch (Exception exception) {
  System.out.println("An exception for caught while accessing an index the 'marks' array");
}
System.out.println("We tried to print marks & an exception must have occurred with index >=3");
```

- **Arrays:-**

Arrays in Java are like a list of elements of the same type i.e. a list of integers, character, string ,float , Booleans etc.



- ***Advantages:-***

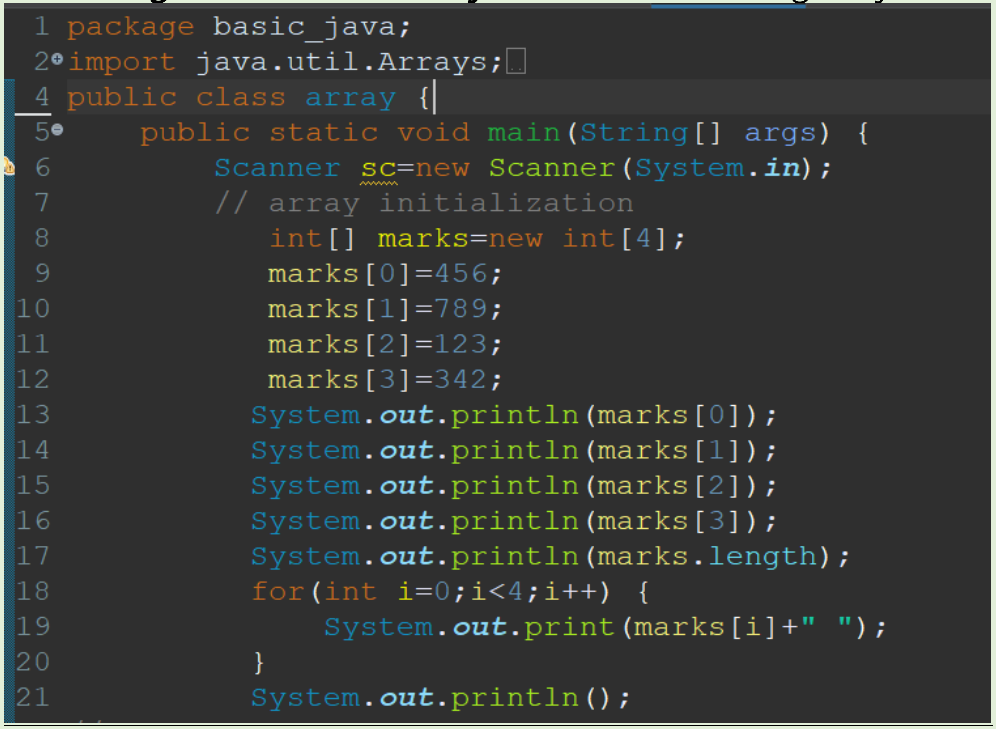
Code Optimization: It makes the code optimized, we can retrieve or sort the data efficiently.
Random access: We can get any data located at an index position.

- ***Disadvantages:-***

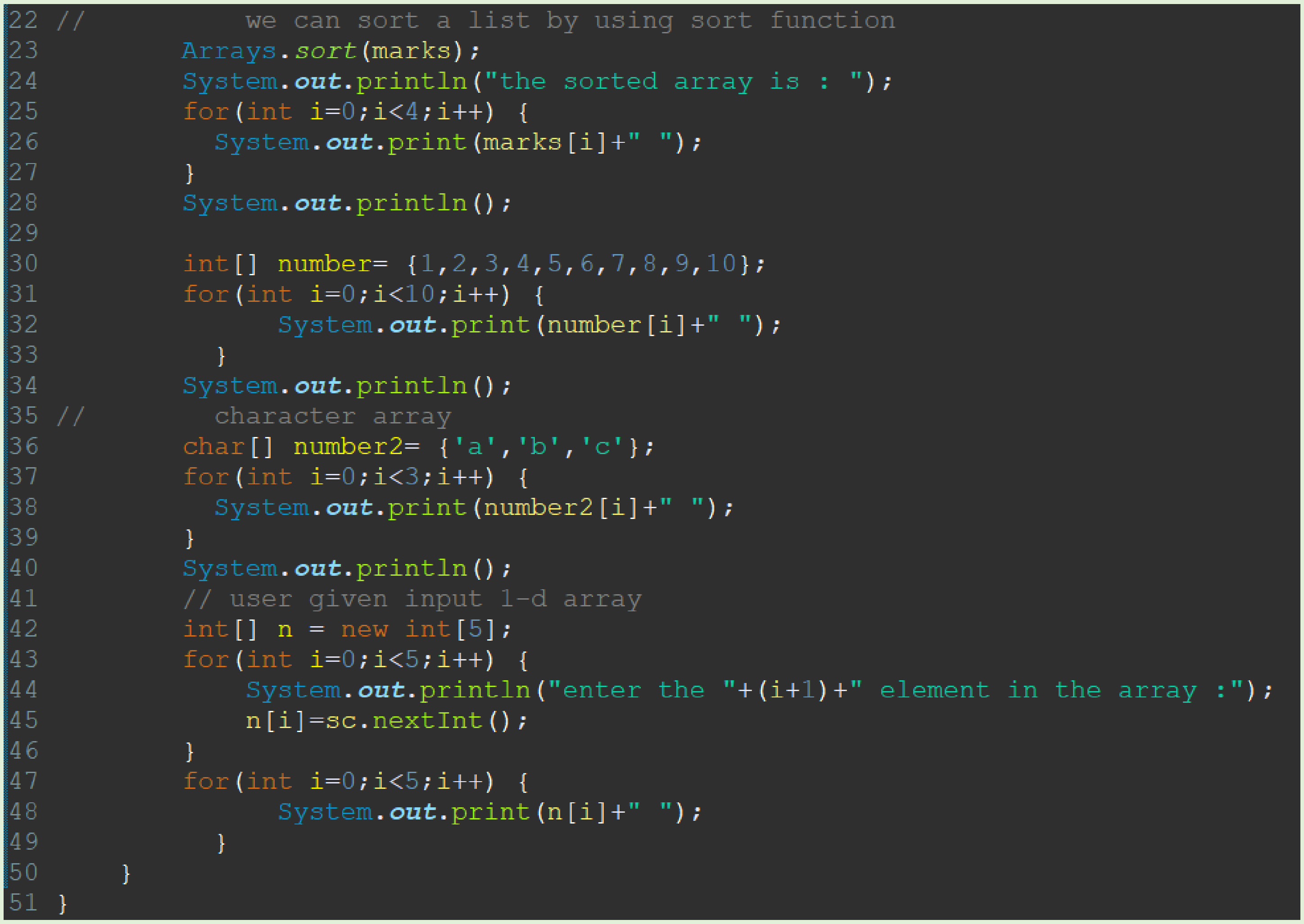
Size Limit: We can store only the fixed size of elements in the array. It doesn't grow its size at runtime. To solve this problem, collection framework is used in Java which grows automatically.
Types of Array in java
There are two types of array.

1. ***Single Dimensional Array.***
2. ***Multidimensional Array.***

- ***Single Dimensional Array:-*** Below code showing arrays.

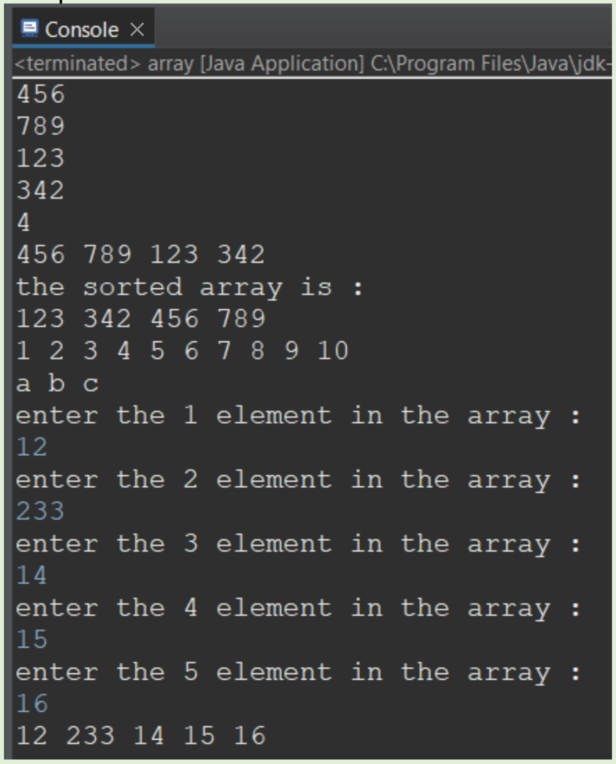


