

# Java Introduction.

09 November 2022 10:09

## What is software?

Software is a solution to solve real world problems using an operating system.

## What is a programming language ?

Programming language is a language which is used to communicate with a machine .

## What can a computer understand?

Binary language (ie ones and zeros 1,0)

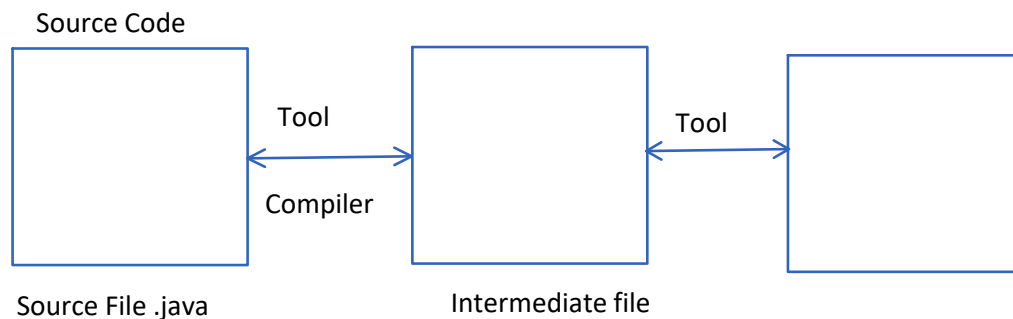
---

## Stages of Programming Languages

There are three types of programming Languages

1. **Low Level Programming Language** ( Language understandable by computer/machine)
2. **Mid-Level Programming Language** ( Language neither understandable by a person or a machine)
3. **High Level Programming Language** ( Language understandable by a person)

Source Code/Source file



**Source File** : Programmer written file

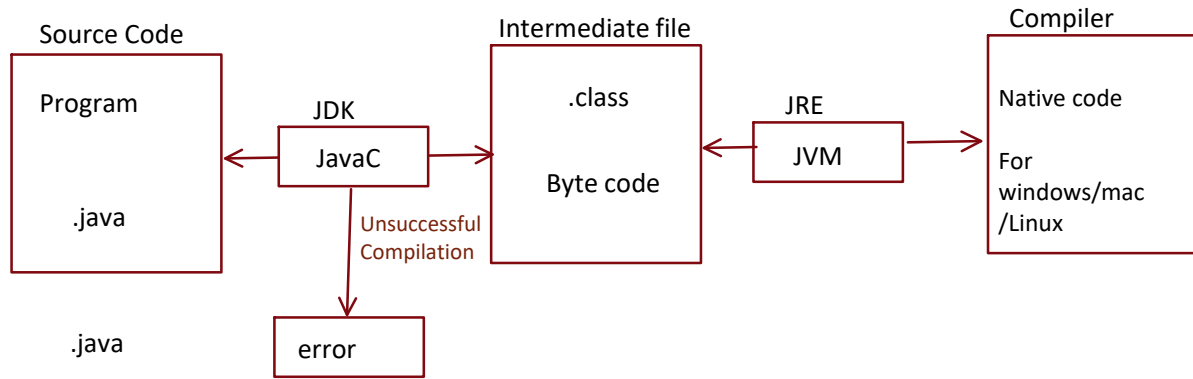
**Intermediate File** : The file generated by a compiler is called as intermediate File

## Q. Why do we use Java?

1. Java is simpler compared to other programming languages.
2. Java is object oriented programming language.
3. In Java memory allocation is automatic.
4. Java is platform independent.
5. Java is secured and fast.
6. Java is strictly typed language. ( Java is datatype conscious )
7. Rich inbuilt resources.

# Java Compilation and execution procedure

11 November 2022 01:43 PM



Java program is written in a **.java** file. This file has to be compiled by java compiler(javaC), which is available inside JDK. This Java compiler is going to throw an error if the program is not written correctly.

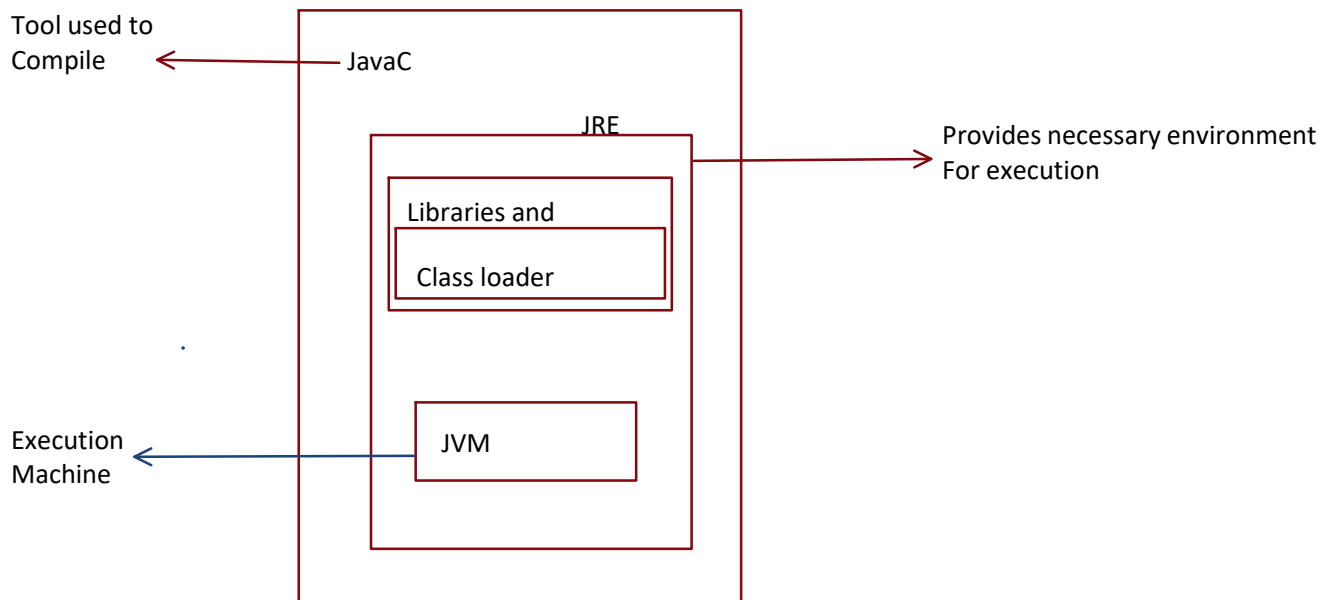
- a. In the case of successful compilation an intermediate file is created with .class extension .. This **.class** is an executable file. This executable file can be executed in a computer. Only if JRE is present.

## JDK( Java Development Kit ) :

This is a kit provided by java for developers to use a tool called javaC . JavaC is java Compiler. JDK enables a programmer to develop and also execute a program.

## JRE( Java Run Environment ) :

JRE is a run time environment which provides all the necessary tools and libraries at the time of execution.



**Path** : it is location / address/ or a file . Path tells you where file is present

Important terminologies of path

1. **Route folder / Source Folder**
2. **Current working folder**

### 3. Destination Folder

**Basic CMD commands:**

**CLS** : clear screen

**CD ..** : change Directory : used to go back a directory/folder

**CD ..\..\..\** : Used to go back multiple folders

**Mkdir** : short for Make Directory . It is used to create a folder

**Absolute Path** : The path to the destination folder from root folder is called as absolute path.

**Relative Path**: The path to the destination folder from the current working folder is called as Relative path

**Token**: Token is a smallest unit of a program which is meaningful to the compiler.

Tokens are categorized in five types:

1. **Keywords**
2. **Identifiers**
3. **Operators**
4. **Literals**
5. **Separators**

1. **Keyword** : Keyword is a reserved word has predefined meaning. There are around 53 keywords in java. All the keywords are in lower case.

- |                       |                         |
|-----------------------|-------------------------|
| 1. <b>Abstract</b>    | 26. <b>Instanceof</b>   |
| 2. <b>Boolean</b>     | 27. <b>Int</b>          |
| 3. <b>Break</b>       | 28. <b>Interface</b>    |
| 4. <b>Byte</b>        | 29. <b>Long</b>         |
| 5. <b>Case</b>        | 30. <b>Native</b>       |
| 6. <b>Catch</b>       | 31. <b>New</b>          |
| 7. <b>Char</b>        | 32. <b>Null</b>         |
| 8. <b>Class</b>       | 33. <b>Package</b>      |
| 9. <b>Const</b>       | 34. <b>Private</b>      |
| 10. <b>Continue</b>   | 35. <b>Protected</b>    |
| 11. <b>Default</b>    | 36. <b>Public</b>       |
| 12. <b>Do</b>         | 37. <b>Return</b>       |
| 13. <b>Double</b>     | 38. <b>Short</b>        |
| 14. <b>Else</b>       | 39. <b>Static</b>       |
| 15. <b>Extends</b>    | 40. <b>Strictfp</b>     |
| 16. <b>False</b>      | 41. <b>Super</b>        |
| 17. <b>Final</b>      | 42. <b>Switch</b>       |
| 18. <b>Finally</b>    | 43. <b>Synchronized</b> |
| 19. <b>Float</b>      | 44. <b>This</b>         |
| 20. <b>For</b>        | 45. <b>Throw</b>        |
| 21. <b>Goto</b>       | 46. <b>Throws</b>       |
| 22. <b>Assert</b>     | 47. <b>Transient</b>    |
| 23. <b>If</b>         | 48. <b>True</b>         |
| 24. <b>Implements</b> | 49. <b>Try</b>          |
| 25. <b>Import</b>     | 50. <b>Void</b>         |
|                       | 51. <b>Volatile</b>     |
|                       | 52. <b>While</b>        |

