Java full stack theory

1-7-2021

**1. Variables**

**2. Datatypes**

**3. Operators**

**4. Structure**

**5. Conditional Statements**

**6. Looping Statements**

**---------------------------------------------------------------------**

**1. Object, Class, Keywords and Identifiers**

**2. Methods, Constructor and Blocks**

**3. Inheritance**

**4. Overloading and Overriding**

**5. Access Modifiers and Encapsulation**

**6. Casting, Abstract keyword, Interface and Arrays**

**7. Polymorphism**

**8. Abstraction**

**----------------------------------------------------------------------**

**1. Java Libraries -> String -> Lambda Functions**

**2. Exception Handling**

**3. File Handling**

**4. Multi-Threading**

**5. Collection Framework**

**Note: After the completion of java course, mini project would be done.**

**What is Java?**

**- Java is a high level, platform independent, object-oriented programming language.**

**- High Level Language is a language which is in normal English i.e. Human Understandable Form.**

**- Programming Language is a medium to interact or communicate with the system.**

**Why do we use Java? or Features of Java.**

**1. Simple**

**2. Platform Independent**

**3. Extensible**

**4. Object Oriented**

**5. Automatic Garbage Collector**

**6. Secured**

**7. Robust**

**2-7-2021**

**Working of a Java Program (or) How is Java platform independent (or) WORA Architecture**

1. **We develop a java program and save the file with the extension (.java).**
2. **Next, we compile a java program to check if there are any error in the program or not.**
3. **If the compilation is unsuccessful (program has errors) then we need to Debug.**
4. **If the compilation is successful, the byte code (intermediate code) gets generated with the extension (.class).**
5. **Once the Byte code is generated, we can execute (interpret) the program on all operating systems.**
6. **WORA stands for Write Once Run Anywhere.**

**Note: Refer Screenshot for diagram**

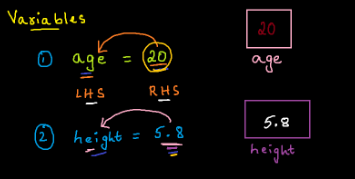
**History:**

1. **Java was introduced by a company called as Sun Micro Systems.**
2. **Java is owned by a company called as Oracle presently.**
3. **James Gosling was the Person who developed JAVA.**
4. **Previous Names of Java are Green Talk and Oak.**

**Variables**

1. **Variable is Container in order to store some data or information.**

**Example:**



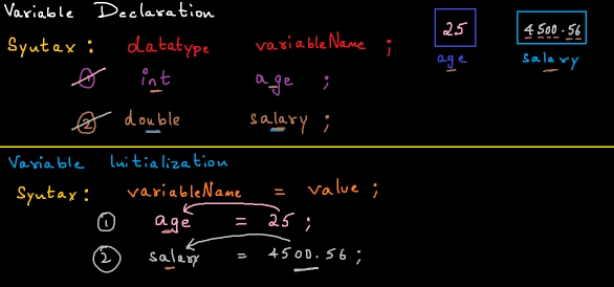
**Datatypes**

1. **Datatype is an indication of the type of data stored into a variable.**
2. **In order to store Non-Decimal Numeric Values, we make use byte, short, int, long.**
3. **In order to store Decimal Numeric Values, we make use float, double.**
4. **In order to store true/false, we make use boolean.**
5. **In order to store Single Character in single quotes, we make use char.**

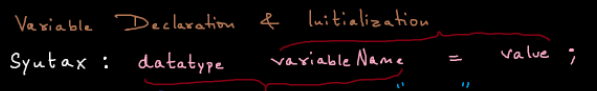
**Note: All the above 8 Datatypes are referred Primitive Datatypes**

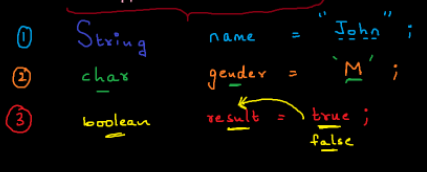
1. **In order to store a sequence of characters we make use of String.**

**1.** **Variable Declaration and Initialization**



1. **Variable Declaration and Initialization**





**5-7-2021**

1. **Arithmetic Operators**
2. **+**
3. **–**
4. **\***
5. **/**
6. **%**

1. **Assignment Operators**
2. **=**
3. **+=**
4. **-=**
5. **\*=**
6. **/=**
7. **%=**

