

Full Stack Development Java

Front End

Back End

Core Java

Basics of Java

Java intro, Installation and Setup for Java, Writing and executing first java program, Data Type, Control flow statements, Operators, Array, Class, Objects, methods, Java Build-class, String classes.

Object Oriented Programming in Java

Encapsulation, Inheritance, Polymorphism, Abstraction, constructor, Java keyword, final, static, super, this, access modifier, package

Advance of Core Java

Exception Handling, Threading, Collection, JDBC.

Advance Java (JSP, Servlet)

Spring Boot Framework

Database

SQL, MYSQL Database

Tools (**DevTool**)

Git

GitHub

Maven

Postman

Swagger

Software

JDK

Eclipse IDE

MySQL Database

Git

Maven

Postman

Download JDK (1.8 and above).

<https://www.oracle.com/in/java/technologies/downloads/>

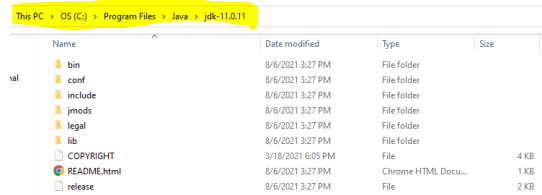
Note: Download Installer instead of zip file

Java Documentation

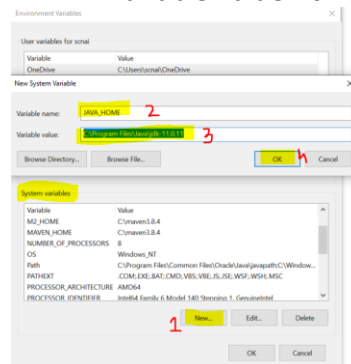
<https://docs.oracle.com/en/java/javase/11/docs/api/>

Java Setup

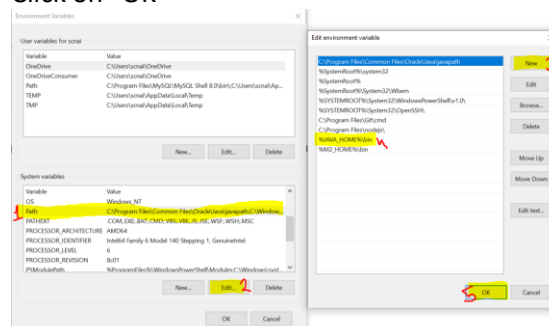
1. Download JDK
<https://www.oracle.com/java/technologies/downloads/>
Download the Installer for you operating System.
2. Install JDK
3. Verify the Installation
 - a. You can verify the installation file into C:\Program File\java\<jdk-version-folder>



4. Setup Environment Variable
 - a. Set **JAVA_HOME**
 - i. Go to Start and Search for “Environment” word and select the “Edit System Environment Variable” option.
 - ii. Click on the “Environment Variable” button on the new window.
 - iii. Select “New” Button from the ‘System Variables’ section
 - iv. Provide the following details into text box
 1. Variable Name: **JAVA_HOME**
 2. Variable Value: <JDK-Path>

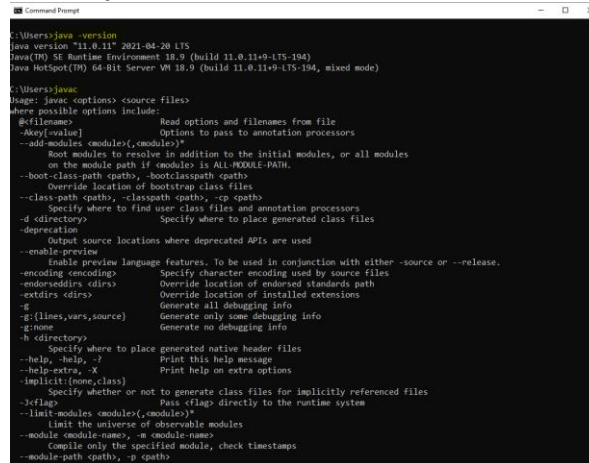


- b. **Set Path**
 - i. Check for “path” variable inside ‘system variables’ section
 - ii. Select “Path” variable and click on “Edit” Button
 - iii. Click the “New” button on the new window
 - iv. And set the variable value as follows
%JAVA_HOME%\bin
 - v. Click on “OK”



5. Verify the Environment Variable Setup

- a. Open CMD
- b. Try the following commands
 - i. **java -version**
 - ii. **javac**



```
C:\Users\java>java -version
java version "11.0.11" 2021-04-20 LTS
Java(TM) SE Runtime Environment 18.9 (build 11.0.11+9-LTS-194)
Java HotSpot(TM) 64-Bit Server VM 18.9 (build 11.0.11+9-LTS-194, mixed mode)

C:\Users\java>javac
Usage: javac <options> <source files>
where possible options include:
  -d <dirname>           Read options and filenames from file
                        Options to pass to annotation processors
  --add-modules <module> Root modules to resolve in addition to the initial modules, or all modules
                        on the module path if <module> is ALL-MODULE-PATH.
  --boot-class-path <path> Override location of bootstrap class files
  --class-path <path>,-classpath <path>,-cp <path> Specify where to find user class files and annotation processors
  -d <directory>         Specify where to place generated class files
  --deprecation           Output source locations where deprecated APIs are used
  --enable-preview       Enable preview language features. To be used in conjunction with either -source or --release.
  --encoding <encoding> Specify character encoding used by source files
  --endorseddirs <dirs>  Override location of endorsed standards path
  --extdirs <dirs>       Override location of installed extensions
  -g                     Generate all debugging info
  -g:{lines,vars,source} Generate only some debugging info
  -g:none                Generate no debugging info
  -h <directory>         Specify where to place generated native header files
  --help, -help, -?      Print this help message
  --help-extra, -X       Print help on extra options
  --implicit:(none,class) Specify whether or not to generate class files for implicitly referenced files
  -J<flag>               Pass <flag> directly to the runtime system
  --limit-modules <module>{,<module>}* Limit the universe of observable modules
  --module <module-name>,-m <module-name> Compile only the specified module, check timestamps
  --module-path <path>,-p <path>
```

Types Of Application/Software

Mobile Application
Web Application
Desktop Application
Embedded Application
Console Based Application

Java Intro

1. Edition of Java
 - a. J2SE/JSE
 - i. JSE stands for Java Standard Edition
 - ii. It is also known as core java.
 - iii. Desktop and console based application can be developed in this edition.
 - b. J2EE/JEE
 - i. JEE stands for Java Enterprise Edition
 - ii. Is also known as Advance Java.
 - iii. Mainly used for developing Web Application
 - c. J2ME/JME
 - i. JME stands for Java Micro Edition
 - ii. Mainly used for Mobile application and Embedded Application.