Keywords for data types are:

**boolean byte char int long short float double**

Keywords for access control are:

**privateprotected public**

Keywords for modifiers are:

**final static synchronized volatile transient native**

Keywords for modifiers are:

**abstract      final      native   private   protected      public**  
**static   transient      synchronized   volatile      strictfp**

Keywords for catch-exception are:

**try      catch      finally      throw**

Keywords for loops or decision-makers are:

**break   case   continue      default   do      while   for**  
**switch   if      else**

Keywords for class functions are:

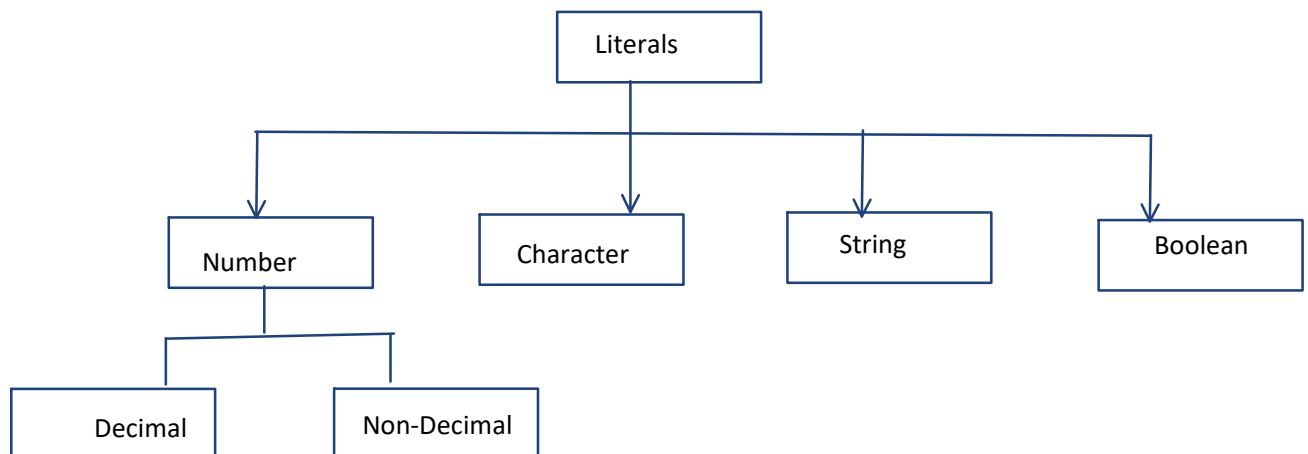
**class   extends      implements   import   instanceof**  
**new   package      return   interface**  
**this   throws   void      super**

Keywords for assigned values are:

**true   false   null**

Outdated keywords are:

**const   goto**



**5. Separators :** Separators are used to separate a block of code from one another.

Some of these separators are **{ } ( ) [ ] . , ;**

Printing statement:

1. **Identifiers :** It is a name given to identify a particular memory location .  
Identifier is a name given to a variable, method , class ,interface etc.  
Identifier is a name given by a user or programmer.
2. **Operators :** Operators are symbols to perform something.  
Eg. Addition
3. **Literals :** Literals mean data or values.

# Java Structure of a Program:

15 November 2022 01:00 PM

## Structure of a program :

Rules for identifiers :

1. You cannot start with a number.

12Abc ✗

A12BC ✓

2. You cannot use a keyword as an identifier.
3. You cannot use any special character while writing an identifier except for **\$ \_** (dollar & underscore)
4. You can use these two characters anywhere in the name

*Note: Space is a special character*

*Note :For a Class first letter of each word should be upper-case*

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

*Note: Class is also called as byte code.*

# Java Variable , Datatypes

16 November 2022 01:11 PM

**Variable** is a container to store data

Variable is a named memory location to store location.

Every container is determined to store a particular data.

## DataTypes :

- > Datatype is nothing but the type of the data.
- > Datatypes are classified into 2 types

- **Primitive**

1 byte = 8 bit range : -  $2^{(n-1)}$  to  $2^{(n-1)}-1$

Byte	1 byte	↑ To store non-decimal numbers ↓
Short	2 byte	
Int	4 byte	
Long	8 byte	
Float	4 byte	↑ To store Decimal Numbers ↓
Double	8 byte	
Char	2 byte	← To store Character Literals
Boolean	1 bit	← True/False

*Syntax to create a variable :*

Datatype Varname ; //declaration

Ex. `byte b;`  
`short s;`  
`char c;`

- **Non-Primitive**

Declare two variables assign some values and calculate the average of them

```
Short abc ; //declaration ;  
Abc = 120; // initialization ;  
Short abc = 120 , xyz = 340 ,pqr = 400 ; //declaration & initialization of multiple variables
```

- 
1. WAP to perform all the arithmetic operations with two variables.
  2. WAP to print the multiplication table for a value in a variable

**note :** The default type for non-decimal number is called is **int** type.  
The default type of any decimal number is **double** .

```
long + int = long  
long + short = long  
byte + long = long
```

int + int = int  
sort + int = int  
short + short = int  
byte + byte = int  
byte + short = int  
byte + int = int  
byte + double = double  
int + double = double  
float + double = double

**ASCII Values :**

'0' - '9' -> 48 - 57

'Q' - '3' -> 97 - 122

'A' - 'Z' -> 65 - 90

SPACE -> 32

; -> 59

ENTER - > 10

# Java Typecasting

17 November 2022 01:57 PM

**Typecasting :** Converting one type of data into another type of data is known as typecasting .

There are two types of typecasting :

1. **Primitive typecasting .**
2. **Non-Primitive typecasting .**

## **Primitive Typecasting :**

In Primitive Typecasting we convert one primitive type of data into another primitive type of data.

It is further divided into two types :

### **a. Widening :**

- Converting a smaller type of data into a larger type of data is called widening.
- Widening is implicit in nature

### **b. Narrowing**

- Converting a larger type of data into into smaller type of data is called as narrowing.
- It is explicit in nature .
- There's always is data loss.

**Non-Primitive Typecasting:** Converting one type of non-primitive data into another type of non-primitive data :

It Has 2 types :

1. **Up-casting**
2. **Down-casting**

**Narrowing in depth :** if we store a larger value in a shorter datatype like from short to a byte it will

**Short n = 130;** Cycle through negative values if we push from positive side and vice versa

**Byte b = (byte)n;**

**System.out.print(b);**

**Output -125**

**Short n = -129;**

**Byte b = (byte)n;**

**System.out.print(b);**

**Output 127**



# Types of Variables :

19 November 2022 07:47 AM

Based on visibility , variables can be classified into two types

1. **Local Variable**
2. **Global Variable**

## Global Variable :

A variable which is **declared inside class block** is known as Global Variable.

Note : Default values exists only for global variables . ( Strings, Non-Primitive, Arrays etc )

Global Variables are of two types :

1. **Static Variable** ex : static int a ;
2. **Non-static Variable**

### 1. Static Variable :

Static int a ;                      var a | 0  
Static float b ;                    var b | 0.0

## Local Variable :

A Variable which is **declared inside method block** or any other block other than class block is Known as Local Variable

Or

A variable which is **declared inside Local area** is known as local variable.

**Note** : there's no default value for local variable. Just declaring local variable will not throw an error.

error is when you try to use a local variable without initializing it.

A local variable cannot be used unless it is initialized .

A variable declared in a block can be used only within that particular block.

**Variable Shadowing** : When there are two variables of the same name : one in global scope and another one in local scope,

Local variable will be dominant this process is called as variable Shadowing .

```
1 package com.TestPackage;
2
3 public class Test {
4
5     public static void main(String[] args) {
6         int n = 13;
7         if (n < 0) {
8             int a = 10;
9             System.out.println(a);
10        }
11        System.out.println(a); // CTE
12        for(int i = 0 ; i <= 2 ; i++) {
13            System.out.println("okay");
14        }
15        System.out.println(i); // CTE
16        System.out.println(n);
17    }
18 }
19
20 }
```

## Characteristics of Local Variable :

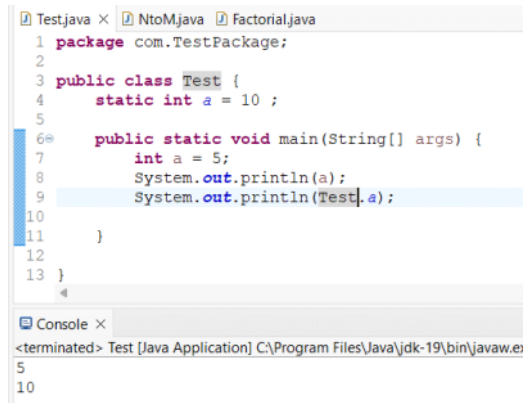
1. Local variables will not be assigned with default values .
2. Local variable must be initialized before using .
3. It can be used only within the block where it is declared.

