**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

**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.

**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 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 you 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 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 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 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**

++, --

These operators are used to increment and decrement the value by 1.

There is pre and post operation in this operator.

Post Operation: First assign the value then increment the value and update the value

Pre Operation: First increment and update the value then assign the value then

1. **Relational Operator** 
   1. Always returns Boolean value only.
   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**

**Task-10-May-2023**

**Task-1**

WAP to find the vaccination slot

age is between 60-120

Your are in 1stslot of vaccination

age is between 40-60

Your are in 2stslot of vaccination

age is between 18-40

Your are in 3stslot of vaccination

If age is between 1-18

Your are in 4stslot of vaccination

If age not between 1-120

Invalid age

**Task-2**

WAP program to print the Day

Create a char variable and print a Day for it. Using switch case

‘M’ or ‘m’: Monday

‘T’ or ‘t’: Tuesday or Thursday

‘W’ or ‘w’: Wednesday

‘F’ or ‘f’: Friday

‘S’ or ‘s’: Saturday or Sunday

**Array**

**Class, Method and Object**

**Class:**

1. Class is a collection of state/variables/data member and behavior/method/member function.
2. Along with variable and method you can also create constructor and class (Inner class).
3. To access the data member and member function of one class into another class you can use the object.
4. Syntax for Class



**Method**

1. Method is collection of variable and the statements (executable statement).
2. Methods are use to write a logical code. this logic can be reuse from the multiple location.
3. Methods use to divide the complex logic into a smaller chunks.
4. To execute the logic from the method you have to call the method explicitly either by using object or by making a direct call.
5. Methods can accept the values which is knowns as **input parameter**. There can be more than one parameter. The values pass to this parameter is called **argument**.
6. Method can return the output of the logic as a **return type**. You can return one value at a time max from the methods.
7. Type of methods
   1. Build-in (predefine) Method
   2. Custom (user define) Method
8. Syntax for Method