Key Component for the Java Program

JVM

1. JVM stands for Java Virtual Machine
2. It is mainly use to execute the java program.
3. Allocation of the memory while execution of the program, Communication with OS for execution, Clearing up memory after specific interval will be taken care by JVM.
4. There are three stages of Program execution in JVM
 - a. Class Loader
 - b. Bytecode verifies
 - c. Execution

JDK

1. JDK stands for Java Development Kit.
2. This provide an environment to develop and execute the java program.
3. JDK is a combination of multiple tools/components such as JVM, JRE, Lib(API) and some of development tools like javac, javap, javadoc

JRE

1. JRE stands for Java Runtime Environment.
2. JRE is used to execute the Java Programs only.
3. JRE must be present on Client system.

API

1. API stands for Application Programming Interface.
2. API are the set of predefine class, interfaces and the functionalities provided by Java.
3. These APIs will be available in the form of .jar file in java.

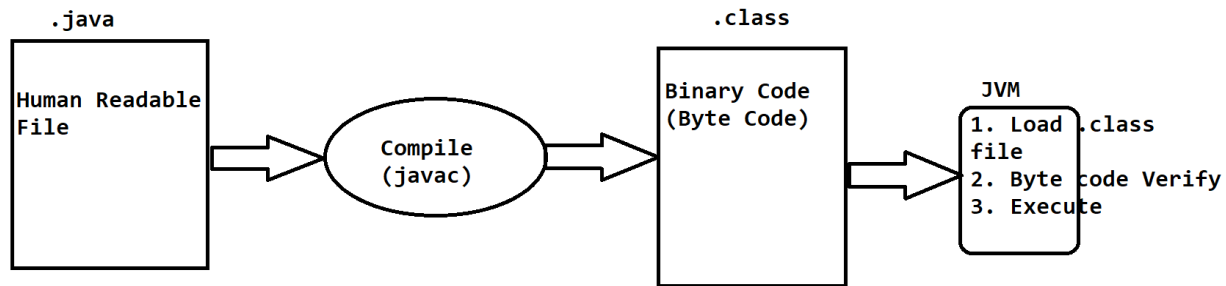
Writing and Executing First Java Program

1. Writing a java code
 - a. You can write a using IDE or Notepad or any editor tool.
2. Java Program Syntax
 - a. Create Java class
 - i. In java everything must be write inside class except import and package statement.
 - ii. Syntax:
public class <Class_Name>
{

}
 - b. Cerate Main Method inside java class
 - i. Main method is the first method which gets executed.
 - ii. This method is a start point of java program.
 - iii. This method will be called internally by java (JVM) when you execute your program.
 - iv. Syntax:
public static void main(String args[])
{

}
 - c. Write an executable code into method.
System.out.println("Hello, Welcome to first java program");
3. Save File
 - a. File Name must be same as public class name.
 - b. File extension must be **.java**.
4. Compile the java code
 - a. In this sept the java code will be converted into a byte code and the binary file will be created.
 - b. As an outcome of the compilation step the .class file will be created.
 - c. In this step all the syntax validation will also happens.
 - d. To compile a code first Open a CMD
 - i. The path of the CMD must be pointing to a location, where you save your java class. (to do this you can go to a folder where you save your java class, click on the address bar and type "cmd" and hit enter)
 - ii. Execute a command to compile
javac FileName.java
5. Execute Java code
 - a. Use same CMD
 - b. Execute a command to run java program
java className

Java Program Execution Process



Rules

1. To execute the java program you don't required a .java file you can execute the program using by .class file.
2. One java file (Source file) can have more than one java class. Just you have to make sure that only one class must be public. You can have main method in all the classes in a file.
3. The .class files will be generated for a classes present inside a file and not for a .java file.

Keyword, Identifiers and literals

Keyword

1. Keywords are the reserve words used by a java programming language.
2. The meaning for these words are implemented by java internally.
3. All keywords are in small case only.
4. There are 52 keywords present in java
5. Example:
public, class, static, void, int, byte, short, long, double, float, boolean, char, if, else, for, while, do, switch, default, final, super, this, null, true, false.

Literal

1. Literals are the values. Mostly String values are also consider as literals.
2. The values which are also consider as keywords are known as literal
3. Example: null, true, false.

Identifier

1. Identifiers are the words which can be used for coding perspective are knowns as Identifier.
2. The words which are used for class, method, variable, Object name are consider as an Identifier.
3. There are rules to create Identifier in java
 - a. Identifier must not be keyword
 - b. Identifier must be start with character or symbol (\$ and _)
 - c. In Identifier there are only two symbols are allowed which \$ and _
 - d. Identifier can contain numbers but it must not be start with number.
 - e. Identifier are case sensitive and spaces are not allowed in it.
 - f. There is no limit of number of character in Identifier.
4. There are conventions to create a class Identifier
 - a. The name of a class should be start with capital case.
 - b. If it is a combination of multiple words then every word should be start with capital
 - c. Example: Student, StudentDetails, EmployeePersonalDetails

5. There are conventions to create a method, variable, Object Identifier
 - a. The name of variable, method or object should be start with lower case.
 - b. If it is a combination of multiple words then it should be start with lower case and all other words should be start with capital.
 - c. Exaxmple: printDetails, employeeId, printStudentDetails
6. There are conventions to create a constant
 - a. The name of the constant should be in a complete capital case
 - b. If it is a combination of more that one word then every word should be separated by '_'
 - c. Example: PI, GRAVITY, COPMPANY_NAME

```
public class Welcome
{
    public static void main(String args[])
    {
        System.out.println("Hello, Welcome to first java program");
    }
}
Keywords: public, class, static, void
Identifier : Welcome, main, String, System, out, println, args
```

Comments in Java

1. Comments are the way to write an information inside a code which can be use for a future use.
2. By using comment you can make any code non-eligible for the execution.
3. There are 3 types of comments in java

- a. Single line comment

- i. Syntax

// line to comment

- b. Multi-line comment

- i. Syntax

/*

line to comment

*/

- c. Documentation Comment

- i. It is use to provides the code level documentation. These comments are also a part of compiled code.