```
package basic_java;
import java.util.Arrays;
public class array {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        // array initialization
           int[] marks=new int[4];
           marks[0]=456;
           marks[1]=789;
           marks[2]=123;
           marks[3]=342;
          System.out.println(marks[0]);
          System.out.println(marks[1]);
          System.out.println(marks[2]);
          System.out.println(marks[3]);
          System.out.println(marks.length);
          for(int i=0;i<4;i++) {
              System.out.print(marks[i]+" ");
          }
          System.out.println();
```



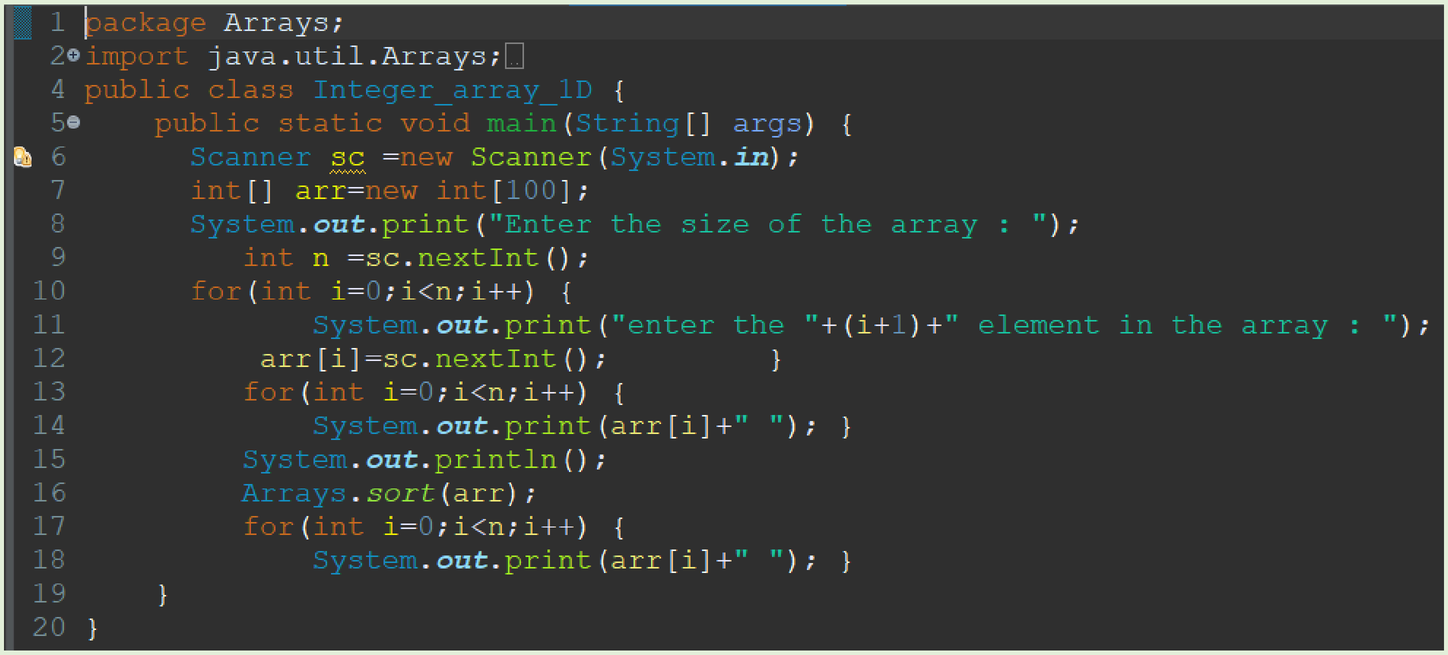
```
//           we can sort a list by using sort function
        Arrays.sort(marks);
        System.out.println("the sorted array is : ");
        for(int i=0;i<4;i++) {
          System.out.print(marks[i]+" ");
        }
        System.out.println();