1. **Conditional or Comparison or Relational Operators**
2. **<**
3. **<=**
4. **>**
5. **>=**
6. **==**
7. **!=**

**Note: Return type is Boolean value (true or false)**

1. **Logical Operators**
2. **&& -> AND**
3. **|| -> OR**
4. **! -> Not**

**Note: Return type is Boolean value**

**Truth Tables:     True-T     False-F**

**AND (&&)**

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **O/P** |
| **T** | **T** | **T** |
| **T** | **F** | **F** |
| **F** | **T** | **F** |
| **F** | **F** | **F** |

**OR (||)**

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **O/P** |
| **T** | **T** | **T** |
| **T** | **F** | **T** |
| **F** | **T** | **T** |
| **F** | **F** | **F** |

**NOT**

**T   🡪   F**

**F   🡪  T**

1. **Unary Operators**

**++    🡪 Increment by 1**

**--     🡪 Decrement by 1**

**int x = 5;**

**int y = x++;**

**Post-Increment -> First Assign, Then Increment**

**---------------------------**

**int a = 10;**

**int b = ++a;**

**Pre-Increment -> First Increment, Then Assign**

**int x = 5;**

**int y = x--;**

**Post-Decrement -> First Assign, Then Decrement**

**---------------------------**

**int a = 10;**

**int b = --a;**

**Pre-Decrement -> First Decrement, Then Assign**

**6-7-2021**

**Comments**

**1. Additional Information which will not affect the execution of a program.**

**// Single Line Comment**

**/\* Multi**

**Line**

**Comment \*/**

**Conditional Statements or Decisional Statements**

* **Conditional Statements are used to take some decision based on the condition specified.**
* **They are used for decision making.**
* **Different Decisional Statements are as follows:**

1. **Simple If**
2. **If Else**
3. **If Else If**
4. **Nested If**
5. **Switch Statement**

**Simple If:**

* **It is a Decision-Making statement wherein we execute a set of the instructions if the condition is true.**

**If Else:**

* **It is a Decision-Making statement wherein we execute a set of the instructions if the condition is true and another set of instructions if the condition is false.**

**If Else If**

* **- If Else If Condition is used when we need to check or compare multiple conditions.**

**Looping Statements:**

* **Looping Statements are generally used to perform repetitive task.**
* **Loops are used to repeat the execution of a set of instructions and traverse a group of elements.**
* **Different Looping Statements are as follows:**
  + **For Loop**
  + **While Loop**
  + **Do-While Loop**
  + **Nested For Loop**

**For Loop:**

* **For loop is used to execute a set of instructions for a fixed no of times.**
* **It has a logical start point and end point.**

**While Loop:**

* **While loop is a looping statement which keeps on executing until the condition is false.**

**Do-While Loop:**

* **Do-While loop is similar to while loop but do while loop executes a set of instructions and then checks the condition.**

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

**Difference between while loop and do-while loop**

|  |  |
| --- | --- |
| **While Loop** | **Do While Loop** |
| 1. **Checks Condition first and then executes a set of instructions.** | 1. **Executes a set of instructions first and then checks the condition.** |
| 1. **Does not execute even once if the initial condition is false.** | 1. **executes at least once even if the initial condition is false.** |

**9-7-2021**

**Methods or Functions**

**1. A Method is a set of instructions or a block of code in order to perform a specific task.**

**syntax: AccessSpecifier returnType methodName( arguments )**

**{**

**/\* statement-1**

**.**

**.**

**statement-n**

**return     \*/**

**}**

**2. If we want to execute a method, we need to call it or invoke it.**

**syntax: methodName();**

**3. A method can be called multiple times.**

**4. Reusuablity of code is increased with the help of methods.**

**5. If a method returns something, store it or print it.**

**void: void is an indication to the caller, that the method does not return anything.**

**return: return is used to transfer the control back to the caller.**

**Different ways of writing a method:**

**1. Method without arguments and without return statement.**

**2. Method with arguments and without return statement.**

**3. Method without arguments and with return statement.**

**4. Method with arguments and with return statement.**

**13-7-2021**

**Object Oriented Programming**

**Object**

* **Anything which is present in the real world and physically existing can be termed as an Object.**
* **The Properties of every object is categorized into 2 types:**
  + **States**
  + **Behaviours**
* **States are the properties used to store some data.**
* **Behaviour are the properties to perform some task.**

