**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 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 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 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, 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**

**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
   1. **Primitive Data Type**
      1. Are the values which is in non-object format.
      2. These values will have a fixed sized.
   2. **Non-Primitive Data Type**
      1. Are the values which is in the form of Object.
      2. There is no fix size for the non-primitive data type.



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

**- 2n-1 to 2n-1 - 1**

Here, n is the number of bits.

byte : - 28-1 to 28-1 – 1

-128 to 127

**Rules to store Values in Data Type**

1. long:
   1. the values should be suffix with l or L.
2. float
   1. The value must be suffix with f or F.
3. char
   1. char values must be in single quotes.
   2. You can store symbols or numeric values also along with alphabets.
   3. You can store all the ASCII values inside char.
   4. 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
   1. Local Variable
      1. Variable which is created inside method or inside method argument is called as Local Variable.
      2. Local variable can be access within a method only it cannot be used outside method.
      3. Local variable has initialized before use.
   2. Instance variable
      1. Variable which is created inside class and outside any method is called instance variable.
      2. This variable is accessible inside class and also outside class using Object.
      3. Instance variable can be initialized with default value if you have not provided any value explicitly.
   3. Static/class variable
      1. Variables which are created outside any method but will present inside class with static keyword.
      2. These variables are accessible inside class and also into another class using Class name.
      3. Static variable can be initialized with default value if you have not provided any value explicitly.

****

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 an 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
   1. **Implicit Casting** 
      1. The casting which is done by java automatically and internally without adding any extra code.
   2. **Explicit Casting** 
      1. The casting will not happen by java, for this developer has to add some extra code in the program.

****

**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 \_\_\_\_\_

?, :

****

**Task-1**

WAP to store the product value and print the value of the product.

Store the Product id, product price, product quantity

And print the product id, product price, product quantity and the total product price (price \* quantity)

**Task-2**

Calculate electricity bill, consider per unit price is 3.5 rupee and tax is 10 percent. Print the payable amount for given consume unit.

Example

Total Unit Consume = 100

Cost Per unit : 3.5

Tax : 10%

100\*3.5 = 350

On 350 10% tax = 35

Payable Amount = 350 + 35 = 385

**Task-3 (26-July)**

WAP to find the Grade using percent.

percent between 75-100 - Distinction

Percent between 60-75 - First Class

Percent between 50-60 - Second Class

Percent between 40-50 - Pass Class

Percent between 0-40 - Failed

Any other percent value - "Invalid Percent...."

Variable Declaration and initialization (start point)

do

{

Statement(s)

Increment/decrement of the variable

}

while(condition/Boolean expression);

**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



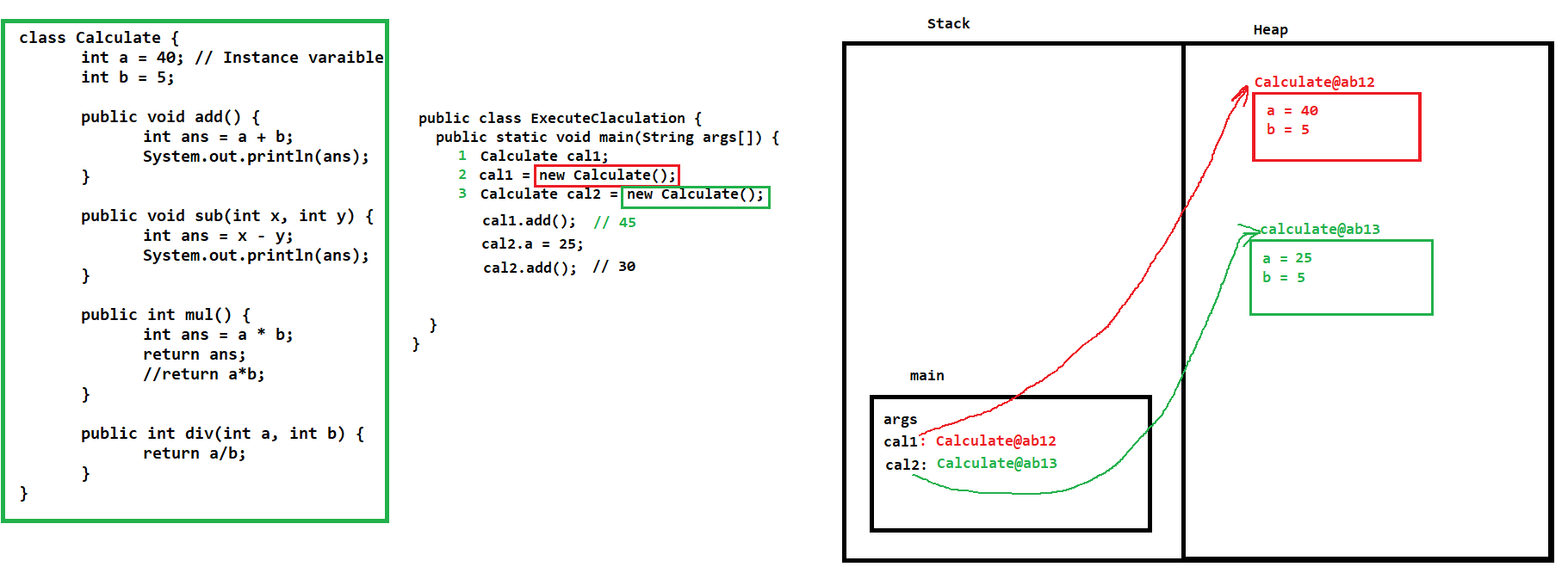
**Object**

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. If you wanted to accept the values from the user. Then you can use predefine classes of java.
2. There are multiple ways or classes to accept user data.
   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. This is use to accept the user at the time of execution of the program.
3. The values will be pass from the execution command.
4. These values will be received inside a main method. Which will be in the string format.