        int[] number= {1,2,3,4,5,6,7,8,9,10};
        for(int i=0;i<10;i++) {
              System.out.print(number[i]+" ");
          }
        System.out.println();
//          character array
        char[] number2= {'a','b','c'};
        for(int i=0;i<3;i++) {
          System.out.print(number2[i]+" ");
        }
        System.out.println();
        // user given input 1-d array
        int[] n = new int[5];
        for(int i=0;i<5;i++) {
            System.out.println("enter the "+(i+1)+" element in the array :");
            n[i]=sc.nextInt();
        }
        for(int i=0;i<5;i++) {
              System.out.print(n[i]+" ");
          }
    }
}
```

Output:-



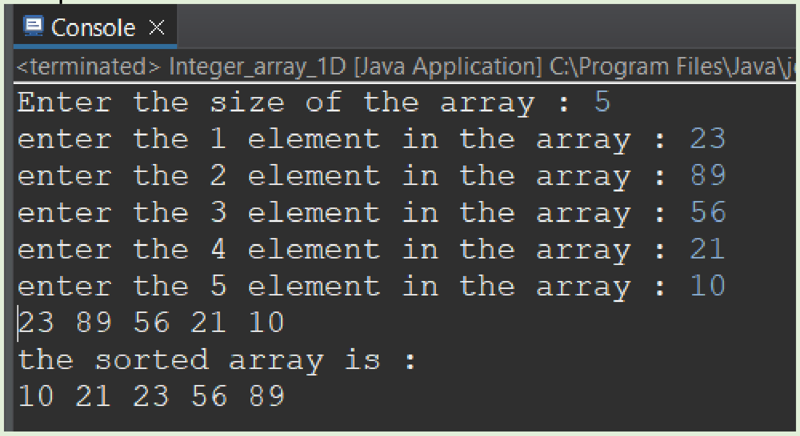
```
<terminated> array [Java Application] C:\Program Files\Java\jdk-
456
789
123
342
4
456 789 123 342
the sorted array is :
123 342 456 789
1 2 3 4 5 6 7 8 9 10
a b c
enter the 1 element in the array :
12
enter the 2 element in the array :
233
enter the 3 element in the array :
14
enter the 4 element in the array :
15
enter the 5 element in the array :
16
12 233 14 15 16
```

Ex:-



```
package Arrays;
import java.util.Arrays;
public class Integer_array_1D {
    public static void main(String[] args) {
      Scanner sc =new Scanner(System.in);
      int[] arr=new int[100];
      System.out.print("Enter the size of the array : ");
         int n =sc.nextInt();
      for(int i=0;i<n;i++) {
             System.out.print("enter the "+(i+1)+" element in the array : ");
         arr[i]=sc.nextInt();         }
         for(int i=0;i<n;i++) {
             System.out.print(arr[i]+" "); }
         System.out.println();
         Arrays.sort(arr);
         for(int i=0;i<n;i++) {
             System.out.print(arr[i]+" "); }
    }
}
```

Output:-



Console

<terminated> Integer_array_1D [Java Application] C:\Program Files\Java\j