- ii. Syntax

/**

Code level documentation

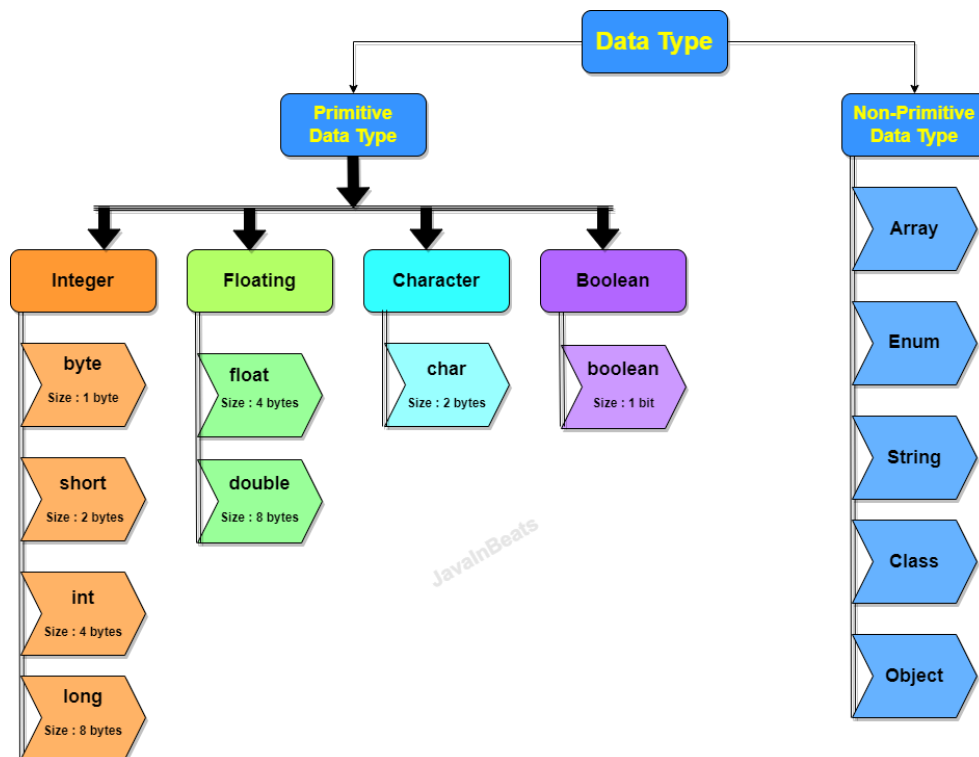
*/

```
/** Documentation Comment
 * This class created to domo the simple java program..
 */
public class Welcome
{
    // this is the method called by JVM single line Comment
    public static void main(String args[])
    {
        // this line use to print the output to the user.
        System.out.println("Hello, Welcome to first java program");
        /* multi line comment
        System.out.println("This is the trial output");
        System.out.println("This is the another trial output");
        */
    }
}
```

Data Type and variable

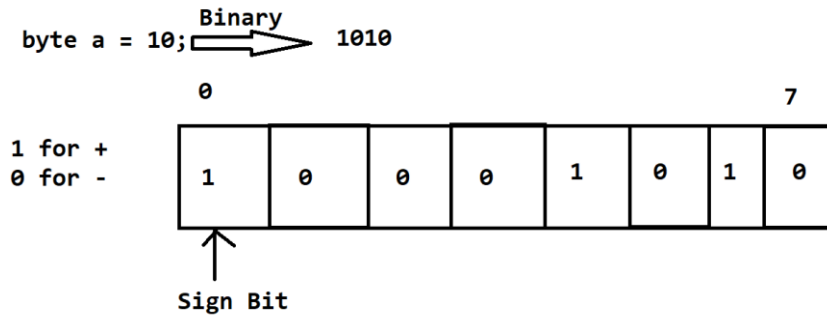
Data Type

1. Using Data type you can store a different type of values in java program.
2. These values will get a specific memory based on the data type specify to this.
3. In Java there are main 2 categories of the Data Type
 - a. **Primitive Data Type**
 - i. Are the values which is in non-object format.
 - ii. These values will have a fixed sized.
 - b. **Non-Primitive Data Type**
 - i. Are the values which is in the form of Object.
 - ii. There is no fix size for the non-primitive data type.



1. **Integer Type**
 - a. In this type you can store a numeric value which is non decimal type.
 - b. You can store the negative and positive values.
2. **Floating type**
 - a. You can store a numeric value which is with or without decimal type.
 - b. You can store negative and positive
3. **Character Type**
 - a. You can store a value with single character.
 - b. In this you can store character, Symbols or also numeric values which must be a positive.
 - c. The symbol and the char value must be store inside single quotes (‘ ’)
4. **Boolean type**
 - a. You can store a **true or false** value only

Data Type Memory Footprints



Range for the numeric values

Formula to get the range of value

$$- 2^{n-1} \text{ to } 2^{n-1} - 1$$

Here, n is the number of bits.

byte : - 2^{8-1} to $2^{8-1} - 1$
-128 to 127

Rules to store Values in Data Type

1. long:
 - a. the values should be suffix with l or L.
2. float
 - a. The value must be suffix with f or F.
3. char
 - a. char values must be in single quotes.
 - b. You can store symbols or numeric values also along with alphabets.
 - c. You can store all the ASCII values inside char.
 - d. The numeric values in char must not be negative.