Class P1

```

{
Public static void main(String[] args)
Int a ;
System.out.println(a); //cte
}
}

```



The screenshot shows an IDE with two tabs: 'Test.java' and 'Factorial.java'. The 'Test.java' tab is active, displaying the following code:

```

1 package com.TestPackage;
2
3 public class Test {
4     static int a = 10 ;
5
6     public static void main(String[] args) {
7         int a = 5;
8         System.out.println(a);
9         System.out.println(Test.a);
10    }
11 }
12
13 }

```

Below the code editor is a console window titled 'Console'. It shows the output of the program:

```

<terminated> Test [Java Application] C:\Program Files\Java\jdk-19\bin\javaw.e
5
10

```

// the variable declared In the local scope is used first , unless the variable created in the global scope is implicitly called .

// by using class name we can differentiate between global variable and local variable with the same name.

# Operators

**Operators** : An operators is a pre-defined symbol which is used to perform a specific task.

We can classify into 3types.

1. **Unary Operators**
2. **Binary Operators**
3. **Ternary Operators**

1. **Unary Operator :**

An Operator which can accept only one operand is known as unary operator.

Ex : increment/decrement , Not Operator , Typecasting Operator.

2. **Binary Operator :** Binary operators takes two operands and operates on them to produce as result.

3. **Ternary :** Ternary operator is a shorthand version of the if-else statement. It has three operands and hence the name ternary.

Ex.

Condition ? If true : if false

## Classification of Operator :

The Operators can also be classified based on task:

1. **Arithmetic Operator.**
2. **Assignment Operator.**
3. **Relational Operator.**
4. **Logical Operator .**
5. **Increment / Decrement Operator.**
6. **Conditional Operator.**
7. **Miscellaneous Operator.**

1. **Arithmetic Operator :**

In Java we have 5 types of Arithmetic Operators, they are :

`+, -, *, /, %`

**'+' :**

It has two behaviours in Java, they are :

- a. **Normal addition**
- b. **Concatenation**

Examples:

```
System.out.println(10+20+'B');//96
System.out.println(10+'a'+20);//127
System.out.println(10+"A");//10A
System.out.println(10+"A"+20);//10A10
System.out.println("A"+10+20);//A1020
System.out.println(10+20+"A");//30A
```

Examples :

```
System.out.println(10+20); // 30
System.out.println(10+20.5); //30.5
System.out.println(10+'a'); // 107 the character 'a' which has ascii value of 97 is added with int 10
System.out.println(10+"sheela"); //10sheela
System.out.println("sheela"+10); //sheela10
System.out.println(10+true); ct error
System.out.println("10"+true); // 10true
```

- Q. Sheela has 500 rs with her. She wants to distribute chocolates to 10 students . The cost of each chocolate is Rs 25 Calculate the number of chocolates she will get and calculate how many chocolates each student gets.
- Q. WAJ program to display a frog with a name Dinga needs to travel from the location A to B too meet his girlfriend Dingi ,the distance between location loc A and loc B is 100m ,Dinga can cover a distance of 5 meters in a single hop. Calculate the total number of hops required for Dinga to reach Dingi.
- Q. The capacity of a the pitcher is 2.2 liters assume it is half filled, we need to empty the pitcher by transferring the liquid into the mug which is having 150ml of capacity .  
WAJP to calculate the number of mugs required to empty the entire pitcher

#### Assignment Operators :

=  
+=  
-=  
\*=  
/=   
%=

```
Int a = 10;
a+=10; // 20
a-=10 ; // 10
a*=10; // 100
a/=10; //10
a%=10; // 0
```

---

#### Relational Operators :

Operators	Function
==	Equals to
!=	Not Equals to
>	Greater Than
<	Lesser Than
>=	Greater than or equals to
<=	Less than or equals to

It is used to verify the condition is true or false, just to compare is it correct or not.  
And the result is always boolean ,ie. True or False.

---

#### Logical Operators :

**Note: the returning value of Logical Operators is Boolean.**  
**And the conditions should return Boolean.**

#### AND Operator (&&)

Logical AND

If all the conditions return true only then the AND (&&) Operator returns true.

C1	C2	R
F	F	F
F	T	F
T	F	F
T	T	T

### OR Operator (||)

Logical OR

If **any one of the condition returns true** then the OR(||) Operator returns true.

C1	C2	R
F	F	F
F	T	T
T	F	T
T	T	T

### NOT Operator (!)

Logical NOT

If the input is true then the output value will be false and vice versa.

### Conditional Operator :

It is a ternary Operator .

Syntax:

**Operand1 ? Operand2 : Operand3**

**Condition ? Statement1 : statement2**

- The return type of Operand 1 must be Boolean.
- If condition is true, statement1 will get executed else statement2 get executed .

**Q. WAJP to check whether the given number is even or odd.**

```

1 class QEvenOdd
2 {
3     public static void main(String[] args)
4     {
5         int num = 10;
6         System.out.println( num % 2 == 0 ? num + " Is an Even number" : num + " Is an Odd number");
7     }
8 }
9

```

**Q. WAJP to check whether the given number is divisible by six**

```

1 class Div6
2 {
3     public static void main(String[] args)
4     {
5         int num = 36;
6         System.out.println(num % 6 == 0 ? num + " Is Divisible by 6" : num + " Is not Divisible By 6");
7     }
8 }
9

```

```

7     }
8 }
9

```

**Q. WAJP to find the greatest among three number . 10 20 420**

```

1 class Big3
2 {
3     public static void main(String[] args)
4     {
5         int num1 = 10;
6         int num2 = 20;
7         int num3 = 420;
8         int big1 = num1>num2 ? num1 : num2 ; // store the bigger number between num1 and num2 in big1
9         int big2 = big1>num3 ? big1 : num3 ; // store the bigger number between big1 and num3 in big2
10
11         System.out.println( big2 +" Is the Largest nubmber amongst " + num1 +" " + num2 + " " + num3);
12     }
13 }

```

```

1 class Big3
2 {
3     public static void main(String[] args)
4     {
5         int num1 = 10;
6         int num2 = 20;
7         int num3 = 420;
8         int big = (num1>num2) ? (num1>num3 ? num1 : num3) : (num2>num3 ? num2 : num3) ;
9         System.out.println(big +" Is the Largest number amongst " + num1 +" " + num2 + " " + num3);
10

```

Increment/Decrement Operator :

Increment Operator (++) :

There are two types of increment operators :

1. Pre-increment Operator
2. Post-increment Operator

Pre Increment Operator( ++i )

- Increase the value by 1 and update.
- Substitute the updated value.
- Use the updated value.

Post Increment ( i++ )

- Substitute the value.
- Increase the value by 1 and update.
- Use the updated value

Int a = 10;

System.out.println(a); //10

++a;

System.out.pritntln(a); //11

Int b = 10

System.out.println(b) ; //10

b++;

System.out.println(b) ; //11

System.out.println(b++); //11

System.out.println(b); //12/

Assignment Questions :

Q. Write difference between pre-increment and post increment Operator.

Q. What should be the type of operands for a logical operator?

Boolean Type

# Control Statements :

26 November 2022 13:11

## Control Statements :

Control statements are used to control the flow of execution. They are classified into two types.

## Control Statements :

1. Decision Making Statements
2. Looping Statements

### Decision Making Statements:

- Decision Making statements are executed on the basis of the condition.
- If the condition is true , the block is executed, else the block is skipped.
- Decision Making statements are of four types :
  - Simple - if
  - If else
  - Else if ladder
  - Switch

Simple - if :

```
Syntax - if (condition)
{
    // Statement
}
```

**Working :** if the condition is true , if is executed ,if the condition is false the block is skipped.

Example:

```
1 class DecisionsMaking
2 {
3     public static void main(String[] args)
4     {
5         System.out.println("main method started");
6         int n = 12;
7         System.out.println(n);
8         if(n%6==0){
9             System.out.println("If block Started");
10        }
11        System.out.println("main method ended");
12    }
13 }
14
```

```
samwa@Sami MINGW64 ~/OneDrive/
$ java DecisionsMaking
main method started
12
If block Started
main method ended
```

```
1 class DecisionsMaking
2 {
3     public static void main(String[] args)
4     {
5         System.out.println("main method started");
6         int n = 13;
7         System.out.println(n);
8         if(n%6==0){
9             System.out.println("If block Started");
10        }
11        System.out.println("main method ended");
12    }
13 }
14
```

```
samwa@Sami MINGW64 ~/OneDr
$ java DecisionsMaking
main method started
13
main method ended
```

Write a program to find a number is divisible by 3 or not.



```

1 class DecisionsMaking
2 {
3     public static void main(String[] args)
4     {
5         int n = 12;
6         if (n%3==0){
7             System.out.println(n + " is Divisible by 3");
8         }
9         if (n%3!=0){
10            System.out.println(n + " is not Divisible by 3");
11        }
12    }
13 }
14

```

```

samwa@Sami MINGW64 ~/OneDrive
$ java DecisionsMaking
12 is Divisible by 3

```

```

1 class DecisionsMaking
2 {
3     public static void main(String[] args)
4     {
5         int n = 13;
6         if (n%3==0){
7             System.out.println(n + " is Divisible by 3");
8         }
9         if (n%3!=0){
10            System.out.println(n + " is not Divisible by 3");
11        }
12    }
13 }
14