```
Enter the size of the array : 5
enter the 1 element in the array : 23
enter the 2 element in the array : 89
enter the 3 element in the array : 56
enter the 4 element in the array : 21
enter the 5 element in the array : 10
23 89 56 21 10
the sorted array is :
10 21 23 56 89
```

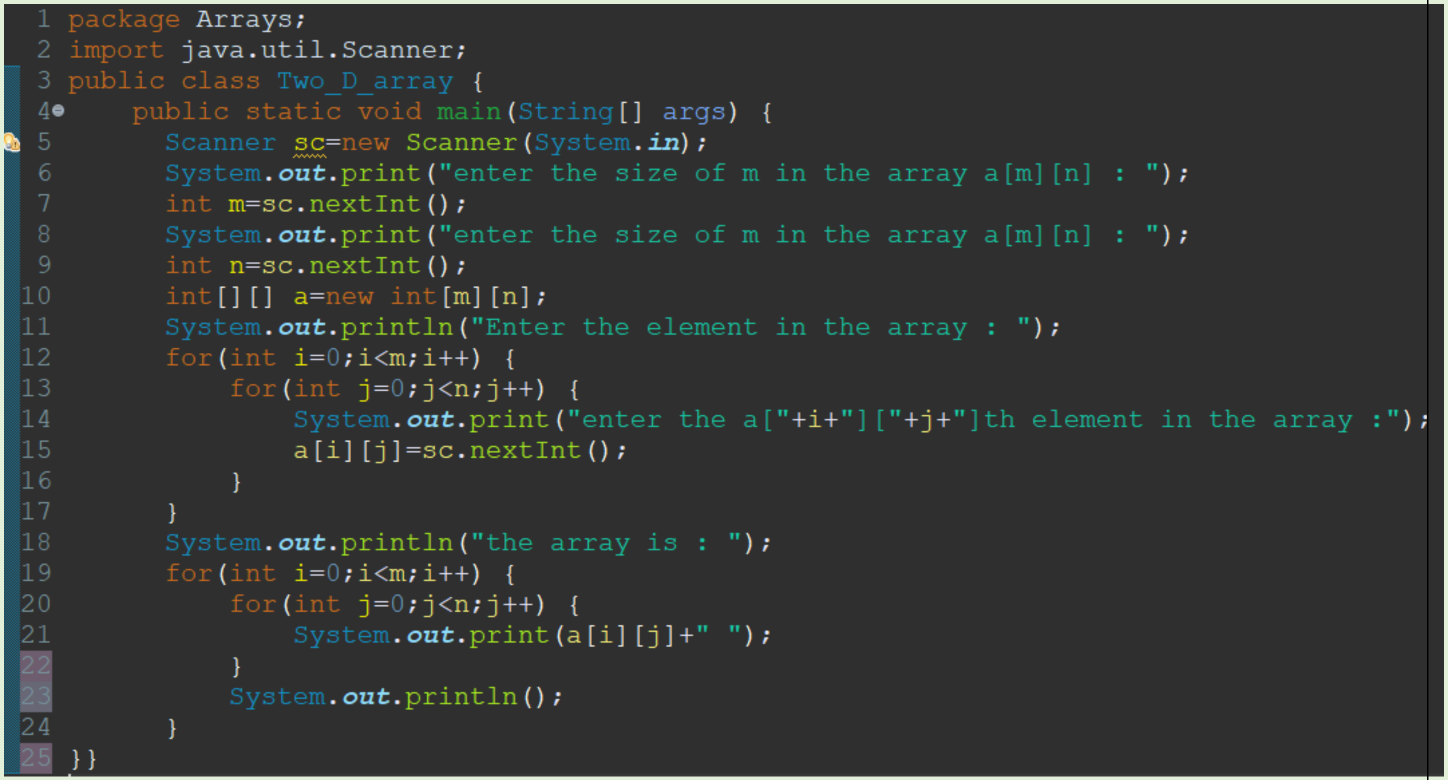
- ***Multidimensional Array:-***

Syntax to Declare Multidimensional Array in Java:-

1. dataType[][] arrayRefVar; (or)
2. dataType [][]arrayRefVar; (or)
3. dataType arrayRefVar[][]; (or)
4. dataType []arrayRefVar[];

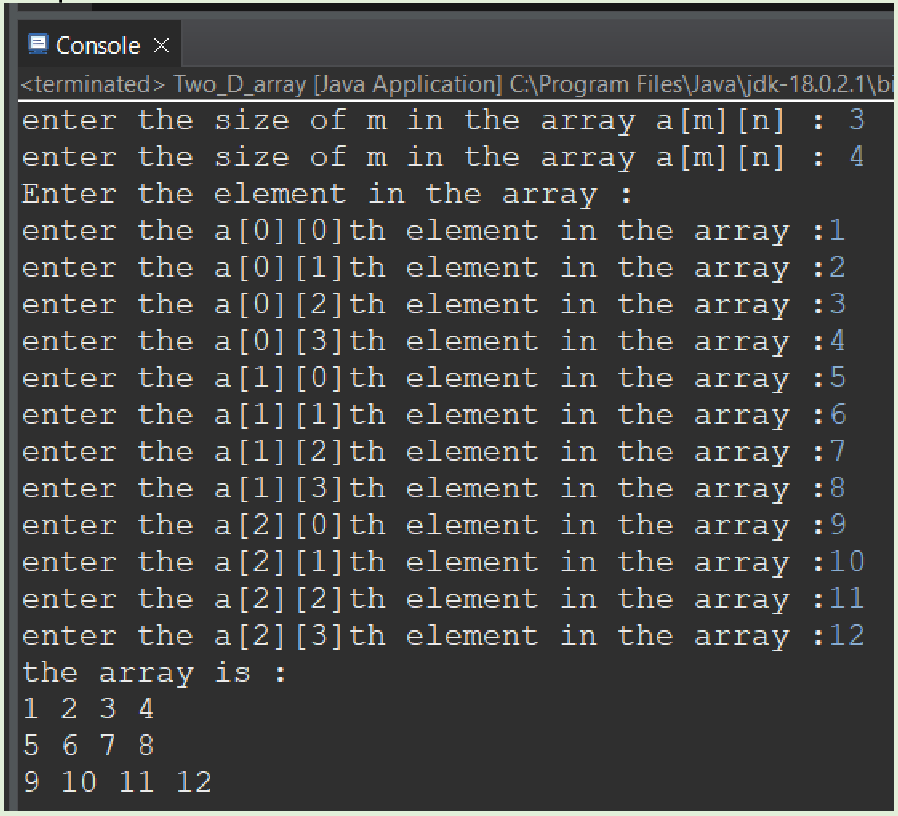
```
int[][] arr=new int[3][3];//3 row and 3 column .
```

code:-



```java
package Arrays;
import java.util.Scanner;
public class Two_D_array {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.print("enter the size of m in the array a[m][n] : ");
        int m=sc.nextInt();
        System.out.print("enter the size of m in the array a[m][n] : ");
        int n=sc.nextInt();
        int[][] a=new int[m][n];
        System.out.println("Enter the element in the array : ");
        for(int i=0;i<m;i++) {
            for(int j=0;j<n;j++) {
                System.out.print("enter the a["+i+"]["+j+"]th element in the array :");
                a[i][j]=sc.nextInt();
            }
        }
        System.out.println("the array is : ");
        for(int i=0;i<m;i++) {
            for(int j=0;j<n;j++) {
                System.out.print(a[i][j]+" ");
            }
            System.out.println();
        }
}}
```

Output:-



