**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, User input.

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

**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 (JDK 8, 11, 17)

<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 applications**

1. Mobile applications
2. Web Applications
3. Desktop applications
4. Console based app
5. Embedded applications

**Java Editions**

1. JSE
   1. Java Standard Edition
   2. Is also known as Core Java
   3. Can develop console based and desktop application.
2. JEE
   1. Java Enterprise Edition
   2. Is also known as Advance Java
   3. Can develop the web applications.
3. JME
   1. Java Micro Edition
   2. Can develop 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 an 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 your 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 your class must have a main method.
4. The main method internally called by the JVM at the time of your class execution.



**Keyword, Identifier, Literals**

**Keyword**

1. Are the words which are reserve by the java programing language.
2. These words cannot used for your own purpose.
3. There are total 52 words are reserve 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 identifier.
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 hence 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, printEmployeeDetails
   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**

Task-1

Create a java class where store the Student data and print the student details

Student Data: Student id, Student gender(M/F), pincode, 10th percentage, completed education flag(true\false)

**Operator**

1. **Arithmetic Operator**

+, -, \*, /, %

1. **Assignment Operator**

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

1. **Increment Decrement Operator**

++, --

This operator is use to increment or decrement by 1. There is pre and post increment and decrement.

Pre increment = Operation, Update values and then Assign the value.

Post increment= Assign the value, Operation and update the value.

1. **Relational Operator**

This operator is use to compare the values which is also known as conditional/Logic/Boolean expression. This operator always return the boolean (true or false) output

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

1. **Logical Operator**

Can use this operator for numeric or Boolean values.

It can be use for 2 numeric values on which the bit level operation will be perform.

It is mostly used to combine 2 or more boolean expression

&, |, !

 

1. **Short Circuit logical Operator**

It is used to combine 2 or more conditional expression.

It cannot be use for a numeric value.

&&, ||

1. **Ternary Operator**

It is use to return the value based on the conditional expression.

In this there are 2 symbol is used ? and : here ? is an if condition and : is an else part.

****

**Class, Method and Object**

Class:

1. Class is collection of state/Data member/variable and behaviors/member function/methods.
2. Along with variable and methods you can also create constructor and class(inner class)
3. You can access the data member and member function of one class into another by using object.



Method:

1. Method is a collection of variables and executable statements.
2. Methods are use to write a logical code. this can reuse at multiple locations.
3. Method is use to reduce the code complexity by diving the logic into smaller chunks.
4. Method can be call/access using class object or directly from another method of same class.
5. Methods can accept the values which is known as **input parameters**. There can be one or more input parameters for a methods. The values pass to the parameters are called **arguments**.
6. Method can return the output after execution of logic which is known as **return type** of the method. You can return max one type of value at a time.
7. Type of methods
   1. Build-in(predefine) method
   2. Custom method
8. Syntax



**Object**

1. Object is a representation of class.
2. Using Object you can access the properties (variable & method) of the class.
3. To access the properties of the class using object you have to use dot(.) operator.
4. In java you can create an object using **new** operator (Keyword)



**Java Memory**

1. Stack
   1. This is use to execute the methods.
   2. The data created inside method is store inside this memory.
   3. This memory based on LIFO (Last In First Out)
   4. The memory allocated for the method will get clear once method execution is completed.
2. Heap
   1. The objects get memory allocation inside Heap.
   2. This memory get cleared as a part of Garbage collection (GC) processor



**String in Java**

1. String is an array of character.
2. In Java String will be store in double quotes (“Value”)
3. String is a non-primitive data type.
4. There are multiple class to store string values.
   1. String class
   2. StringBuffer Class
   3. StringBuilder Class

**String class**

1. String is a **build-in class** of java.
2. Using String class you can **store the array of character**.
3. String class is a **final class**.
4. String class present inside **java.lang package.**
5. String objects are **immutable.** Once you assign a value to a string it will never change even after you use the multiple functions of the string class.
6. To define string value you have to create Object of String class.
7. There are 2 ways to get the object of the String class
   1. Without New Operator

**String identifier = “Value”;**

* 1. With New Operator

**String identifier = new String(“Value”);**

**Task**

1. Create a String “Hello Java” and print the output “0lleH avaJ”
2. Create a String “Hello Java” and print the number of occurrences of char ‘a’

**User Input**

1. You can accept the values from the user.
2. To accepts values java provided a predefine.
3. There are multiple ways to accept input from user
   1. **Command line argument**
   2. Console class
   3. **Scanner Class**
   4. Buffer classes

**Command Line Argument**

1. This is use to accept the values from the user before execution of the program.
2. User has to provide values from the execution command.
3. The values which are provide from command line will be received in java program inside main method and in String array.
4. All the values pass by the user is in string format only. It has to convert into appropriate data type in java program.

**Scanner Class**

1. Scanner Class is build-in class.
2. This class is present inside java.util package.
3. Using this class you can accept the data from user at the time of program execution.
4. Scanner class has multiple method to accept the different type values.
5. To use a scanner class you have to create an object of it and using method you can accept values at runtime.

**Eclipse IDE Installation and Setup**

<https://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/2023-12/R/eclipse-jee-2023-12-R-win32-x86_64.zip>

Set the Eclipse perspective to “Java”

Create Java Project into eclipse

1. Go To “File” menu -> “New” Option -> Click on “Java Project” option.
2. Set the Name of the project
3. Unchecked the Module checkbox.
4. Click on Next -> click on Finish

Creating Java File

1. Select and Right click on the “src” folder -> Go To “New” option -> Click on “Class” option
2. Set the java class name
3. Select option if you wanted to create main method.
4. Click on Finish

**Object Oriented Programming (OOPs)**

* + - 1. Encapsulation
         1. Wrapping of data member(Variable) and member function(Methods) into single unit.
         2. Can achieve data hiding.
      2. Inheritance
         1. Acquiring the property of one class/Object into another class.
         2. Can achieve the reusability and extensibility.
      3. Polymorphism
         1. One thing/Object having a multiple forms.
         2. One function can be use/write is multiple ways.
      4. Abstraction
         1. Hiding the complexity from the use and showing required details/functionalities.
         2. Can achieve implementation hiding.