Variables

1. Variables are use to store values/data into a program.
2. Variables are use to print the values as an output.
3. Variable is use for a mathematical calculation.
4. Variables are use to assign value to another variable
5. Syntax:

Data_type identifier; // Variable declaration
identifier = value; // Initialization of variable.

OR

Data_type identifier = value;

Type of variable

1. There are 3 types of variable
 - a. Local Variable
 - i. Variable which is created inside method or inside method argument is called as Local Variable.
 - ii. Local variable can be access within a method only it cannot be used outside method.
 - iii. Local variable has initialized before use.
 - b. Instance variable
 - i. Variable which is created inside class and outside any method is called instance variable.
 - ii. This variable is accessible inside class and also outside class using Object.
 - iii. Instance variable can be initialized with default value if you have not provided any value explicitly.
 - c. Static/class variable
 - i. Variables which are created outside any method but will present inside class with static keyword.
 - ii. These variables are accessible inside class and also into another class using Class name.
 - iii. Static variable can be initialized with default value if you have not provided any value explicitly.

```
public class VaraibleType
{
    int b = 20;    // Instance Variable
    static int c; // static/class varaivle
    public static void main(String args[])
    {
        int a = 10; // local Variable
        System.out.println(a);
        System.out.println(c);
    }
}
```

Default values for variable

Data Type	Default Value
byte short int long	0
float double	0.0
char	0 or '\u0000'
boolean	false
Non-primitive	null

Note: These default values are only applicable to a instance and Static variables

Primitive Variable Casting

1. Casting is a process in which one data type of values can be converted into another data type.
2. There are 2 types of casting
 - a. **Implicit Casting**
 - i. The casting which is done by java automatically and internally without adding any extra code.
 - b. **Explicit Casting**
 - i. The casting will not happen by java, for this developer has to add some extra code in the program.

```
public class VariableCasting
{
    public static void main(String ar[])
    {
        int a = 20;
        byte b = (byte) a; // explicit casting
        long c = a; // implicit casting

        System.out.println(a);
        System.out.println(b);
        System.out.println(c);
    }
}
```

Operators

1. Operators are use to perform different types of operation.
2. This is use in a math expression or comparison of the values.
3. **Arithmetic Operator**
+, -, /, *, %
4. **Assignment operator**
=, +=, -=, /=, *=
5. **Increment and Decrement Operator**
 - a. You can increment or decrement the original value by 1
 - b. It can be use by 2 ways
 - i. Pre Increment and Decrement
 - ii. Post Increment and Decrement
++, --
6. **Relational Operator** (It always return the boolean values, that is why it is use to write a logical/boolean/conditional expression)
<, <=, >, >=, ==, !=
7. **Logical Operator** (It can use for 2 numeric value or 2 logical expressions)
&, |, !

Expression: 5 & 15	Expression: 5 15
Solution	Solution
0 1 0 1	0 1 0 1
& 1 1 1 1	1 1 1 1
-----	-----
0 1 0 1 -> 5	1 1 1 1 -> 15
8. **Short Circuit Operation** (It is used to combine a multiple logical expression)
&&, ||
9. **Ternary operator**
? (if), : (else)

The diagram illustrates the ternary operator `int value = (a >= 1) ? 1 : 0;`. The condition `(a >= 1)` is enclosed in a red box and labeled "condition" with a red arrow. The question mark `?` is enclosed in a green box and labeled "if" with a green arrow. The value `1` is enclosed in a green box and labeled "true" with a green arrow. The colon `:` is enclosed in a green box and labeled "else" with a green arrow. The value `0` is enclosed in a green box and labeled "false" with a green arrow. Blue curved arrows show the flow: from the condition to the "if" value (1) if true, and from the condition to the "else" value (0) if false.

```
int value = (a >= 1) ? 1 : 0;
```

condition if else

Control Flow Statement

1. Using control flow statement you can control the execution flow of the program.
2. There is different type of control flow statements
 - a. Sequential statement
 - i. By Default program executes in a sequence.
 - b. Conditional Statement
 - i. Based on specific scenario you can decide whether to execute the statement or block of statement or not.
 - ii. It can be achieved by multiple ways such as if conditional statement and its variation, switch cases.
 - c. Looping Statement
 - i. Looping statement you can execute a statement or block of statement multiple time.
 - ii. It can be achieved by multiple ways such as while, do-while, for, enhance for.

If Condition and its variations

If condition:

It is used to execute a statement or block of statement based on the condition/scenario.

Syntax:

```
if(Boolean/conditional expression)
{
    Statement(s)
}
```

If-else condition

1. It use to execute the statement based on condition trye and also you can set up a statement if the condition is false.
2. Else has to followed with if part. You cannot use only else part.
3. Syntax:

```
if(conditional Expression)
{
    Statement(s)
}
else
{
    Statement(s)
}
```

Else-if condition

1. Its use to execute the else statement based on condition. It is also known as else if ladder.
2. Here, you can also set the condition check for your else part.
3. In this structure if any of the expression is true then it will not execute the rest of the expressions.

4. Syntax:

```
if(Condition expression)
{
    Statement(s)
}
else if(Condition Expression)
{
    Statement(s)
}
else if(Condition Expression)
{
    Statement(s)
}
else
{
    Statement(s)
}
```

Nested if cases

1. You can create one if into another if.
2. The once structure can be created inside another structure.
3. Syntax:

```
if(condition expression)
{
    if(condition expression)
    {
        Statement(s)
    }
}
```

Task-1

Create an int variable age

Age between 1-12 – Kids Age

Age Between 13-18 – Teen Age

Age Between 19-60 – Adult Age

Age Between 61-110 – Old Age

And age other than this – Invalid Age

Switch Case

1. This cases will be used in a scenario where you are needs to compare the exact match for the value.

2. Syntax:

```
switch(value)
{
    case label:
        statement(s)
        break;
    case label:
        statement(s)
        break;
    case label:
        statement(s)
        break;
    default:
        statement(s)
}
```

3. Rules to use switch case

- a. The value inside switch must be **byte, short, int, char, String, enum** type only.
- b. All the labels must be unique.
- c. The data type of value and the label has to match.
- d. Break statement is not mandatory but if you don't provide it, then it will generated an unexpected result.
- e. If multiple cases has same execution then you can combine the cases.