```
Console
<terminated> Two_D_array [Java Application] C:\Program Files\Java\jdk-18.0.2.1\bi
enter the size of m in the array a[m][n] : 3
enter the size of m in the array a[m][n] : 4
Enter the element in the array :
enter the a[0][0]th element in the array :1
enter the a[0][1]th element in the array :2
enter the a[0][2]th element in the array :3
enter the a[0][3]th element in the array :4
enter the a[1][0]th element in the array :5
enter the a[1][1]th element in the array :6
enter the a[1][2]th element in the array :7
enter the a[1][3]th element in the array :8
enter the a[2][0]th element in the array :9
enter the a[2][1]th element in the array :10
enter the a[2][2]th element in the array :11
enter the a[2][3]th element in the array :12
the array is :
1 2 3 4
5 6 7 8
9 10 11 12
```

- ***Math class:-***

Java Math class provides several methods to work on math calculations like min(), max(), avg(), sin(), cos(), tan(), round(), ceil(), floor(), abs() etc.

- ***Java Math Methods:-***

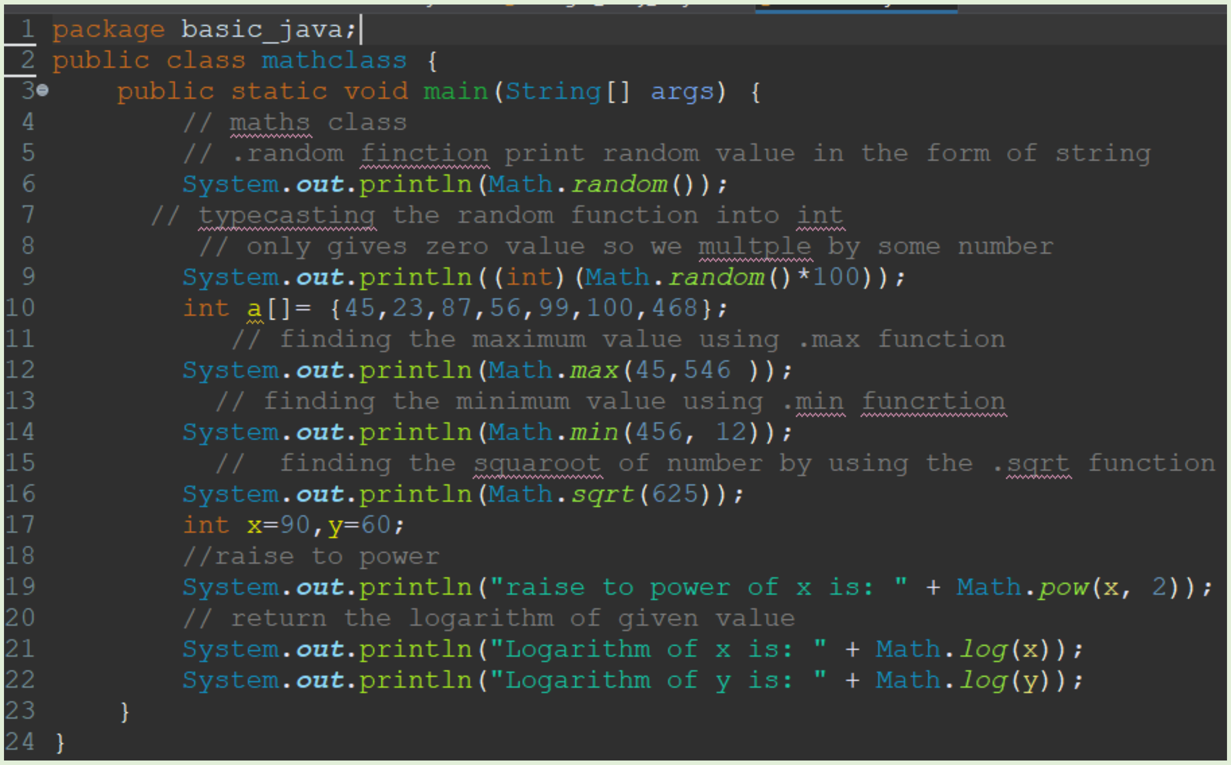
The **java.lang.Math** class contains various methods for performing basic numeric operations such as the logarithm, cube root, and trigonometric functions etc. The various java math methods are as follows:

Basic Math methods:-

Method	Description
Math.abs()	It will return the Absolute value of the given value.
Math.max()	It returns the Largest of two values.
Math.min()	It is used to return the Smallest of two values.
Math.round()	It is used to round of the decimal numbers to the nearest value.
Math.sqrt()	It is used to return the square root of a number.
Math.cbrt()	It is used to return the cube root of a number.
Math.pow()	It returns the value of first argument raised to the power to second argument.
Math.signum()	It is used to find the sign of a given value.
Math.ceil()	It is used to find the smallest integer value that is greater than or equal to the argument or mathematical integer.
Math.copySign()	It is used to find the Absolute value of first argument along with sign specified in second argument.
Math.nextAfter()	It is used to return the floating-point number adjacent to the first argument in the direction of the second argument.
Math.nextUp()	It returns the floating-point value adjacent to d in the direction of positive infinity.
Math.nextDown()	It returns the floating-point value adjacent to d in the direction of negative infinity.
Math.floor()	It is used to find the largest integer value which is less than or equal to the argument and is equal to the mathematical integer of a double value.
Math.floorDiv()	It is used to find the largest integer value that is less than or equal to the algebraic quotient.
Math.random()	It returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0.
Math.rint()	It returns the double value that is closest to the given argument and equal to mathematical integer.

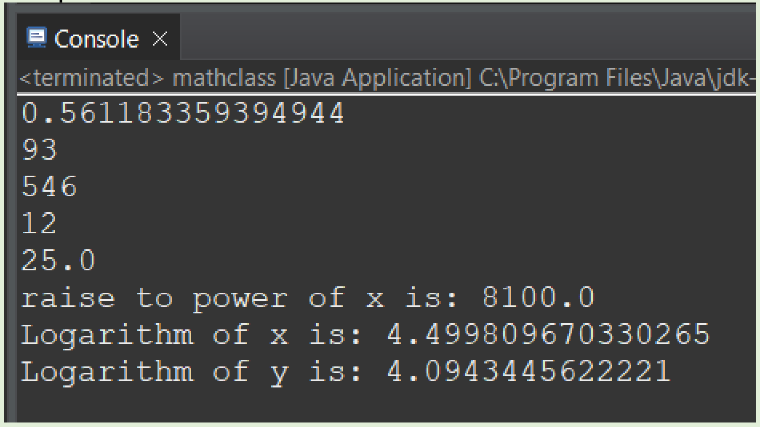
Math.hypot()	It returns sqrt(x^2 +y^2) without intermediate overflow or underflow.
Math.ulp()	It returns the size of an ulp of the argument.
Math.getExponent()	It is used to return the unbiased exponent used in the representation of a value.
Math.IEEEremainder()	It is used to calculate the remainder operation on two arguments as prescribed by the IEEE 754 standard and returns value.
Math.addExact()	It is used to return the sum of its arguments, throwing an exception if the result overflows an int or long.
Math.subtractExact()	It returns the difference of the arguments, throwing an exception if the result overflows an int.
Math.multiplyExact()	It is used to return the product of the arguments, throwing an exception if the result overflows an int or long.
Math.incrementExact()	It returns the argument incremented by one, throwing an exception if the result overflows an int.
Math.decrementExact()	It is used to return the argument decremented by one, throwing an exception if the result overflows an int or long.
Math.negateExact()	It is used to return the negation of the argument, throwing an exception if the result overflows an int or long.
Math.toIntExact()	It returns the value of the long argument, throwing an exception if the value overflows an int.

Code:-some of the method is :-