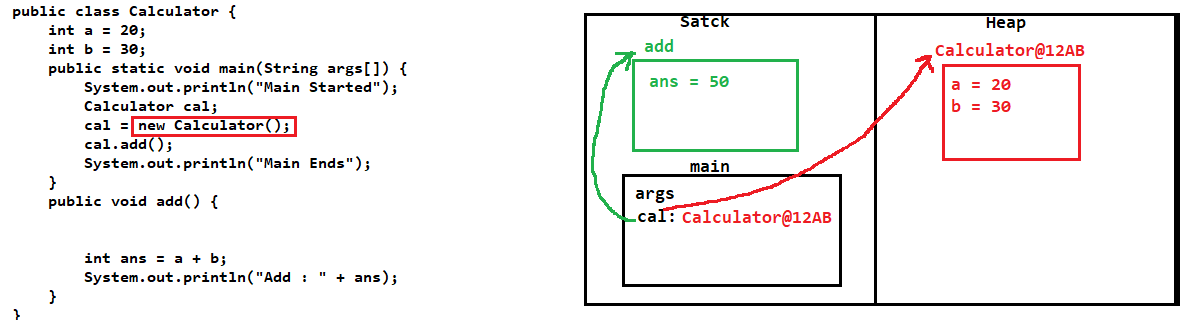
**Object**

1. Object is 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 Object using **new** operator.
5. Syntax

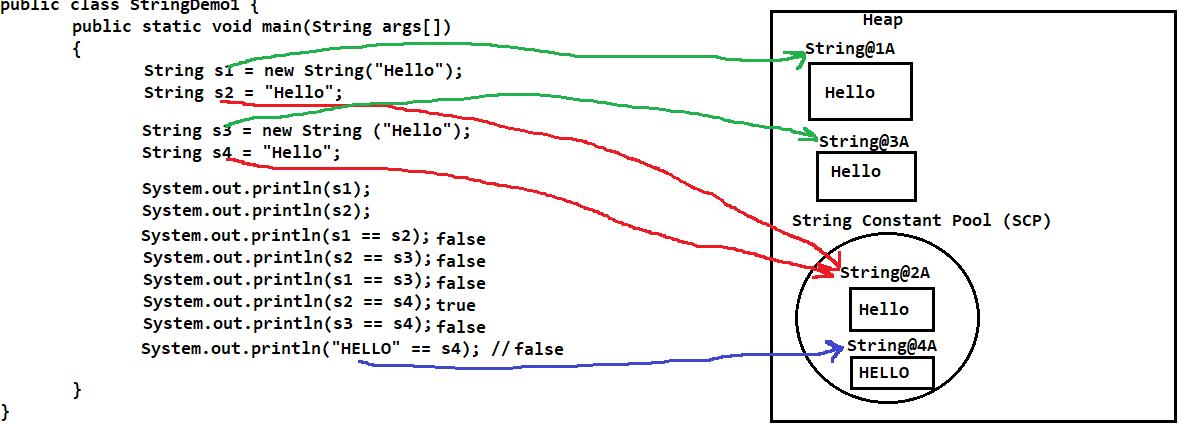


**Java Memory**

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



**String**



**User Input**

1. You can accept the data from the user at the program execution time of before program start up.
2. To accept the input from user you can use following options
   1. Command line argument
   2. Console class
   3. Scanner Class
   4. Buffer Class
3. Command Line Argument
   1. This option is use to accept user data at the start of the program.
   2. The arguments (values) will be provided from the execution command line.
   3. These arguments (values) are for the String array parameter of main method.
   4. Every value will be added in the separate index of the string array.
   5. Value must be provided separated by space and all the value will be received in main method in string format only.
4. Scanner Class
   1. Scanner is a build-in class.
   2. This class is present inside java.util package.
   3. Using this class you can accept the value from the user.
   4. You have to create object of scanner class and use the predefine methods to accept the input from the user.
   5. In this class you will get a specific method to accept the specific type of values.
   6. You can accept the value at the runtime of the program.

**Eclipse Installation and Setup**

<https://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/2023-03/R/eclipse-jee-2023-03-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. Right click on the “src” folder -> Go To “New” option -> Click on “Class” option
2. Set the java class name
3. Select the main method option if you wanted to create main method.
4. Click on Finish

**Object Oriented Programing Concepts (OOPs)**

* + - 1. Java is an Object Oriented Programing language.
      2. Java is not a fully Object oriented because it has primitive data types. In Fully object oriented language you can get everything with class and object.
      3. There are 4 concepts of the OOPs
         1. **Encapsulation**

Wrapping of data member and member function into single unit.

* + - * 1. **Inheritance**

Acquiring the properties of parent class to child class.

* + - * 1. **Polymorphism**

Single thing can be represent in multiple (different) ways.

* + - * 1. **Abstraction**

Hiding the complexity and display only the important functionalities to the user.

**Encapsulation task**

* + - 1. Create a class Employee with id, name, salary, city variable. Create the employee class as encapsulated class. Set and print the employee information from the main class.

**Advantages of Inheritance**

Can achieve the reusability

Can achieve the extensibility

It is used to achieve the Runtime Polymorphism.

**Inheritance Task**

1. WAP to create Calculator class which has 2 numeric variable.
2. Create a child classes Add, Sub, Mul, Dev as a child class of Calculator class and perform the appropriate operation into the class and print the Operation result.

**Object Class**

1. Object class is a build-in class.
2. Object class is a parent of call java class.
3. In this class there are some common properties are created which is accessible in all the java class.
4. If there is no direct parent class of any java class then Object class is a parent for those classes.
5. Common methods of object class
   1. equals()
   2. toString()
   3. hashCode()
   4. getClass()
   5. wait(), wait(int), wait(int,long)
   6. notify()
   7. notifyAll()

**Constructor**

1. Constructor is used to initialize the instance variables at the time of object creation.
2. By default every class has a Constructor.
3. If you do not create any Constructor inside class, then java will provide a default Constructor.
4. If you manually created any constructor inside class, then java will not provide any constructor.
5. Rules to create constructor
   1. Constructor must have a same name as class name.
   2. Constructor do not have a return data type.
   3. Constructor can be created with any access modifier.
   4. There can be multiple constructor in a class which must be created with different parameter list this is also known as constructor overloading.
   5. Constructor always calls at the time of object creation and it can’t be called using class object and dot operator.
6. Every sub class constructor calls super class default or no-param constructor.
7. Every constructor has by default **super()** statement added as a first line.