Task-1 (Using switch case)

Create an int variable day and assign a value for day variable

day=1, 2, 3, 4, 5 => It's a Working Day

day=6, 7 => It's a Weekend

any other value = Invalid Day..

Task-2

Create a char variable month and assign a char value

'J' or 'j' – January or June or July.

'F' or 'f' – Feb.

'M' or 'm' – March or May.

'A' or 'a' – April or Aug.

'S' or 's' – Sept.

'N' or 'n' – Nov.

'D' or 'd' – dec.

'O' or 'o' – Oct.

Other than this = Invalid month Character

Looping Statement

1. Is use to execute a statement or block of statement multiple time.
2. There are different ways to achieve a looping statement

a. While

- i. You can execute the statement(s) multiple time.
- ii. This is also known as pre condition check.
- iii. Syntax:

```
Variable declaration and initialization
while(boolean/conditional expression)
{
    Statement(s)
    Increment/Decrement of variable
}
```

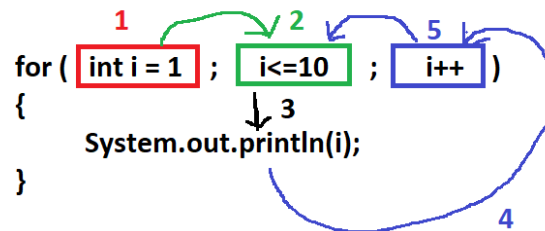
b. Do-while

- i. You can execute the statement(s) multiple time.
- ii. This is also known as Post condition check.
- iii. Syntax:

```
Variable declaration and initialization
do
{
    Statement(s)
    Increment/Decrement of variable
}
while(boolean/conditional expression);
```

c. For

- i. Using for loop you can execute the statement or group of statements multiple time.
- ii. Syntax:
for(declaration/initialization ; conditional expression;Increment/Decrement/statement)
{
 Statement(s)
}



d. Enhance for

i. It is use to iterate the value of collection one by one.

ii. Syntax:

```
for( Variable declaration : collection )
{
    Statement(s)
}
```

f. Nested Loop

i. Nested Loop is use to work with a row and column structure.

ii. Nested loop is a way using which you can write a one loop inside another.

iii. Syntax

```
for( ; ; ) // Used for a ROW
{
    for( ; ; ) // Used for a Column
    {
    }
}
```

Task-1 (Using do-while)

Create a variable num of type int assign a value to it and print the table for the number.

Example: if num=5

Then Ouput: 5
10
15
.
.
50

Task-2 (Using While)

Print all the odd values between 1 – 50

Output: 1

3
5
7
.
.
49

Array

1. Array is a group of similar type of values.
2. Array is use for same type of values having fixed in size.
3. Array is use to store multiple values in a single variable which will be easier to manage and maintain.
4. Array is an indexed basis.
5. Index is a specific position where the values are store.
6. Index always start from 0.
7. Index are always managed by programming language internally.
8. Indexes are use to set and get the values form array.
9. If you handle the index incorrectly then you will get an `java.lang.ArrayIndexOutOfBoundsException`
10. There are different types of array
 - a. 1-Dimentional Array
 - b. 2-Dimentional Array
 - c. Multi-Dimentional/Jagged Array.
11. To create and use array you have to follow 3 syntax
 - a. Array Declaration
 - b. Array Instantiation (Object creation)
 - c. Array Initialization
12. **1-Dimentional (1-D) Array**
 - a. You can store the values in the form of row.
 - b. To crate and use 1-D array you have to follow the following steps
 - i. Array Declaration
 1. Here you can declare the array which is going to create in a program.
 2. Syntax:
`DataType identifier[];`
 - ii. Array Instantiation (Object creation)
 1. In this stage the memory for an array will be created and inside memory you can store the values.
 2. In this stage you have to specify the number of values you wanted to store inside array.
 3. Syntax:
`Identifier = new DataType[Size];`
 - iii. Array Initialization
 1. In this stage you can provide a custom value inside array.
 2. To set the custom value you have to use a index.
 3. Syntax:
`Identifier[index] = value;`

```
double percent[]; // Declaration of Array  
percent = new double[10]; //instantiation of array
```

	0	1	2	3	4	5	6	7	8	9
index										
value	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

↑
64 bits

Different ways to create 1-D array

1. There are 4 ways using which you can create array.

Way-1

```
double percent[]; // Declaration of Array
percent = new double[10]; // Instantiation of array
percent[0] = 77.88; //initialization of array
percent[1] = 72.81;
percent[2] = 62.11;
percent[3] = 72.51;
percent[4] = 45.81;
percent[5] = 76.41;
percent[6] = 65.22;
percent[7] = 89.81;
percent[8] = 55.90;
percent[9] = 67.31;
```

Way-2

```
double percent[] = new double[10]; // Declaration and Instantiation of Array
percent[0] = 77.88; //initialization of array
percent[1] = 72.81;
percent[2] = 62.11;
percent[3] = 72.51;
percent[4] = 45.81;
percent[5] = 76.41;
percent[6] = 65.22;
percent[7] = 89.81;
percent[8] = 55.90;
percent[9] = 67.31;
```

Way-3

```
double percent[] = new double[] {77.88, 72.81, 62.11, 72.51, 45.81, 76.41, 76.41, 65.22};
```

Way-4

```
double percent[] = {77.88, 72.81, 62.11, 72.51, 45.81, 76.41, 76.41, 65.22};
```

Length Function:

1. Using length function of array you can get the total number of values present inside array.
2. Syntax:
array.length
3. You can also find the last index using length function
Last index = array.length – 1

Task-1

Store a percentage for one student (10th, 12th, Higher education semester wise) and print the average of the percent.