**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 user data form the console at the time of program execution.
4. Scanner class has multiple functions which is use to accept the data of different type.
5. To use the scanner class you have to create instance/object of It and use t predefine methods.

**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

**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.

**Advantage of Inheritance**

* + - 1. Can achieve the reusability
      2. Can achieve the extensibility.
      3. Inheritance is required to achieve the Runtime polymorphism.

**Polymorphic Object**

In the inheritance hierarchy you can also create polymorphic object.

Polymorphic object, is an object which has parent reference with child class object.

Every parent class reference can hold the object of its child.

Polymorphic object is required to achieve runtime polymorphism.

**Syntax:**

ParentClass obj = new ChildClass();

**Object Class**

1. Object class is java build in class.
2. Object class is parent class of all the java classes.
3. Every Java class directly or indirectly inherits the property from Object class.
4. Object class has some common properties which are required in all the java classes.
5. Some of the methods of Object class

toString()

hashCode()

getClass()

equals()

finalized()

wait(), wait(int), wait(int, long)

notify()

notifyAll()

**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.

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

**this keyword**

1. this keyword is points to current class object.
2. this is an object of the class.
3. using this keyword you can access the properties of same class such as constructor, instance variable and methods of the same class.

**super keyword**

1. super keyword is use to access the properties of super class such as constructor, method and variable of parent class
2. Super is not an object of parent/super class.
3. Constructor call using super
   1. Every sub class constructor by default calls super class default or no-param constructor. Java Internally adds super() call as a first line of every constructor.
   2. Calling constructor must be a first statement in a constructor.
   3. You cannot call same class constructor using this and parent class constructor using super at a same time.

**Final keyword**

1. Final keyword is use to create a constant.
2. Final keyword can be use for variable, method or class.
3. Final For variable
   1. Value of the final variable is always fix and you cannot change it once it is assign.
   2. You can mark instance or local variable as final.
4. Final for Method.
   1. Final method implementation is fixed for all the uses and no one can change the implementation.
5. Final class
   1. Final class cannot be inherit by any other class.

**Static Keyword**

1. Using static keyword you can create a class level properties which can be use without creating objects.
2. Static keyword is use to create static variable, method, class or block.
3. Static properties load inside a memory at class loading.
4. Static for variable
   1. Static variable must be created inside class and outside any method. It cannot be create inside method.
   2. Static variables are the sharable variable and there will be no separate copy of variable provided to an object.
5. Static method
   1. Static methods can be access using class name without object.
   2. Static method you cannot access non static properties of the class.
6. Static class
   1. Static class must be an inner class. you cannot create outer class as a static class.
7. Static block
   1. Static block is use to initialize the static properties of the class.
   2. Static block gets call only once and before the constructor call.

**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.
4. Conventions to create a package

**domain.project/client.module**

Example: com.icici.loan

**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 to 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 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. 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.
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 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.
13. 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**

1. Default Methods
   1. You can create an implemented method inside interface, using default keyword.
2. Static Methods
   1. You can create a static implemented method inside interface.
3. Functional Interface
   1. The interface which is created without only one abstract method is called functional interface.
   2. These functional interface is use to achieve a functional programing and lambda expression in java.
   3. To make sure the interface is functional interface you can use @FunctionalInterface annotation.
   4. Functional interface can have any number of default or static methods but it must have only one abstract method.

**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.

**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 important methods**

|  |  |
| --- | --- |
| **Method** | **Description** |
| currentThread() | This is the static method which return the object of currently executing thread |
| run() | This method is use to assign task to a thread. This method will be call by JVM once the execution of the thread started |
| start() | This method is use to make  to run. The thread will be added in the execution queue. |
| setName(“String”)  getName() | Using this method you can set or get the name of the thread. |
| setPriority(int)  getPriority() | Using this methos you can set and get the thread priority. Thread priority must be between 1-10.  Where 1 is MIN\_PRIORITY, 5 MID\_PRIORITY, 10 MAX\_PRIORITY |
| sleep(long)  sleep(long, int) | This is the static method. Using this method you can pause the execution of currently executing thread and the execution will resume back once the given time expire. It throws the checked InterruptedException which has to handle. |
| join()  join(long)  join(long, int) | This is non static method, using this method you can pause the execution of currently executed thread for the another thread completion or till the given time expire. It throws the checked InterruptedException which has to handle. |

**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