```

```

samwa@Sami MINGW64 ~/OneDrive
$ java DecisionsMaking
main method started
13 is not Divisible by 3

```

## If else

Syntax :

```

if( condition)
{
    // Statement
} else {
    // Statement
}

```

**Working :** first if the **if** condition is true the **if** block is executed if not it is skipped and the **else block** executes

# Assignment if, if-else

27 November 2022 20:13

## Assignment Questions :

### 1. WAP to find whether a number is positive or negative

```
1 class PosNeg
2 {
3     public static void main(String[] args)
4     {
5         int num = -3;
6         if (num > 0)
7         {
8             System.out.println( num + " Is a Positive Number");
9         }
10        if (num < 0)
11        {
12            System.out.println( num + " Is a Negative Number");
13        }
14        if (num == 0)
15        {
16            System.out.println( num + " Is a neither Positive or Negative");
17        }
18    }
19 }
20 }
```

```
samwa@Sami MINGW64 ~/OneDrive/desk
$ java PosNeg
-3 Is a Negative Number
```

For num = 3;

```
samwa@Sami MINGW64 ~/OneDrive/desk
$ java PosNeg
3 Is a Positive Number
```

For num = 0;

```
samwa@Sami MINGW64 ~/OneDrive/desktop
$ java PosNeg
0 Is a neither Positive or Negative
```

### 2. WAP to print the absolute value of a number

```
1 class AbsNum
2 {
3     public static void main(String[] args)
4     {
5         int num = -9 ;
6
7         System.out.println("Given Number : " + num);
8
9         if(num < 0){
10            System.out.println("Absolute Num: " + -num);
11        }else{
12            System.out.println("Absolute Num: " + num);
13        }
14    }
15 }
16 }
```

```
samwa@Sami MINGW64 ~/OneDrive/desktop
$ java AbsNum
Given Number : -9
Absolute Num: 9
```

```
samwa@Sami MINGW64 ~/OneDrive/desktop
$ java AbsNum
Given Number : 9
Absolute Num: 9
```

Q.2

```
int n = 6;
if (n < 0)
{
    n = (n * -1);
}
Sopln(n);
```

### 3. WAP a program to print whether a number is divisible by both 3 and 5

```
1 class Div5n3
2 {
3     public static void main(String[] args)
4     {
5         int num = 15 ;
6
7         if (num%5 == 0 && num%3 == 0)
8         {
9             System.out.println( num + " Is divisible by both 5 & 3");
10        }else{
11            System.out.println( num + " Is not divisible by both 5 & 3");
12        }
13    }
14 }
15 }
```

```
samwa@Sami MINGW64 ~/OneDrive/desktop
$ java Div5n3
15 Is divisible by both 5 & 3
```

For num = 12

```
samwa@Sami MINGW64 ~/OneDrive/desktop
$ java Div5n3
12 Is not divisible by both 5 & 3
```

### 4. WAP to find whether a number is single digit or not . Note that number can be either positive or negative.

```
1 class OneDig
2 {
3     public static void main(String[] args)
4     {
5         int num = 9;
6
7         if (num > -10 && num < 10)
8         {
```

```
samwa@Sami MINGW64 ~/OneDrive/desktop
$ java OneDig
9 is a Single Digit Number
```

For num = -8 ;

```
samwa@Sami MINGW64 ~/OneDrive/desktop
$ java OneDig
-8 is a Single Digit Number
```

```

4  {
5      int num = 9;
6
7      if (num > -10 && num < 10)
8      {
9          System.out.println(num + " is a Single Digit Number");
10     }else{
11         System.out.println(num + " is not a Single Digit Number");
12     }
13 }
14 }
15

```

For num = -8 ;

```

samwa@Sami MINGW64 ~/OneDrive/desk
$ java OneDig
-8 is a Single Digit Number

```

For num = 10;

```

samwa@Sami MINGW64 ~/OneDrive/desk
$ java OneDig
10 is not a Single Digit Number

```

For num = -10;

```

samwa@Sami MINGW64 ~/OneDrive/desk
$ java OneDig
-10 is not a Single Digit Number

```

5. WAP to find whether a number is two digit number or not. Note that number can be positive or negative.

```

1  class TwoDig
2  {
3      public static void main(String[] args)
4      {
5          int num = 69;
6
7          if ((num > -100 && num < -9 ) || (num < 100 && num > 9) )
8          {
9              System.out.println(num + " is a Two Digit Number");
10         }else{
11             System.out.println(num + " is not a Two Digit Number");
12         }
13     }
14 }
15

```

```

samwa@Sami MINGW64 ~/OneDrive/desk
$ java TwoDig
69 is a Two Digit Number

```

For different values of num :

```

samwa@Sami MINGW64 ~/OneDrive/desk
$ java TwoDig
9 is not a Two Digit Number

```

```

samwa@Sami MINGW64 ~/OneDrive/desk
$ java TwoDig
-80 is a Two Digit Number

```

```

samwa@Sami MINGW64 ~/OneDrive/desk
$ java TwoDig
-100 is not a Two Digit Number

```

6. WAP to find whether a number is three digit number or not .The number can be positive or not.

```

1  class ThreeDig
2  {
3      public static void main(String[] args)
4      {
5          int num = 420;
6
7          if (( num > -1000 && num < -99 ) || (num < 1000 && num > 99))
8          {
9              System.out.println(num + " is a Three Digit Number");
10         }else{
11             System.out.println(num + " is not a Three Digit Number");
12         }
13     }
14 }
15

```

```

samwa@Sami MINGW64 ~/OneDrive/desk
$ java ThreeDig
420 is a Three Digit Number

```

For different values of num :

```

samwa@Sami MINGW64 ~/OneDrive/desk
$ java ThreeDig
-911 is a Three Digit Number

```

```

samwa@Sami MINGW64 ~/OneDrive/desk
$ java ThreeDig
-17 is not a Three Digit Number

```

```

samwa@Sami MINGW64 ~/OneDrive/desktop/java/
$ java ThreeDig
2611 is not a Three Digit Number

```

# Else-if Ladder

28 November 2022 07:46

## Syntax :

```
If ( condition 1)
    // statement;
Else if (condition 2)
    // statement;
Else if (condition 3)
    // statement;
.
.
.
Else if (condition nth)
    // statement;
Else
    // statement;
```

- Q. Print "Three Three Three" if a number is divisible by three .  
Print "Four Four Four " if a number is divisible by four.  
Print "Five Five Five " if a number is divisible by five.**

```
1  class ifElse
2  {
3      public static void main(String[] args)
4      {
5          int num = 20;
6
7          if (num%3 == 0)
8          {
9              System.out.println("Three Three Three");
10         }else if (num%4 == 0)
11         {
12             System.out.println("Four Four Four");
13         }else if (num%5 == 0)
14         {
15             System.out.println("Five Five Five");
16         }else{
17             System.out.println("Not Not Not");
18         }
19     }
20 }
21 }
22 }
```

- Q. WAP to find the largest of three numbers**

```
1  class Q2ifElse
2  {
3      public static void main(String[] args)
4      {
5          int a = 7;
6          int b = 98;
7          int c = 56;
8
9          if(a > b && a > c){
10             System.out.println( b + " is greater than " + b + " & " + c);
11         }else if( b > c){
12             System.out.println( b + " is greater than " + a + " & " + c);
13         }else{
14             System.out.println( c + " is greater than " + a + " & " + b);
15         }
16     }
17 }
18 }
```

Q. Write a program to print:

"reddy" if number is divisible by 3,

"rajput" if number is divisible by 5,

"reddy weds rajput" if number is divisible by 3 and also by 5,

"breakup" if number is neither divisible by 3 and 5

```
1 class QifElse
2 {
3     public static void main(String[] args)
4     {
5         int num = 11;
6
7         if (num%3 == 0 && num%5 == 0)
8         {
9             System.out.println("reddy weds rajput");
10        }else if (num%3 == 0)
11        {
12            System.out.println("reddy");
13        }else if(num%5 == 0)
14        {
15            System.out.println("rajput");
16        }else{
17            System.out.println("breakup");
18        }
19    }
20 }
21 }
```

8. WAP to find the largest among 4 distinct numbers.

```
1 class Lar4
2 {
3     public static void main(String[] args)
4     {
5         int num1 = 48;
6         int num2 = 666;
7         int num3 = -103;
8         int num4 = 944;
9         String nums = num1 + "," + num2 + "," + num3 + "," + num4;
10
11
12         if (num1 > num2 && num1 > num3 && num1 > num4)
13         {
14             System.out.println(num1 + " is The largest number among " + nums);
15         }else if(num2 > num3 && num2 > num4)
16         {
17             System.out.println(num2 + " is The largest number among " + nums);
18         }else if(num3 > num4)
19         {
20             System.out.println(num3 + " is The largest number among " + nums);
21         }else
22         {
23             System.out.println(num4 + " is The largest number among " + nums);
24         }
25     }
26 }
27 }
28 }
```