```
package basic_java;
public class mathclass {
    public static void main(String[] args) {
        // maths class
        // .random finction print random value in the form of string
        System.out.println(Math.random());
      // typecasting the random function into int
         // only gives zero value so we multple by some number
        System.out.println((int)(Math.random()*100));
        int a[]= {45,23,87,56,99,100,468};
           // finding the maximum value using .max function
        System.out.println(Math.max(45,546 ));
          // finding the minimum value using .min funcrtion
        System.out.println(Math.min(456, 12));
          //  finding the squaroot of number by using the .sqrt function
        System.out.println(Math.sqrt(625));
        int x=90,y=60;
        //raise to power
        System.out.println("raise to power of x is: " + Math.pow(x, 2));
        // return the logarithm of given value
        System.out.println("Logarithm of x is: " + Math.log(x));
        System.out.println("Logarithm of y is: " + Math.log(y));
    }
}
```

Output:-



```
<terminated> mathclass [Java Application] C:\Program Files\Java\jdk-
0.561183359394944
93
546
12
25.0
raise to power of x is: 8100.0
Logarithm of x is: 4.499809670330265
Logarithm of y is: 4.0943445622221
```

- ***String Class:-***

Strings are immutable non-primitive data types in Java. Once a string is created it's value cannot be changed i.e. if we wish to alter its value then a new string with a new value has to be created. This class in java has various important methods that can be used for Java objects. These include:

a. Concatenation(concat)

```
String name1 = new String("Aman");
String description = new String("is a good boy.");
String sentence = name1 + description;
System.out.println(sentence);
Sytem.out.println(name1.concat(description));
```

b. CharAt

```
String name = new String("Aman");
System.out.println(name.charAt(0))
```

c. Length

```
String name = new String("Aman");
System.out.println(name.length());
```

d. Replace

```
String name = new String("Aman");
System.out.println(name.replace('a', 'b'));
```

e. Substring

