**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 Programing 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.

Spring Boot Framework

**Database**

SQL, MYSQL Database

Tools (**Dev and DevOps Tool**)

**Git**

**GitHub**

**Maven**

Postman

Swagger

Software

JDK (JDK-11)

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/>

**Logical Program Solving:**  
<https://www.hackerearth.com/>

<https://www.hackerrank.com/>

**Java Setup**

1. Download JDK

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

Download the Installer for you operating System.

1. Install JDK
2. Verify the Installation
   1. You can verify the installation file into C:\Program File\java\<jdk-version-folder>



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



* 1. **Set Path**
     1. Check for “path” variable inside ‘system variables’ section
     2. Select “Path” variable and click on “Edit” Button
     3. Click the “New” button on the new window
     4. And set the variable value as follows

**%JAVA\_HOME%\bin**

* + 1. Click on “OK”



1. Verify the Environment Variable Setup
   1. Open CMD
   2. Try the following commands
      1. **java -version**
      2. **javac**



**Type of Application**

1. Desktop Application
2. Web Application
3. Mobile Application
4. Embedded Application
5. Console Application

**Java Introduction**

1. Java Editions
   1. JSE
      1. Java Standard Edition
      2. Also known as Core Java
      3. It is use to develop the console based app and desktop application
   2. JEE
      1. Java Enterprise Edition
      2. Also known as Advance Java
      3. It is use to create Web Application
   3. JME
      1. Java Micro Edition
      2. It is use to create Mobile and embedded application

**Writing First Java Program and Execution of the program**

1. Open Notepad
2. Write a Java Program
   1. Create a Class
      1. Everything in java must be in a class except import and package statement.
      2. Syntax for Class

**public class <className>**

**{**

**}**

* 1. Create a main method
     1. At the time of execution of the program java search for main method internally.
     2. This method is a start point of all java program.
     3. The main purpose of method is use to write a logic and execute it to produce an output
     4. Syntax:

**public static void main(String args[])**

**{**

**}**

* 1. Write an executable statement
     1. These statement executes and produce an output
     2. To print the output you can use following line of code

**System.out.println(“Message”);**

1. Save the file
   1. File Name must be same as public class Name
   2. File extension must be .java.
2. Open a CMD to compile and execute the java program
   1. Open a command prompt from a location where you save your java file.
      1. Go to a location where you save your java file.
      2. Type ‘CMD’ in the address bar
      3. Hit Enter.
3. Compile the Java code
   1. To Compile A java code use following command

**javac <filename with extension>**

* 1. As a outcome of the compilation you will get a .class file.

1. Execute The Java Code
   1. To Execute use following command

**java <ClassName>**



**Rules**

1. You can create multiple java classes in a file but only one class must be public and you java file name must be save as a public class name.
2. The .class file will be generated after compilation of the java file. The .class file is generated for the java classes created inside the file. Every java class will have a separate .class file.
3. It is not mandatory to provide the main method inside java class. but if you are trying to execute the java class from JVM then you class must have a main method.
4. The main method internally called by the JVM at the time of your class execution.

**Important components**

1. JVM
   1. Java Virtual Machine
   2. It is use to execute the java program.
   3. The memory required from the program execution will be provided by JVM.
   4. Heap and Stack is the main memories which is allocated by JVM.
2. JRE
   1. Java Runtime Environment.
   2. JRE is a combination of the JVM and the APIs.
   3. It help you to create a runtime environment for the application.
   4. This has be present on the client system.
3. APIs
   1. Application Programming Interface.
   2. APIs are the predefine class, interfaces or functionalities by the programming language.
   3. These functionalities you can use to achieve you purpose.
4. JDK
   1. Java Development Kit
   2. It help us to develop and execute the java program, It is combination of Dev. Tools, JRE, JVM, APIs.
   3. It has to be install on developers system.

**Keyword, Identifier, Literals**

**Keyword**

1. Are the words which are reserve by the java programing language.
2. These words cannot used for you own purpose.
3. There are total 52 words are serve by java.
4. All keyword are in small case only.
5. Example:

public, class, static void, if, else, do, for, int, byte, short, long, char, float, double, boolean, case, break, continue etc.

**Identifier**

1. The words which are used by the developer, these words are use to identify the java components.
2. The words which are used to create class, method, object, variable name are called keywords.
3. Using Identifier, you can provide the name which can be further use for the reference.
4. Rules to create identifier
   1. Identifier must not be a keyword.
   2. Identifier can contains alphabets, symbol, number.
   3. There are only 2 symbols are allowed \_ and $
   4. The identifier must start with alphabets or symbol and it must not start with number.
   5. Identifier cannot contain spaces.
   6. There is no character limit to create identifier.
   7. Java is Case sensitive and hance you have to use it in a same way in which you declare.
5. Conventions to use Identifier
   1. Identifier for class
      1. The name should be start with capital case.
      2. If there is a combination of multiple words then every word should be start with capital.
      3. Example: Welcome, System, String, EmployeeDetails
   2. Convention to create Method, variable and object
      1. The name should be start with small case.
      2. If there is more than one word then 2nd word onward it should start with capital case.
      3. Example: main, println, args, out, printDetails
   3. Convention for Constants
      1. The name of the constants should in capital case.
      2. If it is a combination of multiple words then it should be separated by ‘\_’
      3. Example: PI, GRAVITY, COMPANY\_NAME

**Literal**

1. Literals are the values, mostly string values are also consider as a literal.
2. The values which is reserve by java is also known are literals. These values are also consider as keyword.
3. Example: null, true, false.