9. WAP to find the smallest among 4 distinct numbers.

```
1 class Small4
2 {
3     public static void main(String[] args)
4     {
5         int num1 = 48;
6         int num2 = 666;
7         int num3 = -103;
8         int num4 = 944;
9         String nums = num1 + "," + num2 + "," + num3 + "," + num4;
10
11
12         if (num1 < num2 && num1 < num3 && num1 < num4)
13         {
14             System.out.println(num1 + " is The Smallest number among " + nums);
15         }else if(num2 < num3 && num2 < num4)
16         {
17             System.out.println(num2 + " is The Smallest number among " + nums);
18         }
19         else if(num3 < num4)
```

```

16 {
17     System.out.println(num2 + " is The Smallest number among " + nums);
18 }
19 else if(num3 < num4)
20 {
21     System.out.println(num3 + " is The Smallest number among " + nums);
22 }else
23 {
24     System.out.println(num4 + " is The Smallest number among " + nums);
25 }
26 }
27 }

```

#### 10. WAP to find the middle value of 3 numbers.

```

1 class MidNum
2 {
3     public static void main(String[] args)
4     {
5         int num1 = 98;
6         int num2 = 24;
7         int num3 = 49;
8
9         if ((num1 > num2 && num1 < num3) || ( num1 < num2 && num1 > num3))
10        {
11            System.out.println( num1 + " is the middle value" );
12        }
13        else if((num2 > num1 && num2 < num3) || ( num2 < num1 && num2 > num3))
14        {
15            System.out.println( num2 + " is the middle value" );
16        }
17        else
18        {
19            System.out.println( num3 + " is the middle value" );
20        }
21    }
22 }
23

```

Q. WAP to check if the program is a leap year or not.

- Should be divisible by 4
- The centuries should be divisible by 4

Self-done

```

1 class LeapY
2 {
3     public static void main(String[] args)
4     {
5         int Y = 1983;
6
7         if(Y%4 == 0) // if Year is divisible by 4 enter the block
8         {
9             if(Y%100 == 0 && Y%400 != 0) // if The Year is a century and not divisible by 400
10            {
11                System.out.println(Y + " is not a Leap Year");
12            }
13            else
14            {
15                System.out.println(Y + " is a Leap Year" );
16            }
17        }else
18        {
19            System.out.println(Y + " is a not a Leap Year" );
20        }
21    }
22 }
23 }
24 }
25

```

Stolen\_from\_web

```

1 class LeapY2
2 {
3     public static void main(String[] args)
4     {
5         |
6         int Y = 2023 ;
7
8         if (( Y%4 == 0) && ( (Y%100 != 0) || (Y%400 == 0) ))
9         {
10            System.out.println( Y + " Is a Leap Year" );
11        }
12        else
13        {
14            System.out.println( Y + " Is not a Leap Year" );
15        }
16    }
17 }
18 }
19

```

Q. WAP to calculate how many numbers are there between 1-100 which are divisible by 3 and not divisible by 5;

```

class div3nt5
{
    public static void main(String[] args)
    {
        int a = 100;
        int res = a/3 - a/5 ;

        System.out.println(a);
    }
}

```

Q. WAP program to calculate how many leap year are there from 1 till present day

```

1 //WAP program to calculate how many leap year are there from 1 till present day
2 class LeapYtillD
3 {
4     public static void main(String[] args)
5     {
6
7         System.out.println( 2022/4 + 2022/400 - 2022/100 + "number of Leap Years are present till date ");
8     }
9 }
10

```

Q. WAJP to validate the date




# Switch

01 December 2022

12:58

Syntax :

Switch ( expression )  
{  
    Case value 1 : // code ; break ;  
    Case value 2 : // code ; break ;  
    .  
    .  
    .  
    .  
    Case value n : // code ; break ;



Int , String ,char

(optional) default : // code

Break : Break keyword is used to terminate a switch case. In other words once JVM sees break keyword the execution of switch block will stop .

Note : ex.  $17\%18$  if  $RHS > LHS$  then answer is LHS

Note :  $n\%0$  // exception error;

Q. WAP to perform a mathematical operation based on the operator using switch case

Q. WAP any program on switch case

# Loops

02 December 2022 12:55

Loop Is used to perform a task repeatedly .  
There are four types of looping statements :

1. **While**
2. **Do while**
3. **For**
4. **For - each**

All the loops have initialization condition and updation.  
All the loops run till the condition is true .

## While :

Syntax

```
While(condition)
{
    //statement
}
```

Example:

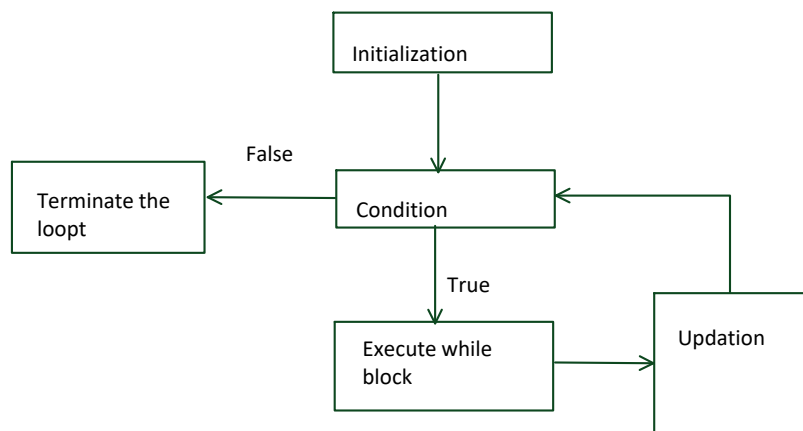
```
1 class WhileEx
2 {
3     public static void main(String[] args)
4     {
5         int i = 1 ; //initilization
6         while( i <= 10){
7
8             System.out.println("Hello World!");
9             i++;
10        }
11    }
12 }
13 }
14 }
```

```
samwa@Sami MINGW64 ~/OneDr
$ java WhileEx
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
```

Example :

```
1 class WhileEx
2 {
3     public static void main(String[] args)
4     {
5         int i = 5 ; //initilization
6         while( i >= 1 ){
7
8             System.out.println("Hello World!");
9             i--;
10        }
11        System.out.println("Out of the Loop" ) ;
12    }
13 }
14 }
15 }
```

```
samwa@Sami MINGW64 ~/OneDri
$ java WhileEx
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
Out of the Loop
```



Q. WAP to Print numbers from 1 to 20

```
1 class WhileEx
2 {
3     public static void main(String[] args)
4     {
5         int i = 1 ; //initilization
6         while( i <= 20 ){
```

```
$ java WhileEx
1
2
3
4
5
6
7
```

```

1 class WhileEx
2 {
3     public static void main(String[] args)
4     {
5         int i = 1 ; //initialization
6         while( i <= 20 ){
7
8             System.out.println(i);
9             i++;
10        }
11        System.out.println("Out of the Loop" ) ;
12    }
13 }
14
15

```

```

3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
Out of the Loop

```

Q. WAP to Print numbers from 1 to n .

```

1 class WhileEx
2 {
3     public static void main(String[] args)
4     {
5         int i = 1 ;
6         int n = 7;
7
8         while( i <= n ){
9
10            System.out.println(i);
11            i++;
12        }
13    }
14 }
15
16
17

```

```

C:\Users\samwa\OneDrive
1
2
3
4
5
6
7

```

Q. WAPTP numbers from m to n

```

1 class WhileEx
2 {
3     public static void main(String[] args)
4     {
5         int m = 12 ;
6         int n = 17 ;
7
8         while( m <= n){
9
10            System.out.println(m);
11            m++;
12        }
13    }
14 }
15
16

```

```

C:\Users\samwa\OneDrive
12
13
14
15
16
17

```

Q. WAPTP numbers from m to n

```

1 class WhileEx
2 {
3     public static void main(String[] args)
4     {
5         int m = 12 ;
6         int n = 17 ;
7
8         if( m < n ) {
9             while( m <= n){
10                System.out.println(m);
11                m++;
12            }
13        }
14        else
15        {
16            while( n <= m){
17                System.out.println(n);
18                n++;
19            }
20        }
21    }
22 }
23
24
25

```

Q. WAPTP even numbers from m to n

```

1 class NtoMeven
2 {
3     public static void main(String[] args)
4     {
5         int m = 33 ;
6         int n = 44 ;
7
8         while( m <= n ){
9
10            if( m % 2 == 0 ){
11                System.out.println(m);
12                m++;
13            }
14            else{
15                m++;
16            }
17        }
18    }
19 }
20
21
22

```



```

2  {
3      public static void main(String[] args)
4      {
5          int m = 33 ;
6          int n = 11 ;
7
8          if (m > n)
9          {
10             while( n <= m )
11             {
12                 if(n%2 == 0)
13                 {
14                     System.out.println(n);
15                 }
16                 n++ ;
17             }
18         }
19         else
20         {
21             while(m <= n)
22             {
23                 if (m%2 == 0)
24                 {
25                     System.out.println(m);
26                 }
27                 m++ ;
28             }
29         }
30     }
31 }
32

```

```

C:\Users\samwa\OneDrive\Desktop
12
14
16
18
20
22
24
26
28
30
32

```

Q. WAPTP Odd numbers from m to n

```

1  class MtoNOdd
2  {
3      public static void main(String[] args)
4      {
5          int m = 10 ;
6          int n = 21;
7
8          if(m%2 == 0)
9          {
10             m++ ;
11         }
12         while (m <= n)
13         {
14             System.out.println(m);
15             m += 2;
16         }
17     }
18 }
19
20

```

```

samwa@Sami MINGW
$ java MtoNOdd
11
13
15
17
19
21

```

Q. WAPTP sum of natural numbers from 1 to n using loops :

```

1  class SumNat
2  {
3      public static void main(String[] args)
4      {
5          int n = 10 ;
6          int i = 1 ;
7          int res = 0 ;
8          while( i <= n)
9          {
10             res = res + i;
11             i++ ;
12         }
13         System.out.println(res);
14     }
15 }
16

```

```

samwa@Sami MINGW64 ~/C
$ java SumNat
55

```

**Do While Loop :** Do while loop is executes first and then checks the condition. Hence a do while loop block is executed at least once even if the condition is false.