**Java Keyword (super, this, static, final)**

super keyword

1. Super Keyword is use to access the properties of super class.
2. You can access the super class variable, method, and constructor.
3. Access Constructor using super keyword
   1. Sub class constructor can only access the super class constructor.
   2. By Default super keyword will be added as a first line of the every constructor.
   3. One constructor can only call another constructor and that must be a first line.

this keyword

1. this keyword is use to access the properties of same class.
2. it is a current object of same class.
3. Using this keyword you can access the constructor, variable and method of same class.
4. Access Constructor from same class.
   1. You can access the constructor of same class using this keyword.
   2. It must be a first line in a constructor.
   3. One Constructor can only call another Constructor

**Static Keyword**

1. Using static keyword you can create a class level properties
2. You can create static variable, method and static class (inner/nested class).
3. static properties can be access without creating object and only using class name.
4. Static variable
   1. Static variables has to create inside class and outside any method.
   2. Static variables are the sharable between all the objects of the class.
5. Static Method
   1. Static methods can be access by class name only.
   2. Static methos can access the static properties of the class only.
   3. You cannot use super and this keyword inside static method.

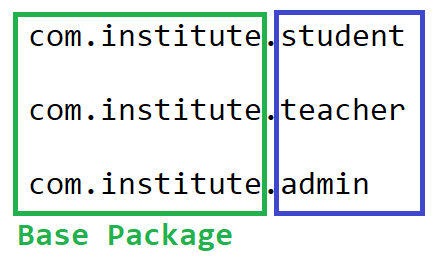
**Final Keyword**

1. Final keyword is use to create constants.
2. You can create final variable, method, class.
3. Final variable
   1. Final values will be unchanged once it is created.
   2. You cannot reassign the final variable value.
4. Final Methods
   1. The implementation of the final method is fixed.
   2. The final methods cannot be override.
5. Final class
   1. You cannot use final class as a super class for any other class.
   2. The final class properties are not final implicitly.

**Package, Import and access modifier**

**Package**

1. To create a group of java classes having similar functionality.
2. Packages are the folder in the file system.
3. Rules to create package
   1. Package statement must be a first statement in a file.
   2. It must be outside any class.
   3. Package statement is applicable for all the classes of the file.
   4. It must be creates using package keyword.
   5. There can be only one package statement present for a java file.



**Import**

1. Import statements are used access the classes from one package to another.
2. Rules to use import statement
   1. Import statement must be after package statement (if present)
   2. Import statement must be outside any class.
   3. It must be created with import keyword.
   4. Import statement can be write multiple time.
   5. It can be use to import all the classes from the package or access only single class.

Example:

import pack1.pack2.\*; 🡪 Import all the classes from the package.

import pack1.pack2.ClassName; 🡪 Import a specific class from the package.

* 1. In all the java files by default java.lang package is imported.

**Access Modifier**

1. Access modifier are used to manage the access of the classes and its properties into another class.
2. There are 4 access modifier. Out of which default is the access modifier which is added by default if you not explicitly define.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Access Modifier** | **Access in same class** | **Access in different class of same package using object** | **Access in different class of same package using Inheritance** | **Access in different class of different package using Inheritance** | **Access in different class of different package using Object** |
| public | **YES** | **YES** | **YES** | **YES** | **YES** |
| protected | **YES** | **YES** | **YES** | **YES** | **NO** |
| default/package | **YES** | **YES** | **YES** | **NO** | **NO** |
| private | **YES** | **NO** | **NO** | **NO** | **NO** |