```
String name = new String("AmanAndAkku");
System.out.println(name.substring(0, 4));
```

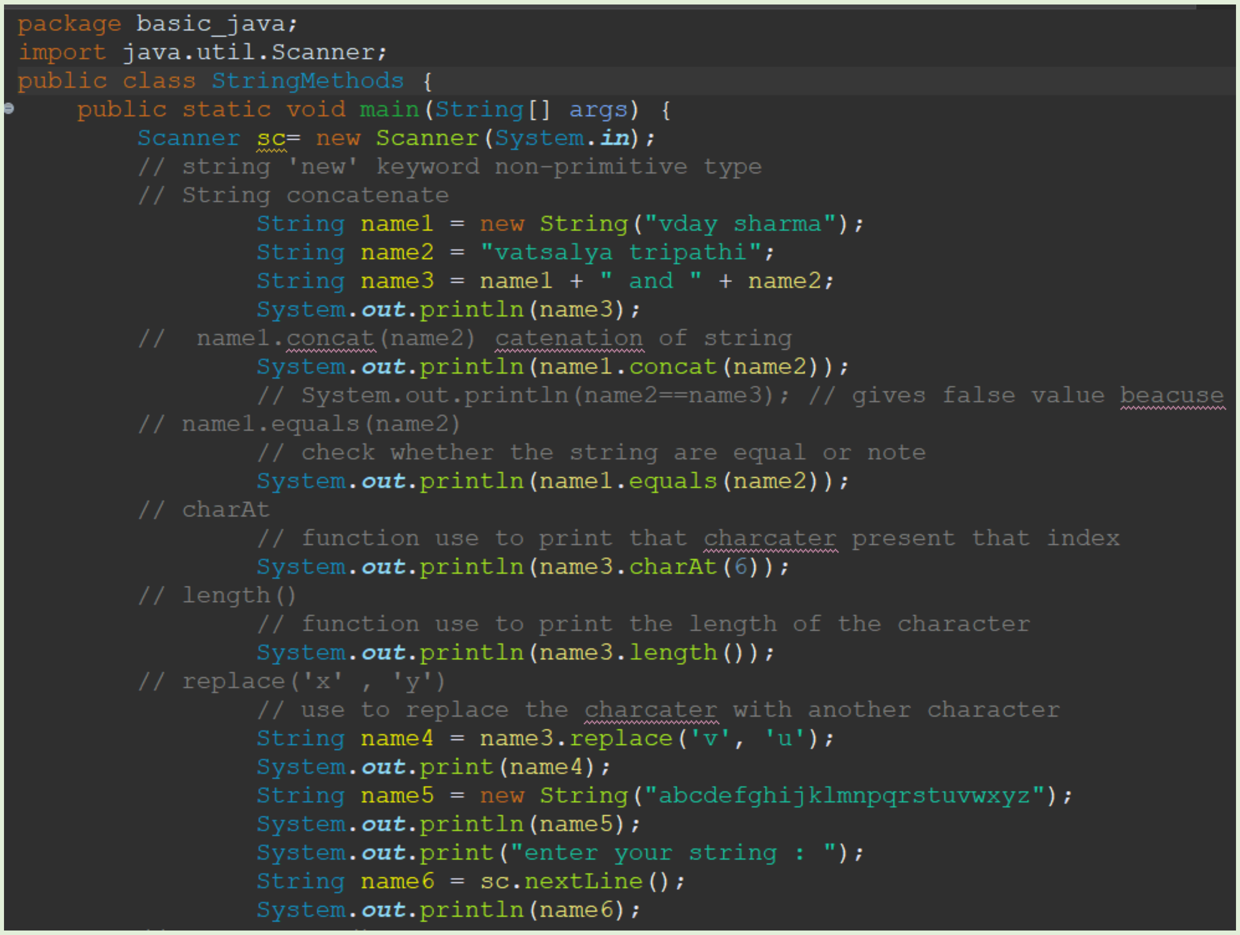
- ***Java String class methods:-***

The java.lang.String class provides many useful methods to perform operations on sequence of char values.

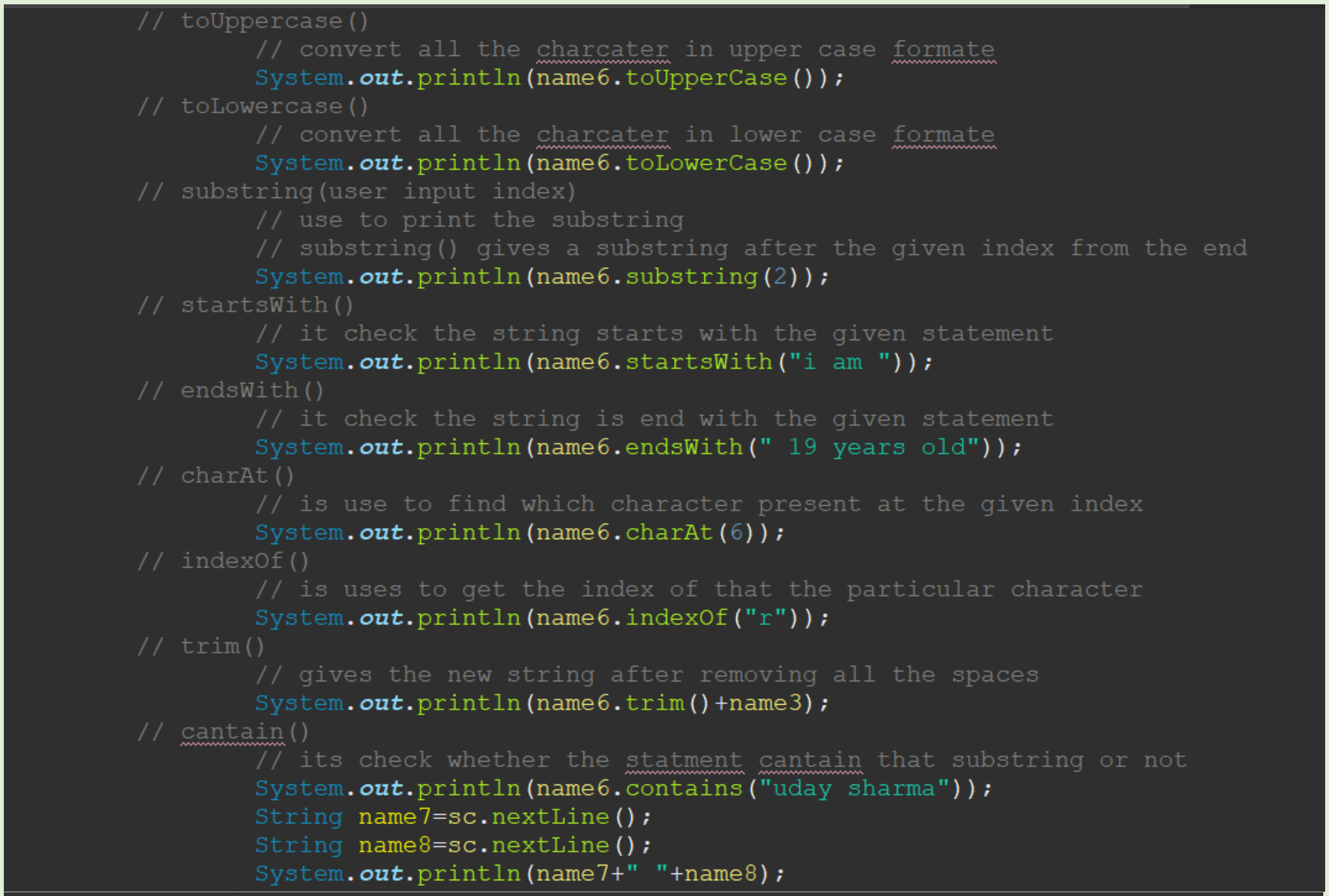
Method	Description
1. length()	Returns the length of String .
2. toLowerCase()	Converts all the characters of the string to the lower case letters.
3. toUpperCase()	Converts all the characters of the string to the upper case letters.
4. trim()	Returns a new String after removing all the leading and trailing spaces from the original string.

5. substring(int start)	Returns a substring from start to the end. Substring(3) returns “ substring”. [Note that indexing starts from 0 (default)]
6. substring(int start, int end)	Returns a substring from the start index to the end index. The start index is included, and the end is excluded.
7. replace(‘C1’, ‘C2’)	Returns a new string after replacing C 1with C2. (This method takes char as argument)
8. startsWith(“ string ”)	Its check the string is starts with the given statement or string .
9. endsWith(“string ”)	Its check the string is ends with the given statement or string .
10. charAt(2)	Returns the character at a given index position.
11. indexOf(“s”)	Returns the index of the first occurrence of the specified character in the given string.
12. lastIndexOf(“”)	Returns the last index of the specified character from the given string.
13. equals(string)	Returns true if the given string is equal to string false otherwise [Case sensitive]
14.equalsIgnoreCase(string)	Returns true if two strings are equal, ignoring the [lower and upper case] of characters.

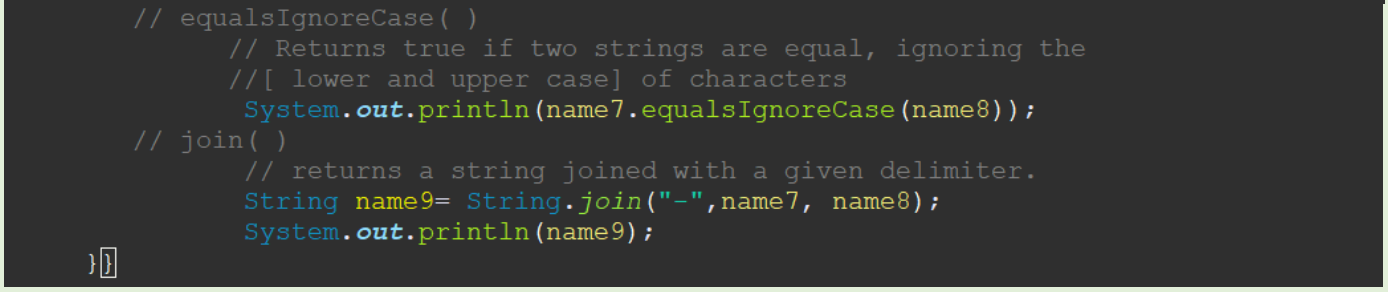
Code:-