2-Dimensional (2-D) Array

1. In 2-D array you store the data in the form of row and column which is also known as Metrix.
2. Syntax
 - i. Array Declaration
 1. Here you can declare the array which is going to create in a program.
 2. Syntax:
`DataType identifier[][];`
 - ii. Array Instantiation (Object creation)
 1. In this stage the memory for an array will be created and inside memory you can store the values.
 2. In this stage you have to specify the number of values you wanted to store inside array.
 3. Syntax:
`Identifier = new DataType[Row-Size][Column-Size];`
 - iii. Array Initialization
 1. In this stage you can provide a custom value inside array.
 2. To set the custom value you have to use a index.
 3. Syntax:
`Identifier[row-index] [column-index] = value;`

```
double percent[][]; // Declaraing Array
```

```
percent = new double[3][5];
```

```
percent[0][2] = 77.12;  
percent[2][3] = 80.01;
```

	column				
	0	1	2	3	4
0	0.0	0.0	77.12	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	80.01	0.0

Length Function in 2-D Array

1. `Array.length`: It will return a total number of rows.
2. `Array[row-index].length`: It will return a total number of columns for the specified row.

Different way to create 2-D array

Way-1

```
double percent[][]; // Declaring Array
percent = new double[3][5]; // Instantiation of array
percent[0][0] = 67.65; // Initialization
percent[0][1] = 87.15;
percent[0][2] = 61.45;
```

Way-2

```
double percent[][] = new double[3][5]; // Declaring and Instantiation of Array
percent[0][0] = 67.65; // Initialization
percent[0][1] = 87.15;
percent[0][2] = 61.45;
```

Way-3

```
double percent[][] = new double[][]{
    {76, 78, 45, 78, 56},
    {76, 89, 56, 98, 56},
    {88, 66, 89, 98, 67}
};
```

Way-4

```
double percent[][] = {
    {76, 78, 45, 78, 56},
    {76, 89, 56, 98, 56},
    {88, 66, 89, 98, 67}
};
```

Task-1

Create an array which can hold the 5 students 4 subject marks.

Print the Marks of every student in table format along with the percent.

Also Print the Grade of the Student.

And Also print the Highest Percentage.

Multi-dimenational Array

1. This array is also based of the row and column structure.
2. It is used to store the values which has fixed and variable number of column.
3. Its is also known as **Jagged array**.
4. Example:

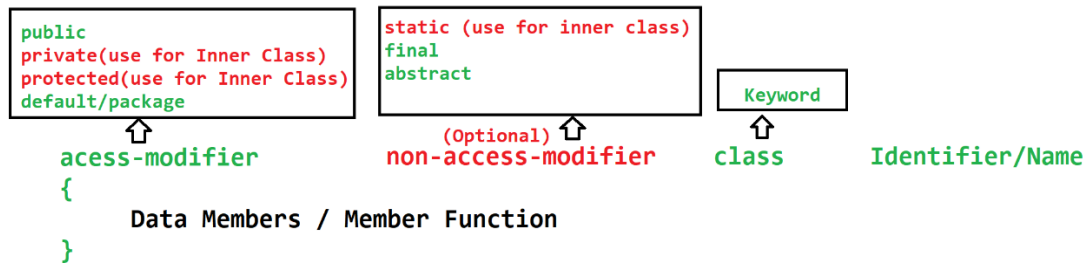
```
int array[][] = {  
    {34, 56, 768},  
    {23, 45},  
    {54, 67, 89, 34, 768, 23},  
    {32, 54, 76}  
};
```

row \ column							
0	34	56	768	3 column			
1	23	45	2 column				
2	54	67	89	34	768	23	6 column
3	32	54	76	3 column			

Class, Object and Method

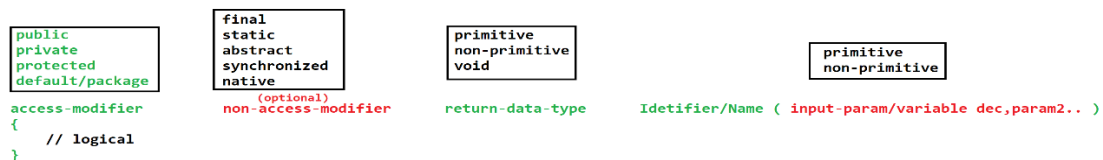
Class

1. Class is a collection of state/variables/data member and behavior/methods/member function.
2. Along with variables and methods you can also create a constructure or another class(Nested class) in a class.
3. The data member and member function of class can be used outside class using the Object.
4. Syntax to create class



Method

1. Method is a collection of variables and the executable statement.
2. Methods are used to write a logical code. The code written inside methods can be reuse from the different location of the program/project.
3. Method is also help to reduce the complexity of the program by dividing a logic into a smaller chunk or code.
4. To execute the logic from the method, methods has to execute manually by calling them directly or by using Object.
5. Methods can accept the values which is known as **input parameter**. There can be more than one parameter. The values which is passed to parameters are called **arguments**.
6. Method can return the output of the execution as a **return type**. There can be onlt single return type.
7. Types of methods
 - a. Build-In Method (Predefine) Method
 - i. The methods which are provided by language/framework.
 - b. Custom method (User define) Method
 - i. The methods created programmatically.
8. Syntax:



Object

1. Object is a representation of the class.
2. Using an object you can access the properties of the class (variables and methods).
3. To access the properties of the class you have to use dot(.) operator.
4. In Java you can create Object using **new** operator
5. Syntax

ClassName Name; // Reference variable

Name = **new** ClassName(); // Object of class

Object Constructor Call

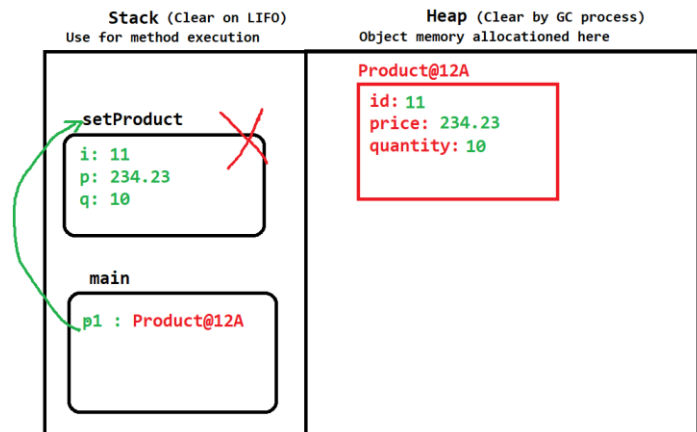
Object Name : is a reference of class and also a object of class