**Comments in Java**

1. Comments are use to add code level documentation, to skip the specific line of code from the execution.
2. There are 3 types of comments in java
   1. Single Line Comment
      1. Syntax

// line to comment.

* 1. Multi-line comment
     1. Syntax

/\*

Lines to comment

\*/

* 1. Documentation comment
     1. This comment is use to write a code level documentation.
     2. The commented lines will be included inside the .class file after compilation.
     3. Syntax:

/\*\*

Comments

\*/

**Data Type and variable**

**Operator**

1. Arithmetic Operator

+, -, \*, /, %

1. Assignment Operator

=, +=, -=, /=, \*=

1. Increment and Decrement Operator

++, --

1. Relational Operator
   1. returns only Boolean values
   2. It use to create a conditional/logical expression (decision making statement)

>, >=, <, <=, ==, !=

1. Logical Operator
   1. Logical operator can return numeric value or Boolean values.
   2. It can be use with the numeric value or to combine the 2 or more conditional expression.
   3. It can be use to perform operation on Bit level.

& , |, !

1. Short Circuit Operator
   1. Is use to combine the 2 or more conditional expression.
   2. It cannot be use for a numeric value bit level operations.

&&, ||

1. Ternary Operator
   1. It is use to execute the operations based on the conditional expression
   2. Here ‘?’ is consider as If and ‘:’ is consider as else.
   3. Example :

(salary<25000) ? 200\*10 : 200\*5 ;

If \_\_\_\_\_ else \_\_\_\_\_

?, :

**Control Flow Statement**

1. It is use to control the execution flow of the program.
2. By default the program executes sequentially.
3. Using control flow statement you can decide when how many time the statement should gets executes.
4. This can achieve by using 2 ways
   1. Conditional Statement
      1. This is use to execute a statement or block of statement base on some scenario or condition.
      2. There are different options to execute statement based on condition

**If condition** and its variation and **switch case**.

* 1. Loop Statement
     1. This is use to execute the statement or block of statement multiple time.
     2. Different options to execute statement multiple times

**While, do-while, For loop, nested loop and enhance for(for-each)**

**If Conditional Statement**

1. Using this you can execute the statement based on the condition.
2. There are different variations to achieve this statement
3. **If Condition**

Syntax:

if(conditional Expression)

{

Statement(s)

}

1. **If-else**

Syntax:

if(conditional Expression)

{

Statement(s)

}

else

{

Statement(s)

}

1. **Else-if ladder**

Syntax:

if(conditional Expression)

{

Statement(s)

}

else if(conditional Expression)

{

Statement(s)

}

else if(conditional Expression)

{

Statement(s)

}

else

{

Statement(s)

}  
**Note: In this case only one condition gets executes. It there are multiple conditions satisfies but it will execute only the first condition block.**

1. **Nested if**

You can create one if structure into another structure which is called nesting.

Syntax:

if(condition) {

if(condition) {

}

}

Task:

1. Create a two int variable and print the grater number from them.

Example: int a =20, int b = 30

O/P : **The Grater Number is 30**

1. WAP to print the vaccination slot based on age

If age is between 60-120 “You are in Slot-1”

If age is between 30-60 “You are in Slot-2”

If age is between 18-30 “You are in Slot-3”

If age is between 1-18 “You are in Slot-4”

Any other age values than this print “Invalid Age…”

1. WAP to print the Age Group

If age is between 60-120 “Old Age”

If age is between 22-60 “Adult Age”

If age is between 12-22 “Teen Age”

If age is between 0.1-12 “Kids Age”

Any other value “Invalid Age”

Task for 22-March

1. WAP to print a day type

If value of day is 1,2,3,4,5 then print “Working Days”

6, 7 then print “This is Weekend”

**While and Do-While**

1. WAP to print the table of the given number using while loop

Example : number = 5

O/P 5

10

15

20

.

.

.

50

1. Print all the odd numbers from the 1 – 50 using a do-while loop.

Hint : if number%2 != 0 then the number is Odd

O/P

1

3

5

.

.

.

49

**Nested Looping**

1. You can create one loop inside another loop.
2. It is mostly used for a scenario where you wanted handle the table structure data.
3. That is you can perform the operation on row and columns using nested loop.
4. Syntax

for( ; ; ) // Row

{

for( ; ; ) // column

{

}

}

Task 24-March: WAP to print following pattern

\*

\* \* \*

\* \* \* \* \*

\* \* \* \* \* \* \*

1

12

123

1234

12345

**Enhance For Loop**

1. It is mainly used to get the values one by one from the collection/group of values which is also known as iterating collection.
2. Syntax:

for(datatype identifier : collection )

{

Statement(s)

}

**Array**

**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 only single return type.
7. Types of methods
   1. Build-In Method (Predefine) Method
      1. The methods which are provided by language/framework.
   2. Custom method (User define) Method
      1. 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



**Java Memory**

1. Stack:
   1. This memory is allocated for the method execution.
   2. The data created inside method will be allocated in this memory.
   3. This is based in LIFO (Last In First Out) structure.
   4. This memory keep on clearing as an when the method execution is completed.
2. Heap:
   1. This memory is allocated for a objects.
   2. The object creation happens inside this memory.
   3. To clear this memory java perform the Garbage Collection (GC) process.
   4. The GC process will be happen after a specific interval.