Syntax :

```

Do
{
    // Statement

```

```

    }
    While(condition);

```

```

1  class Dowhile
2  {
3      public static void main(String[] args)
4      {
5          int i = 1, n = 5 ;
6          do
7          {
8              System.out.println("inside do While loop");
9              i++ ;
10         }
11         while (i <= n);
12     }
13 }
14 }
15

```

```

samwa@Sami MINGW64 ~/OneDr
$ java Dowhile
inside do While loop
inside do While loop
inside do While loop
inside do While loop
inside do While loop

```

Q. WAP to find the sum of natural numbers from 1 to n using do-while loop

```

1  //Sum of natural numbers from 1 to n with do while
2  class NatNumSum
3  {
4      public static void main(String[] args)
5      {
6          int n = 5 ;
7          int i = 1 ;
8          int sum = 0 ;
9          do
10         {
11             sum += i ;
12             i++ ;
13         }
14         while (i <= n);
15         System.out.println(sum) ;
16     }
17 }
18

```

Q. WAP to Print all the numbers from m to n which are divisible by 15 using do while do while loop in a very efficient way.

```

HelloWorld.java  NumBlock.java  Div15TillN.java x  Console
1  package com.TestPackage;
2
3  public class Div15TillN {
4
5      public static void main(String[] args) {
6
7          int m = 7 ;
8          int n = 100 ;
9
10         if(m%15 != 0)
11         {
12             m = m + (15 - m%15) ;
13         }
14
15         for(int i = m ; i <= n ; i+=15)
16             System.out.println(i);
17     }
18 }
19 }
20

```

```

<terminated>
30
45
60
75
90

```

**For loop :**

Syntax :

```

For ( initialization ; condition ; updation )
{
    //statements
}

```

Inr n = 10

```

For( int i= 1 ; i<n ; i++ )
{
    sopln(i) ;
}

```

Q. WAP to find the factorial of a number .

```

1  class Factorial
2  {
3      public static void main(String[] args)

```

```

1  class Factorial
2  {
3      public static void main(String[] args)
4      {
5          int n = 5;
6          int fac = 1 ;
7          for (int i = n; i >= 1 ; i-- )
8          {
9              fac = fac * i ;
10         }
11
12         System.out.println(n + " factorial is " + fac);
13     }
14 }
15

```

Q. Write a program to print 5 stars in the same line

```

int n = 5 ;
for( int i=1 ; i<n ; i++ )
{
    Sop("* ");
}

```

Q. Write a program to print n number "\*" cube :

```

1  class Stars
2  {
3      public static void main(String[] args)
4      {
5          int n = 5 ;
6          for (int i = 1; i <= n ; i++ )
7          {
8              for(int q = 1; q <=n ; q++)
9              {
10                 System.out.print("* ");
11             }
12             System.out.println();
13         }
14     }
15 }
16

```

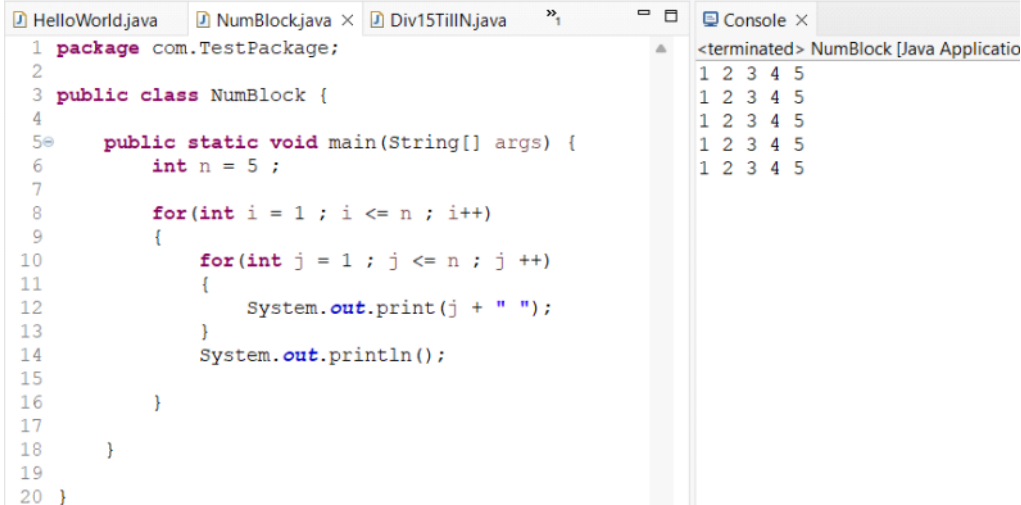


```

C:\Users\samw
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *

```

Q. Write a program to print 1 to n block of square



```

1  package com.TestPackage;
2
3  public class NumBlock {
4
5      public static void main(String[] args) {
6          int n = 5 ;
7
8          for(int i = 1 ; i <= n ; i++)
9          {
10             for(int j = 1 ; j <= n ; j ++ )
11             {
12                 System.out.print(j + " ");
13             }
14             System.out.println();
15         }
16     }
17 }
18
19
20 }

```

```

<terminated> NumBlock [Java Applicatio
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5

```

Q. Write a program to print n to 1 block of square

```

1 package com.TestPackage;
2
3 public class NumBlockReverse {
4
5     public static void main(String[] args) {
6         int n = 5 ;
7
8         for(int i = 1 ; i <= n ; i++)
9         {
10             for(int j = n ; j >= 1 ; j--)
11             {
12                 System.out.print(j + " ");
13             }
14             System.out.println();
15         }
16     }
17 }
18
19
20 }
21

```

Console: <terminated> NumBlockRevers  
5 4 3 2 1  
5 4 3 2 1  
5 4 3 2 1  
5 4 3 2 1  
5 4 3 2 1

### Differences between for While , Do While and For loop

While	Do-While	For
Initialization has to be done outside The block	Initialization has to be done outside the Block	Initialization can be done outside the for block, within the for Syntax or not done at all
Updation has to be done inside the block	Updation has to be done inside the block	Updation can be done inside the Syntax or within the Syntax
Condition is mandatory	Condition is mandatory	Condition is not mandatory, by default the condition is true.
We generally use while loop when the number of iterations is unknown	We use do-while loop when we want to perform a task at least once	We generally use for loop over any collection (Arrays)

### Q. WAP to print multiplication table of a number

```

1 package com.TestPackage;
2
3 public class Test {
4
5     public static void main(String[] args) {
6         int n = 7;
7
8         for(int i = 1 ; i <= 10 ; i++)
9         {
10             System.out.println(n + "*" + i + "=" + n*i );
11         }
12     }
13 }
14
15
16 }

```

Console: <terminated> Test [Jav  
7\*1=7  
7\*2=14  
7\*3=21  
7\*4=28  
7\*5=35  
7\*6=42  
7\*7=49  
7\*8=56  
7\*9=63  
7\*10=70

### Q. WAP to print 1 by 1 digit from the end of a n

```

1 package com.TestPackage;
2
3 public class Test {
4
5     public static void main(String[] args) {
6         int n = 123456;
7
8         for(int i = 1 ; n != 0 ; i++)
9         {
10             System.out.print(n%10);
11             n = n/10 ;
12         }
13     }
14 }
15
16 }

```

Console: <terminated> Test [Jav  
654321

### Q. WAP a program to count how many digits are there in a number

```

1 package com.TestPackage;
2
3 public class NumLength {
4
5     public static void main(String[] args) {
6         int n = 0 ;
7         int temp = n ;
8         int i = 0 ;
9         do
10            {
11                n = n/10 ;
12                i++ ;
13            }
14        while(n != 0);
15        System.out.println(temp + " is a " + (i) + " digit number");
16    }
17
18 }
19

```

<terminated> NumLength [Java Applic  
0 is a 1 digit number



# Methods :

07 December 2022 13:17

- Method is a block of memory to store the instructions
- Method is a named memory location to perform some specific task

Syntax :

```
Public static datatype methodName ( parameter1, parameter2 , .....parameterN); ← Method declaration
optional
{
//statement ← Method implementation
}
```

The screenshot shows an IDE with a file named 'Test.java' open. The code defines a class 'Test' with two methods: 'main' and 'displayFeatures'. The 'main' method calls 'displayFeatures' and prints 'execution starts from main method' and 'execution stops at main method'. The 'displayFeatures' method prints several statements about Java: 'java is Simple', 'java is fast', 'java is object oriented', 'java is platform independent', 'java is secure', 'java has rich inbuilt resources', and 'java is case sensitive'. The console output on the right shows the execution of the 'main' method, which prints 'execution starts from main method', calls 'displayFeatures', which prints the seven statements, and then 'main' prints 'execution stops at main method'.