OR

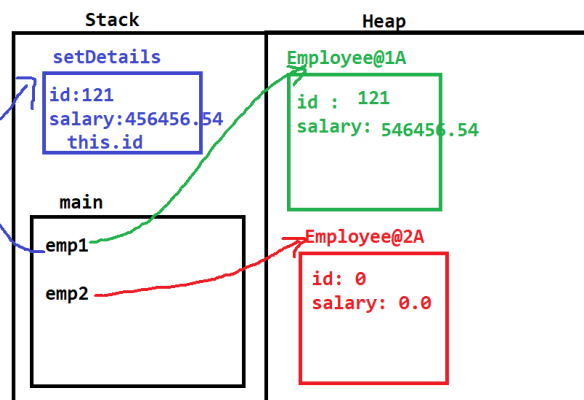
Calculation calculate = new Calculation();

```
public class ProductDetails
{
    public static void main(String args[])
    {
        Product p1 = new Product();
        System.out.println(p1);
        p1.setProduct(11, 234.23, 10);
        System.out.println(p1.id);
        System.out.println(p1.price);
        System.out.println(p1.quantity);
    }
}
class Product
{
    int id;
    double price;
    int quantity;

    public void setProduct(int i, double p, int q) {
        id = i;
        price = p;
        quantity = q;
    }
}
```



```
public class EmployeeDetails {
    public static void main(String a[]) {
        ✓ Employee emp1 = new Employee();
        ✓ Employee emp2 = new Employee();
        ✓ emp1.setDetails(121, 456456.54);
        ✓ System.out.println(emp2.id);
        ✓ System.out.println(emp2.salary);
    }
}
class Employee{
    int id;
    double salary;
    public void setDetails(int id, double salary) {
        this.id=id;
        this.salary=salary;
    }
}
```



Accept User Input in Java

1. You can accept the values from the user using console or UI.
2. Here, we will see how to accept values from console.
3. There are multiple ways and classes provided by java to accept the user input.
 - a. Command Line Argument
 - b. Scanner Class
 - c. Console Class
 - d. Buffer Classes
4. Command Line Argument
 - a. This is use to accept the user input before the execution of program.
 - b. Command Line argument are the values which is provided while tying the command for java program execution.
 - c. These arguments (values) will be received into a java program inside main method and the String array.
 - d. All the arguments (values) will be in the string format only.
 - e. You have to convert it into specific format manually.

```
E:\Batches\IntuTech\FSD-22Nov22\code\corejava\basic>javac CommandLineArg.java  
  
E:\Batches\IntuTech\FSD-22Nov22\code\corejava\basic>java CommandLineArg Abc 20 21 2.2  
Command Line Argument Example...  
Abc  
43.2
```

5. Scanner Class
 - a. Scanner class is use to accept the user input in the middle of program execution.
 - b. You can accept input into any data type without any manual conversion.
 - c. Scanner class is the build-in class by java
 - d. This class is present inside **java.util** package.
 - e. To use this class you have to import it into java program and also create an object of the class.
 - f. In the scanner class you can use different methods to accept the data of different type.
 - g. There is nextXXX(), is use to accept the different type of values.
 - h. You can use next() method to accept a string type of values.

Wrapper Class

Primitive Data Type	Wrapper Classes
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
char	Character
boolean	Boolean

String in Java

1. String is an array of character.
2. String is use to store a multiple character or combination of symbol, Numbers and characters.
3. In java string values always provides into double quotes ("Value")
4. To Store String in java you can use following build-class
 - a. String
 - b. StringBuffer
 - c. StringBuilder

String Class

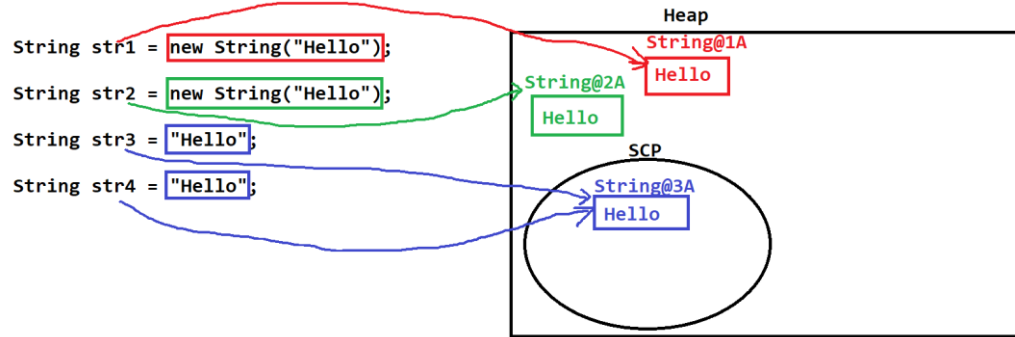
1. String class is used to store an array of character.
2. String class is a build-in class. This class has multiple methods to perform the operations on the string values.
3. String class is present inside **java.lang** package.
4. String class is a final class.
5. To Use String class you have to create object of String class, which can be achieve by 2 ways.
 - a. With new Operator
String str = new String("Value");
 - b. Without new operator
String str = "Value";
6. Memory foot prints of string
7. String **Objects are immutable**. Immutable means the original value of the string remain same (Does not change) even after using any of the method of the string class.

```
String str1 = new String("Hello");
```

index	0	1	2	3	4
value	H	e	l	l	o

Array of char

8. The string Object will be store inside Heap always, and there is a SCP memory allocation for the string which created without new operator.
9. SCP
 - a. SCP is a String Constant Pool
 - b. The String value which is created without new operator are store inside SCP.
 - c. While creating object (Storing data) inside SCP, it will first check if same object is already present or not.
 - d. If Object with same value is present then, no new object will be created the same will be return to a new reference.
 - e. If no same value object present then only new object will be created inside SCP.



