**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/>

**Revision Session**

Writing and executing first java program, Data Type, Operators, Control flow statements,

Array (Example),

Array Extra Example

Nested For loop Extra Example

Memory Allocation (heap and stack)

**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 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

\*/

**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 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



**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/2023-06/R/eclipse-jee-2023-06-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 option if you wanted to create main method.
4. Click on Finish

**Polymorphic Object**

* + - 1. Its an object in which you can create a reference of parent class and object of child class.
      2. Polymorphic object is applicable in the inheritance only.
      3. These Object will be use to achieve the runtime polymorphism.
      4. Using this object, you can access only the common properties of parent and child class
      5. Syntax:

**Parent obj = new Child();**

**Object class**

* + - 1. Is a parent class of all java classes.
      2. There are some common properties present inside Object class, which are available inside all the java class.
      3. If you are not defining any parent class then by default java includes the parent class as a Object class.
      4. Important method of Object class.
         1. toString()
         2. hashCode()
         3. getClass()
         4. equals()
         5. finalized()
         6. wait(), wait(long), wait(long, int)
         7. notify()
         8. notifyAll()

**Advantage of Inheritance**

Can achieve the reusability.

Can achieve the extensibility.

To achieve the runtime polymorphism inheritance is required.

**Constructor**

1. Constructor is known as special method of the class, which is use to **initialize the instance variable**.
2. Constructor get called automatically at the time of object creation and you can provide the values for instance variable from constructor.
3. There are some rules to create constructor
   1. Constructor must have same name as class name.
   2. Constructor must not have a return data type.
   3. Constructor can be created using any access modifier such as private, public, protected, default.
   4. There can be input parameters inside constructor.
   5. There can be more than one constructor inside class.
   6. Constructor gets called at time of object creation, and you cannot call them manually using object or dot operator.
4. Every class has a constructor, If you do not provide any constructor manually then, java will provide a default constructor internally and if you are providing any constructor then, java is not responsible to provide any constructor.

**super, this, final, static keyword**

Super Keyword

1. Super keyword is use to access the properties of the super class inside sub class.
2. Super class properties included the variable, constructor and method of the super class.
3. By Default every java constructor has a first line as super(); which give call to a parent class default or no param constructor.
4. Super class constructor must be call from a sub class constructor and it must be a first line of a constructor.

This keyword

1. Using this keyword you can access the properties of same class.
2. You can access the method, instance variable or constructor of same class using this keyword.
3. It points to a current class object.
4. You can access the constructor of same class from another constructor of same class.
5. The constructor call must be the first line in a constructor.
6. You can use either super or this to call constructor, but cannot use both at a time.

Final Keyword

1. Final keyword is use to create constants.
2. Final keyword can be use for variable, method or class.
3. If you create final variable then its values will be fixed and it cannot be change once assign.
4. Final variable can be static, instance or local variable.
5. Final Variables has to initialized at declaration time.
6. By making method as final you can restrict the method implementation changes by overriding the method. The implementation of the method is fixed for all the users.
7. Final classes cannot be inherit.

Static keyword

1. Static keyword is use for variable, method, class or static block.
2. Static properties can be accessible using a class name without object of class.
3. Static variable can be accessible using a class name without object.
4. Static properties are sharable properties for all the objects and in application, it will be created in a static/method area.
5. Static properties gets loaded in the memory at the class loading.
6. Static methods can be access using class name and without class object.
7. Static methods can access only static properties of the class. Non static properties cannot be access directly inside static method.
8. In static method you cannot use super or this keyword.
9. Static class must be an inner/nested class. You cannot create outer class as static.
10. Static block is use to initialize the static variable and it gets called before constructor.

**Package, Import & access modifier**

**Package**

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

Package name should be combination of domainName , project/client and module

Example: com.amazon.order

com.amazon.user

com.amazon.product

com.icici.accounts

**import**

1. Import statement is use to access the properties of one class into another class from the different packages.
2. Rules to use import statement
   1. Import statement must be write after package statement (if present)
   2. Import statement must be write outside any class and inside file.
   3. There can be multiple import statement in a file.
   4. The import statement written inside file is applicable for all the classes of that files.
   5. Import statement can use for a single class from a package or you can import all the classes from a package at the same time.

Example:

**import package1.package2.ClassName;** 🡪 To import single class from a package

**import package1.package2.\*;** 🡪 to import class the classes from the package.

1. By default every java files imports all the classes from the java.lang package.

**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 of 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** |

Arrang Access modifier by restriction level from high to Low

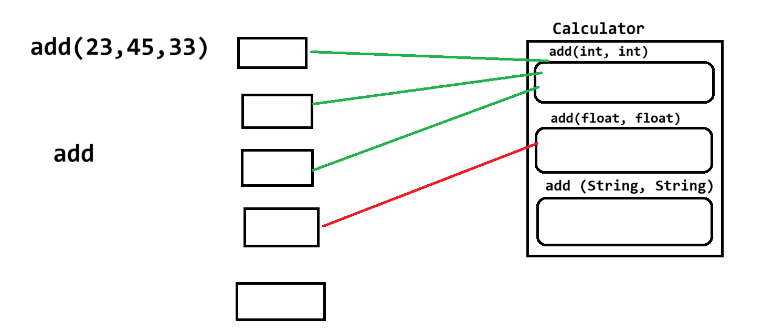
private 🡪 default 🡪 protected 🡪 public

**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. It is a way to call a methods dynamically using a object at run time of the program.
2. To achieve runtime polymorphism following steps needs to follow
   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 keyword 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 class will be used as a parent class in the hierarchy.
6. Every sub class of the abstract class, has to provide the implementation (override) of all the abstract methods.
7. You cannot create an object of abstract class. but you can use it as a reference (like runtime polymorphism).
8. You can extends one abstract class on another abstract class, in this case there is no need to provide the implementations of all the abstract methods.
9. Abstract class can have a constructure. These constructure gets called whenever you create object of sub class of abstract class.
10. Abstract method or abstract class cannot be static or final.
11. You can create a static and final method inside abstract class, but it must be non-abstract method.
12. It is not mandatory to have any abstract method inside abstract class.