## Class Loading :

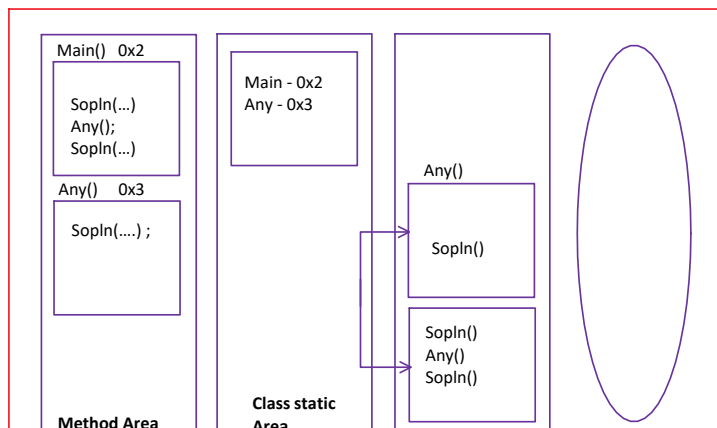
- Loading the class into the memory is called as class loading process .
- JRE is responsible for loading a class.
- JVM is responsible for execution .
- JRE memory is classified into four types :
  1. Class static area / Static area
  2. Method area
  3. Stack area
  4. Heap area
- Class is loaded in Class static area.
- Class loader which is a part of JRE is responsible for loading a class.
- In this area Static members are stored .
- Method area is used to store instructions .
- Stack area is used for execution .
- Heap area is used to Store an Object .
- Execution starts at **main** method and ends at **main** method.

Q. WAP your personal method using a method

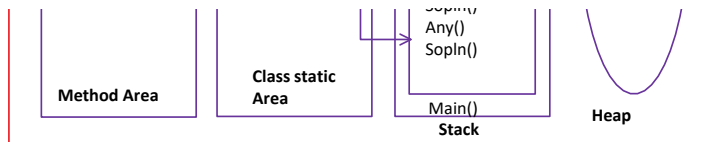
The screenshot shows an IDE with a file named 'Test.java' open. The code defines a class 'Test' with two methods: 'main' and 'myDetails'. The 'main' method calls 'myDetails' and prints 'main method started' and 'main method ended'. The 'myDetails' method prints several statements about a person: 'name : Samiran S Waghmare', 'phone : 8087422098', 'Stream : Instrumentation', 'YOP : 2021', and 'Address : Pen,Raigad 402107'. The console output on the right shows the execution of the 'main' method, which prints 'main method started', calls 'myDetails', which prints the five statements, and then 'main' prints 'main method ended'.

Class Loading Procedure :

JRE- RAM



The screenshot shows an IDE with a file named 'Test.java' open. The code defines a class 'Test' with two methods: 'main' and 'any'. The 'main' method calls 'any' and prints 'main method started' and 'main method ended'. The 'any' method prints 'any method is executed'. The console output on the right shows the execution of the 'main' method, which prints 'main method started', calls 'any', which prints 'any method is executed', and then 'main' prints 'main method ended'.



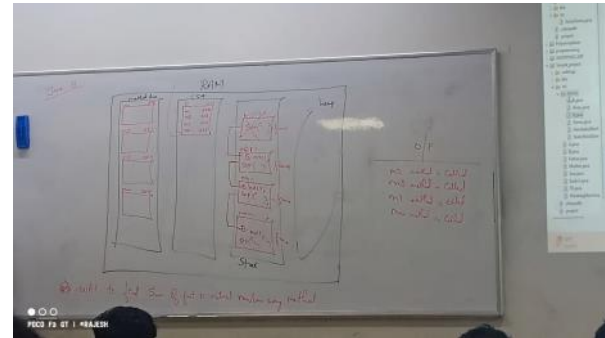
any method is executed  
main method ended

```

Test.java x
1 package com.TestPackage;
2
3 public class Test {
4     public static void m1() {
5         Test.m2();
6         System.out.println("m1 method is called");
7     }
8     public static void m2() {
9         Test.m3();
10        System.out.println("m2 method is called");
11    }
12    public static void m3() {
13        System.out.println("m3 method is called");
14    }
15
16    public static void main(String[] args) {
17        Test.m1();
18        System.out.println("main method started");
19    }
20
21 }
22
  
```

```

Console x
<terminated> Test (Java Applicati
m3 method is called
m2 method is called
m1 method is called
main method started
  
```



Q. Find the sum of first n natural numbers using method

```

Test.java x
1 package com.TestPackage;
2
3 public class Test {
4
5     public static void main(String[] args) {
6         SumNat();
7     }
8
9     public static void SumNat() {
10        int n = 10;
11        int i = 1;
12        int sum = 0;
13        while(i <= n) {
14            sum = sum + i;
15            i++;
16        }
17        System.out.println(sum);
18    }
19
20 }
21
  
```

```

Console
<terminated>
55
  
```

Q. WAP to print numbers from 1 to 15 using method

```

1 package com.TestPackage;
2
3 public class NtoM {
4
5     public static void main(String[] args) {
6         PrintNtoM();
7     }
8
9     public static void PrintNtoM() {
10        int n = 1;
11        int m = 15;
12        for(int i = n; i <= m; i++) {
13            System.out.println(i);
14        }
15    }
16
17 }
18
19
  
```

```

<terminated> NtoM
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
  
```

Q. WAP to find the factorial of the number using method

```

Test.java x NtoM.java x Factorial.java x
1 package com.TestPackage;
2
3 public class Factorial {
4     public static void main(String[] args) {
5         facto();
6     }
7
8     public static void facto() {
9        int n = 5;
10       int factorial = 1;
11       for (int i = 1; i <= n; i++) {
12           factorial = factorial * i;
13       }
14       System.out.println(factorial);
15    }
16
17 }
  
```

```

Console x
<terminated>
120
  
```

Q. WAP to find the max of 3 numbers using methods

```

1 package com.TestPackage;
2
3 public class MaxOf3 {
4     public static void main(String[] args) {
5         max3(10,4,23);
6     }
7
8     public static void max3(int a, int b, int c) {
9
10        if(a>b && a>c) {
11            System.out.println(a + " is the largest number");
12        }else if(b>c){
13            System.out.println(b + " is the largest number");
14        }else {
15            System.out.println(c + " is the largest number");
16        }
17    }
18 }
19

```

Console x

<terminated> MaxOf3 [Java Application] C:\Program Files\Java\jdk-19\bin\javaw.exe (10-Dec-2022, 12:5  
23 is the largest number

#### Return Type Methods :

Return is a key word which is used to perform three important tasks

1. Return a value back to the calling method
2. Return the control back to the calling method
3. Stop the execution of the method

#### Rules for Return Type Methods :

1. At least one return statement should be executed from a return type method.
2. Return statement should be in the last line of code in any block
3. The returning value and the return type of the method should be same.

**note :** if there are multiple return statements in a method , the first statement whichever JVM encounters will be executed.

#### Q. WAP to find the square of largest of three numbers

```

1 package com.TestPackage;
2
3 public class Test {
4
5     public static void main(String[] args) {
6         int a = Test.max3(10,4,1);
7         System.out.println(a*a);
8     }
9
10    public static int max3(int a, int b, int c) {
11
12        if(a>b && a>c) {
13            return a;
14        }else if(b>c){
15            return b;
16        }return c ;
17    }
18 }
19
20 }
21

```

Console x

<terminated> Test [Ja  
100

#### Q. WAP to find factors of a number

```

1 package com.TestPackage;
2
3 public class Factors {
4
5     public static void main(String[] args) {
6         int n = 8 ;
7
8         for(int i= 1 ; i<= n ; i++) {
9
10            if(n%i == 0) {
11                System.out.println(i);
12            }
13        }
14    }
15
16 }
17 }

```

Console x

<terminated>  
1  
2  
4  
8

#### Q. WAP to find if a given number is a perfect number or not

```

1 package com.TestPackage;
2
3 public class PerfectNum {
4
5     public static void main(String[] args) {
6         boolean p = PerfectNum.Perfect(28);
7         System.out.println(p);
8     }
9
10    public static boolean Perfect(int n) {
11
12        int Sum = 0;
13
14        for(int i = 1; i < n; i++) {
15            if(n%i == 0) {
16                Sum = Sum + i;
17            }
18        }
19        return Sum == n;
20    }
21 }
22
23
24 }

```

Console Output: true

#### Advantages of Methods :

- Reduces code complexity
- Modification is easy
- Polymorphism can be achieved

#### Method Overloading :

- Methods with the same name irrespective of the return type but with different signature(collection of parameters) is called as method overloading
- Method over loading is nothing but performing the same task on the basis of argument given .

#### Signature :

- Parameters collectively are called as signature.