Task-1

Create a String variable with value **"Welcome"** and print the String in Reverse way using String class
Output : **emocleW** (Hint: use `toCharArray()` Method)

```
String str = "Welcome"
```

0	1	2	3	4	5	6
W	e	l	c	o	m	e

```
str.toCharArray();
```

Task-2

Create a String variable with value **"FirstName MiddleName LastName"** and print the output in to following way

Output : **LastName MiddleName FirstName** (Hint: use `split()` Method)

StringBuilder

1. This class is use to store a string value.
2. This is the java build-in class.
3. This class is present inside **java.lang** package.
4. This class is a final class.
5. By Creating Object of this class you use the functionalities of the class
StringBuilder builder = new StringBuilder("<Value>");
6. There are multiple methods inside string class using which you can perform operation on existing string.
7. Here the SCP concept is not applicable.
8. The object of this class is **mutable**. The Original Value will be change after implementation the methods of this class.

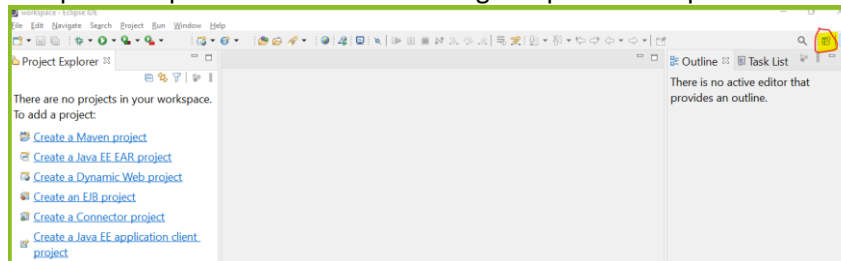
StringBuffer

1. This class is use to store a string value.
2. This is the java build-in class.
3. This class is present inside **java.lang** package.
4. This class is a final class.
5. By Creating Object of this class you use the functionalities of the class
StringBuffer builder = new StringBuffer("<Value>");
6. There are multiple methods inside string class using which you can perform operation on existing string.
7. Here the SCP concept is not applicable.

8. The object of this class is **mutable**. The Original Value will be change after implementation the methods of this class.
9. StringBuffer Objects is **thread safe**. At a time only one thread can perform the operation on StringBuffer object.
10. The methods of StringBuffer are **synchronized**.
11. It is slower in the performance than StringBuilder.

Eclipse IDE (Integrated Development Environment)

1. Download Eclipse .zip file.
https://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/2022-12/R/eclipse-je-2022-12-R-win32-x86_64.zip
2. Extract a ZIP file into appropriate location
3. Click on the eclipse application file.
4. Setup a Workspace
 - a. Create a Location where you wanted to create all your projects, and java files.
5. Set the Perspective of the eclipse
 - a. Perspective option is available on the right top of the eclipse window



- b. Select a “Java” option from list and click on “Open” button

Object Oriented Programing (OOPs)

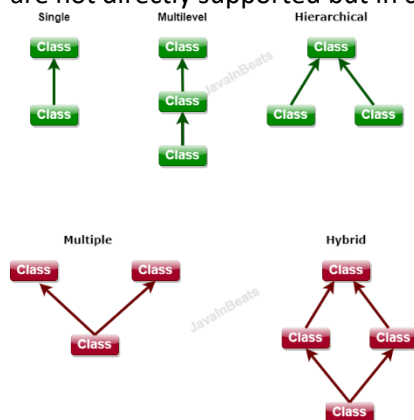
1. Base of OOPs concept is class and Object
2. There are 4 main concepts of OOPs
 - a. Encapsulation
 - i. It is a wrapping of data member(variable) and member function(method) into single unit.
 - b. Inheritance
 - i. Inheriting the property (Variables and methods) of one class into another class which is also known as Parent and child.
 - c. Polymorphism
 - i. One object can be behave in a multiple form.
 - d. Abstraction
 - i. Hiding the complexity and display only the important details/function is called abstraction.

Encapsulation

1. Encapsulation is wrapping of data member (variable) and member function (method) into single unit.
2. Class is also a kind of encapsulation.
3. As per the recommendation, do not access the instance variable of one class directly into another class using object, instead use the methods to access them.
4. In Encapsulation you can hide the data using Private access modifier and provide them access using a getter and setter methods.
5. Setter Methods
 - a. Setter methods are used to set the values for the instance variable.
 - b. This method mostly starts with “set” word as a prefix and will be followed by variable name.
 - c. This method accepts the value which needs to be assign to an instance variable and not return anything.
6. Getter methods
 - a. Getter Methods is used to get the values of the variable.
 - b. This method mostly starts with “get” word as a prefix followed with variable name.
 - c. This method will not accept anything but return the values.
7. Advantage of Encapsulation
 - a. Can Achieve the data hiding.
 - b. Can control who can access what.
 - c. Can achieve the loose coupling.

Inheritance

1. Access the properties of one class into another class. Inheritance will create a parent and child relation between two classes.
2. The parent and child relation are also known as **IS-A relation** in java.
3. Using inheritance, you can access the non-private variables and methods of the parent class into child class.
4. To achieve inheritance, you can use **extends keyword** in java.
5. In OOPs there are 5 types of inheritance, but in java only 3 are directly supported and other 2 are not directly supported but in can be achieve by interface.



6. In java Single, Multilevel and Hierarchical inheritance is directly supported and Multiple and Hybrid is not directly supported in java but it can be achieve by using interface.
7. In Java One class cannot extends (inherit) more than one class at a time.

8. Parent class is also known as **super class or Base class** and child class is also known as **Sub class or derived class**.
9. **Advantages of Inheritance**
 - a. Reusability
 - b. Extensibility
 - c. Inheritance is required to achieve runtime polymorphism.

Object Class in Java

1. Object class is a super (parent) class of all the java classes.
2. If class do not have any parent class then java will add a Object class as a parent class.
3. Every Java class will have the properties of Object.
4. Object class is use to define all the common properties which are required inside all the java classes.
5. There are multiple methods present inside this class, which is as follows
 - a. toString()
 - b. equals()
 - c. hashCode()
 - d. wait(), wait(long), wait(long,int)
 - e. notify()
 - f. notifyAll()
 - g. finalized()
 - h. getClass()