**Highest to Lowest restrictive**

private 🡪 default 🡪 protected 🡪 public

**Polymorphism**

1. One object can have different form (one thing can be achieve in multiple ways)
2. There are 2 types of polymorphism
   1. Compile time polymorphism
      1. The method call resolves at the time of compilation and same will be followed at execution.
      2. To achieve this you have to use **overloading**
   2. Runtime polymorphism
      1. The method call is decided at the time of execution (compilation linking will not be followed at execution time).
      2. To achieve this you have to use **overriding**

**Overloading**

1. Method name must be same with different parameter list.
2. Overloading can be done in same class or sub class.
3. In Overloading parameter list must be different by following ways
   1. You can change the data type of the parameter.
   2. You can change the number of parameters.
   3. Sequence of parameter can be change.
4. Return data type may or many not be same.
5. Access modifier may or many not be same.
6. Can overload static methods. (you can overload main method)
7. Can overload final methods.
8. Can throw new and broader checked exception.
9. Advantage of overlading
   1. To improve the readability of the program.
   2. Provide easy interface to the user to use the functionality.

**Overriding**

1. Method name must be same and input parameter must be same.
2. Overriding must have to do in the sub class.
3. Return data type must be same if it is primitive or void else it can be change to sub type.
4. Access modifier can be change but must be less restrictive.
5. You cannot override the final methods.
6. You cannot override the static methods.
7. Cannot throw new and broader checked exception.
8. Advantage
   1. You can change the default implementation of parent class methods into sub class.

**Runtime Polymorphism**

1. There must be an inheritance between the classes.
2. The method must be override.
3. Method call happened using a polymorphic object.

**Abstraction**

1. Is use to hide the complex implementation/functionality and display only the important one to the user.
2. Abstraction can be achieved by 2 ways.
   1. Abstract Class
      1. Is use to achieve 0 to 100% abstraction. You can achieve the partial abstraction using abstract class.
   2. Interface
      1. You can achieve 100% abstraction.

**Abstraction Class**

1. Abstract class is use to achieve 0-100 abstraction which is also known as partial abstraction.
2. Abstract classes are mark with word abstract. Which contains abstract or non-methods.
3. Abstract method which do not have any implementation and it is just declare. Abstract method will be created using abstract keyword.
4. If any class has any abstract method created then the class must have to mark as abstract.
5. Abstract will be used as a parent class in the hierarchy.
6. You cannot create an object of abstract class. but you can use it as a reference.
7. Every sub class of the abstract class, has to provide the implementation (override) of all the abstract methods.
8. Abstract class can have a constructure. These constructure gets called whenever you create object of sub class of abstract class.
9. Abstract method cannot be static or final.
10. You can create a static and final method inside abstract class but it must be non-abstract method.
11. You cannot mark your abstract class as static or final.

**Interface**

1. Using Interface you can achieve 100% abstraction (till Jdk 1.7).
2. It is not a class, but every interface will get a .class file after compilation.
3. Interface will be created using interface keyword.
4. The methods created inside interface are by default public and abstract.
5. The variables created inside interface are by default public static and final.
6. Interfaces are implements by the class (abstract or non-abstract class).
7. Every sub(non-abstract) class of the interface has to provide the implementation for all abstract method.
8. Cannot create Object of interface but can be used as a reference.
9. Cannot create constructor inside interface.
10. Can create only public properties inside interface, no other access modifier is allowed inside interface.
11. You cannot create final method inside interface.
12. Cannot create static methods inside interface till JDK 1.7.
13. One interface can extends another interface, you can also extends more than one interface on single interface. Which is also known as multiple inheritance.
14. One class can implements more than one interface.
15. One class can extends another class and also implements the interfaces.

**Interface in JDK 1.8 and above**

1. You can create an implemented methods inside interface which is known as default method. (Can create default methods)
2. Can create static method inside interface.
3. Functional interface is introduce to achieve the functional programing in java.
   1. Functional interface is an interface which has only one abstract method.
   2. To ensure the interface is functional interface you can use an annotation as @FuntionalInterface.
   3. There can be any number of default or static methods inside functional interface, but there must be a single abstract method.