#### Method Binding

- Associating method call statements with respect to method implementation is called as method binding.
- During Method overloading, method binding happens purely on the basis of
  - a. number of parameters
  - b. type of parameters
  - c. order of the parameters

*note : We cannot have duplicate methods inside a Class .*

```

psvm(String args[])
{
    int d = (int)(add(10.5,11.5));
    sop(d) ; //22
}

```

#### You can achieve method overloading

- By different number of parameters
- By different types of parameters
- By changing the order of parameters

To overload a method the number of parameters should be different or the datatype of the parameter should be different or the order of the parameters should be different

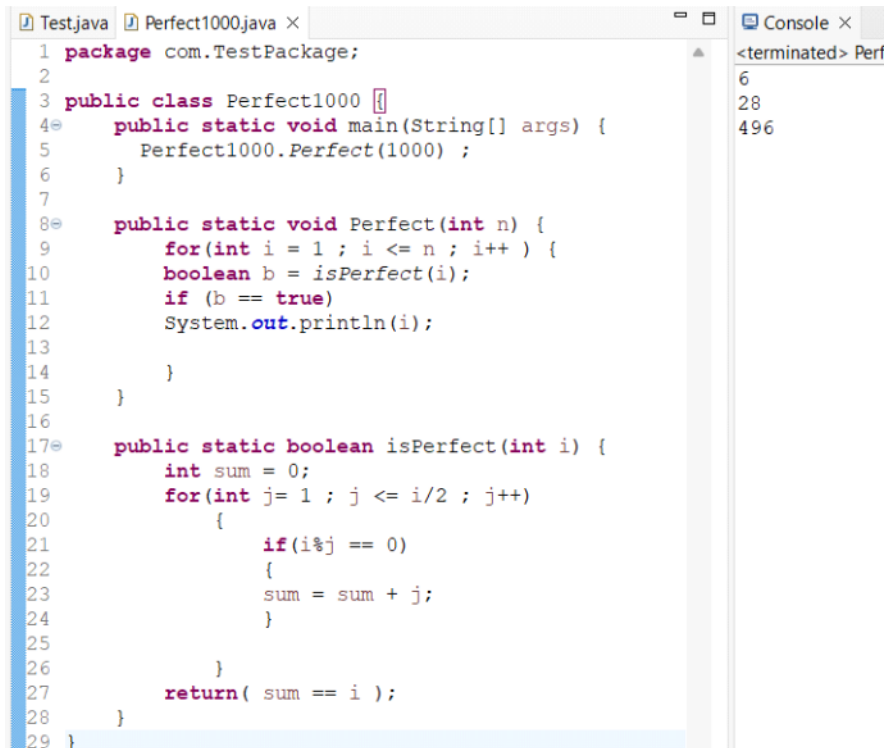
# Recursion

12 December 2022 14:20

## Recursion :

- A Method calling itself is called as recursive method.
- Without a base condition many method frames are created in stack area which will lead to stack overflow error.

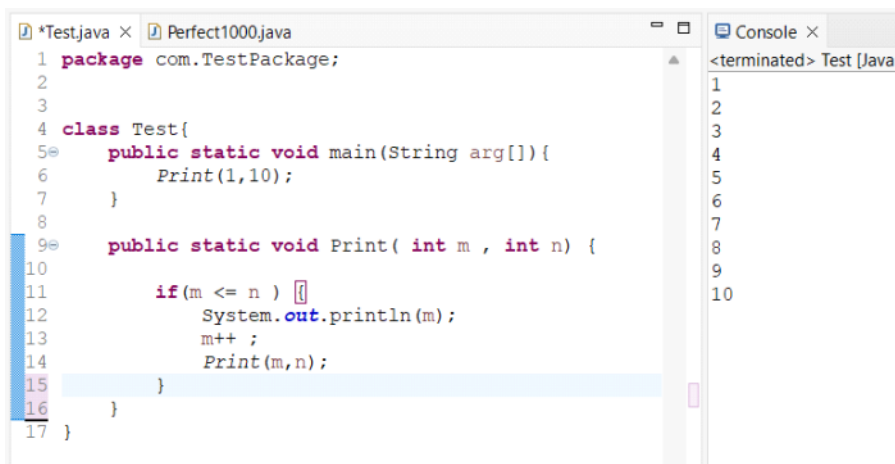
**Q. WAP to print perfect numbers from 1 to 1000 .**



```
1 package com.TestPackage;
2
3 public class Perfect1000 {
4     public static void main(String[] args) {
5         Perfect1000.Perfect(1000) ;
6     }
7
8     public static void Perfect(int n) {
9         for(int i = 1 ; i <= n ; i++ ) {
10             boolean b = isPerfect(i);
11             if (b == true)
12                 System.out.println(i);
13         }
14     }
15
16     public static boolean isPerfect(int i) {
17         int sum = 0;
18         for(int j= 1 ; j <= i/2 ; j++)
19         {
20             if(i%j == 0)
21             {
22                 sum = sum + j;
23             }
24         }
25         return( sum == i );
26     }
27 }
28
29 }
```

Console <terminated> Perf  
6  
28  
496

**Q. WAPTP numbers from 1 to 10 using recursion**



```
1 package com.TestPackage;
2
3
4 class Test{
5     public static void main(String arg[]){
6         Print(1,10);
7     }
8
9     public static void Print( int m , int n) {
10
11         if(m <= n ) {
12             System.out.println(m);
13             m++ ;
14             Print(m,n);
15         }
16     }
17 }
```

Console <terminated> Test [Java  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10

**Q. WAPTP numbers from 1 to 10 using recursion**

```

Test.java x Perfect1000.java
1 package com.TestPackage;
2
3
4 class Test{
5     public static void main(String arg[]){
6         Print(1,10);
7     }
8
9     public static void Print( int m , int n) {
10
11         if(m <= n ) {
12             System.out.println(n);
13             n--;
14             Print(m,n);
15         }
16     }
17 }

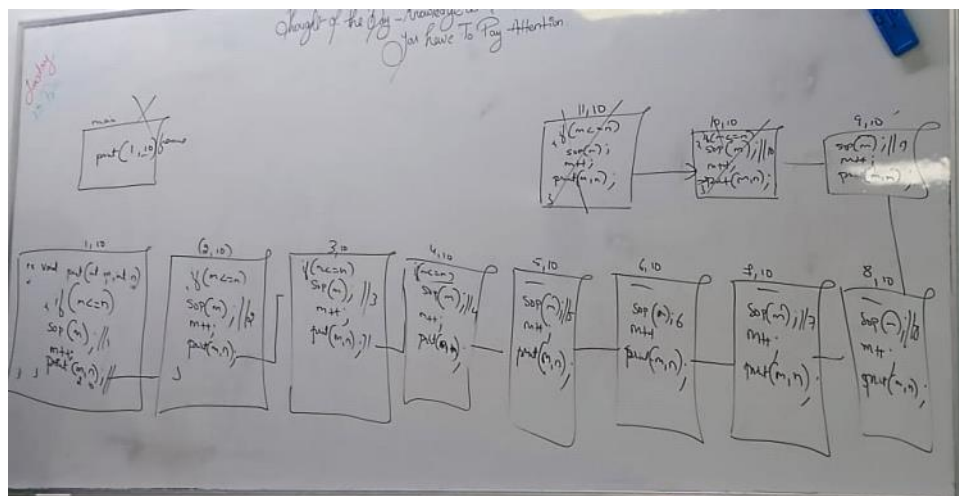
```

Console x

```

<terminated> T
10
9
8
7
6
5
4
3
2
1

```



Q. WAPTP numbers from 10 to 1 using recursion

```

Test.java x Perfect1000.java
1 package com.TestPackage;
2
3
4 class Test{
5     public static void main(String arg[]){
6         Print(1, 10 );
7     }
8
9     public static void Print( int m , int n) {
10
11         if(m <= n ) {
12             m++ ;
13             Print(m,n);
14             System.out.println(m-1);
15         }
16     }
17 }

```

Console

```

<terminate
10
9
8
7
6
5
4
3
2
1

```

# Parameters & Arguments

09 December 2022 13:14

## Parameter :

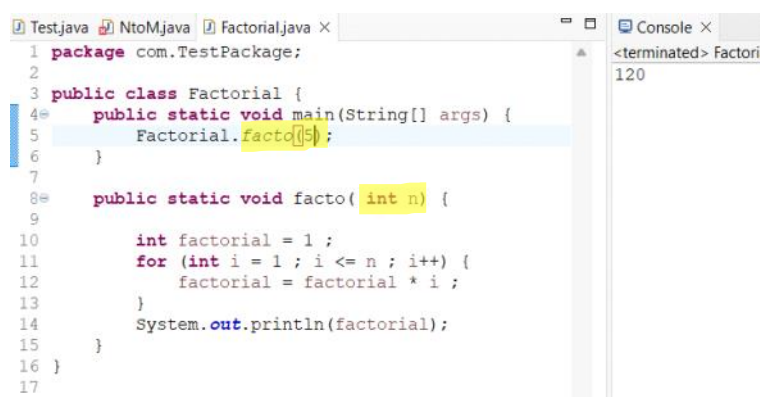
- Method with one or more parameters can be termed as parameterized methods.
- Parameters are nothing but local variables declared at the time of method declaration part

## Arguments :

- Arguments are values given to a method during method calling statement.

Note : 1. the number of parameters and number of arguments should be same.

2. the datatype of the parameter and the datatype of the argument passed should be same.



```
1 package com.TestPackage;
2
3 public class Factorial {
4     public static void main(String[] args) {
5         Factorial.fact(5);
6     }
7
8     public static void fact(int n) {
9
10        int factorial = 1 ;
11        for (int i = 1 ; i <= n ; i++) {
12            factorial = factorial * i ;
13        }
14        System.out.println(factorial);
15    }
16 }
17
```

Console <terminated> Factori  
120