```
package basic_java;
import java.util.Scanner;
public class StringMethods {
    public static void main(String[] args) {
        Scanner sc= new Scanner(System.in);
        // string 'new' keyword non-primitive type
        // String concatenate
                String name1 = new String("vday sharma");
                String name2 = "vatsalya tripathi";
                String name3 = name1 + " and " + name2;
                System.out.println(name3);
        //  name1.concat(name2) catenation of string
                System.out.println(name1.concat(name2));
                // System.out.println(name2==name3); // gives false value beacuse
        // name1.equals(name2)
                // check whether the string are equal or note
                System.out.println(name1.equals(name2));
        // charAt
                // function use to print that charcater present that index
                System.out.println(name3.charAt(6));
        // length()
                // function use to print the length of the character
                System.out.println(name3.length());
        // replace('x' , 'y')
                // use to replace the charcater with another character
                String name4 = name3.replace('v', 'u');
                System.out.print(name4);
                String name5 = new String("abcdefghijklmnpqrstuvwxyz");
                System.out.println(name5);
                System.out.print("enter your string : ");
                String name6 = sc.nextLine();
                System.out.println(name6);
```

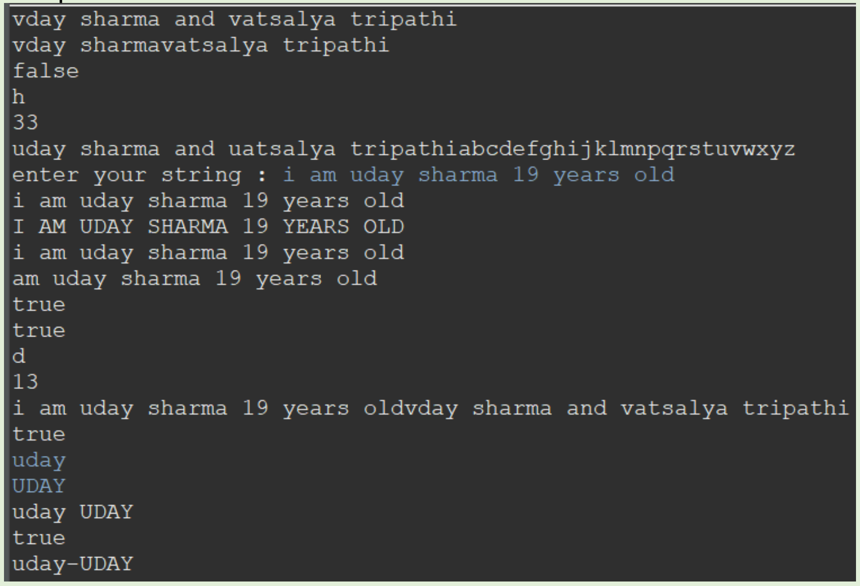


```
        // toUppercase()
                // convert all the charcater in upper case formate
                System.out.println(name6.toUpperCase());
        // toLowercase()
                // convert all the charcater in lower case formate
                System.out.println(name6.toLowerCase());
        // substring(user input index)
                // use to print the substring
                // substring() gives a substring after the given index from the end
                System.out.println(name6.substring(2));
        // startsWith()
                // it check the string starts with the given statement
                System.out.println(name6.startsWith("i am "));
        // endsWith()
                // it check the string is end with the given statement
                System.out.println(name6.endsWith(" 19 years old"));
        // charAt()
                // is use to find which character present at the given index
                System.out.println(name6.charAt(6));
        // indexOf()
                // is uses to get the index of that the particular character
                System.out.println(name6.indexOf("r"));
        // trim()
                // gives the new string after removing all the spaces
                System.out.println(name6.trim()+name3);
        // cantain()
                // its check whether the statment cantain that substring or not
                System.out.println(name6.contains("uday sharma"));
                String name7=sc.nextLine();
                String name8=sc.nextLine();
                System.out.println(name7+" "+name8);
```



```
        // equalsIgnoreCase( )
              // Returns true if two strings are equal, ignoring the
              //[ lower and upper case] of characters
               System.out.println(name7.equalsIgnoreCase(name8));
        // join( )
               // returns a string joined with a given delimiter.
               String name9= String.join("-",name7, name8);
               System.out.println(name9);
    }}
```

Output:-



```
vday sharma and vatsalya tripathi
vday sharmavatsalya tripathi
false
h
33
uday sharma and uatsalya tripathiabcdefghijklmnpqrstuvwxyz
enter your string : i am uday sharma 19 years old
i am uday sharma 19 years old
I AM UDAY SHARMA 19 YEARS OLD
i am uday sharma 19 years old
am uday sharma 19 years old
true
true
d
13
i am uday sharma 19 years oldvday sharma and vatsalya tripathi
true
uday
UDAY
uday UDAY
true
uday-UDAY
```

- Go check out my ***LinkedIn profile*** for more notes and other resources content

@Uday Sharma

mrudaysharma4600@gmail.com

https://www.linkedin.com/in/uday-sharma-602b33267