**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 Updates in JDK-8**

1. Static implemented methods you can create inside interface.
2. Main method is allowed inside interface
3. The method with implementation can be created inside interface which must be mark as default.
4. Functional Interface
   1. The interface which has only one abstract method must be present.
   2. Functional Interfaces are us to achieve the functional programing using Lambda expression.
   3. To make sure that created interface is functional interface you can use @FunctionaInterface annotation on the interface.
   4. You can create any number of static or default inside functional interface.

**Exception Handling**

**Custom Exception**

1. It is use to provide the custom exception implementation for the custom scenarios.
2. Custom Exception can be created using following steps
   1. Create a java class.
   2. Extends the Exception class or its any sub class.
   3. Provide your own implementation in to the exception class.
3. Custom exception has to be thrown manually.

FAQ:

1. Difference between Error and Exception
2. Difference between checked and unchecked exception
3. Difference between throw and throws
4. Difference between final and finally

**Multi-Threading**

1. Thread is a light weight process is a part of main process.
2. Every thread has its own memory which is also known as call by stack.
3. Every thread can execute independently.
4. Can achieve the faster performance using multiple threads.
5. You can achieve the maximum CPU utilization
6. Every thread has some task to execute.
7. Every thread will execute by JVM. JMV has full control on the execution of the thread.
8. Every java program use a thread internally, by default java create an thread and execute the program. The default thread created by java internally is a main thread.

**Create thread in Java**

1. In java to create thread there are 2 ways
   1. Extending **Thread class**
   2. Implements **Runnable interface**
2. Internally Thread class implements the Runnable interface.
3. You have to override the **run() method** to assign task to a thread.
4. To make thread ready to run you have to use **start() method**.

**Thread mostly used method**

|  |  |
| --- | --- |
| **Method** | **Description** |
| run() | It is use to assign a task to a thread. This method gets called whenever thread start executing. |
| start() | Using this method you can make thread ready to run. Thread will be added in the queue and wait for the execution. |
| currentThread() | This is the static method of the thread class, It will return the currently executing thread object. |
| setName(“Name”)  getName() | This method is use to set or get the name of the thread. |
| setPriority(int)  getPriority() | You can set or get the priority of the thread. Thread Priority muse be between 1-10. You will get IllegalArgumantException if value is other than 1-10. Here 1 is the MIN\_PRIORITY, 5 is the MID\_PRIORITY, 10 MAX\_PRIORITY. |
| sleep(int)  sleep(int, long) | Sleep is a static method, using this method you can pause the execution of current thread for a given time. Once given time is expire then thread will come back to runnable stage. This method throws checked InterruptedException which has to handle. |
| join()  join(int)  join(int, long) | Join is a non static method, it pause the execution of current thread either until completing execution of other thread or till the given time expire. This method throws checked InterruptedException which has to handle. |
| suspend() | It will pause the execution of the thread until some other thread calls the resume method. |
| resume() | It will resume the execution of the suspended thread, suspended thread will pull from the pause stage to runnable stage. |
| wait()  wait(int)  wait(int, long) | Using this thread will release the lock and go into pause state, Thread will stay into pause state until given time expire or until some other thread calls notify or notifyAll method. Wait method has to call within a synchronized only. This method throws checked InterruptedException which has to handle. |
| notify() | Using this method you can notify the single thread which is in the waiting state and it will come into runnable state. |
| notifyAll | Using this method you can notify the all the thread which is in the waiting state and it will come into runnable state. |

**Thread Life Cycle**

1. JVM is responsible to execute the thread.
2. JVM maintains the life cycle of the thread.
3. There are different stages of the life cycle
   1. New/Born Stage
   2. Runnable Stage
   3. Running Stage
   4. Terminate/Dead stage
   5. Wait/sleep/pause stage



**Synchronization**

1. If multiple thread using a share resources then there can be data inconsistency issue in the execution to avoid it you can use the synchronization.
2. Synchronization is a process of acquiring and releasing lock on the share resources to avoid the data inconstancy issue.
3. In this process the acquiring and releasing lock will be achieve internally by java.
4. Due to synchronization the waiting time of the thread can be increase which affects the application performance.
5. Synchronization is one of the reason of getting dead lock scenarios in the application.
6. Synchronization can use by 2 ways
   1. Using synchronized block
   2. Using synchronized method

Wrapper class

|  |  |
| --- | --- |
| **Primitive Type** | **Wrapper Class** |
| byte | Byte |
| short | Short |
| int | Integer |
| long | Long |
| float | Float |
| double | Double |
| boolean | Boolean |
| char | Character |

**Java DataBase Connectivity (JDBC)**

1. Java application will connect with database and execute the operations.
2. In Database SQL language is used to perform operation. This can be done from the java application.