**Class**

* **class is a blue print of an object.**
* **It’s a platform to store states and behaviours of an object.**
* **class has to be declared using class keyword.**
* **class can also act as datatype.**

**Object Creation or Instantiation**

**- Object is a copy or instance of a class.**

**- In order to store or load non static members/properties within the memory, we need to create an object.**

**- Objects are created inside a memory location called as HEAP Area.**

**- new Operator is responsible for creating an object internally.**

**syntax:  ClassName referenceName  =  new ClassName();**

**- Any number of objects can be created for a class.**

**- We can access non static properties in same class or another class with the help of object creation and dot operator.**

**14-7-2021**

**DEFAULT VALUES**

**- If a variable is declared and not initialized to any value, then the compiler will automatically initialize to its default value.**

**- Default values are applicable only for Member Variables (Static and Non-Static Variables).**

**Default values are as follows**

**byte, short, int, long ----> 0**

**float, double ----> 0.0**

**char ----> '/n10' (Unicode value) Java does not understand empty white space( )**

**boolean ----> false**

**String ----> null**

**Scanner**

**1. Scanner is a pre-defined class in java.util package.**

**2. Scanner class is used to accept dynamic input from the User.**

**Rules to accept dynamic input from the user (or) Rules to work around with Scanner class**

**1. Create an object of Scanner class.**

**2. Pass System.in to the Constructor call.**

**syntax: Scanner scan = new Scanner(System.in);**

**3. import Scanner class from java.util package.**

**syntax: import java.util.Scanner;**

**4. Make use of pre-defined methods to accept dynamic input.**

**Important method used with respect to Scanner class**

**1. byte - nextByte()**

**2. short - nextShort()**

**3. int - nextInt()**

**4. long - nextLong()**

**5. float - nextFloat()**

**6. double - nextDouble()**

**7. boolean - nextBoolean()**

**8. String - next() or nextLine()**

**9. char - next().charAt(0)**

**16-7-2021**

**Method Overloading**

**1. In a class having multiple methods with the same name, but difference in arguments is called as Method Overloading.**

**In order to achieve method overloading we need to satisfy either 1 of the following 3 rules.**

**1. There should be a change in the No of Arguments.**

**2. There should be a change in the Datatype of the Arguments.**

**3. There should be a change in the order/sequence of the Datatypes.**

**Note:**

**1. Both Static and Non-Static methods can be Overloaded.**

**2. Yes, we can overload Main() as well, But the execution starts from the main() which accepts String[] as the argument.**

**3. returntype might be same or different.**

**4. Method Overloading is also referred as Compile time Polymorphism.**

20-7-2021

**Blocks**

**1. Blocks are a set of Instructions/Block of code used for initialization.**

**2. Blocks are generally categorized into**

**a. static block**

**b. non-static block (instance block)**

**static block**

**1. Static blocks are a set of instructions used to initializing static variables.**

**syntax: static {**

**}**

**2. Static blocks always gets executed even before main() or during class loading time.**

**3. We can have Multiple Static Blocks and the execution will happen in a sequential Manner.**

**non-static block (Instance Block)**

**1. Non-Static blocks are a set of instructions used to initializing static variables and non-static variables.**

**syntax :**

**{**

**}**

**2. Non-Static blocks always gets executed during Object creation (Instantiation).**

**3. We can have Multiple Non-Static Blocks and the execution will happen in a sequential Manner.**

**JDK**

**- Java Development Kit**

**- JDK is a software which contains all the resources in order to develop and execute java programs.**

**JRE**

**- Java Runtime Environment**

**- JRE is a software which provides a platform for executing java programs.**

**JIT Compiler**

**- Just In Time Compiler**

**- JIT Compiler complies/coverts Byte Code into machine understandable langauage.**

**Class Loader**

**- Loads the Class from secondary storage to executable area.**

**Interpreter**

**- Interprets the code line by line.**

**JVM**

**- Java Virtual Machine**

**- JVM is the Manager of the JRE.**

**JVM Architecture**

**----------------**

**1. Heap Area - Objects get created here.**

**2. Class Area or Static Pool - All the static members gets stored here.**

**3. Stack - Execution happens inside stack.**

**4. Method Area - Implementation of methods is stored here.**

**5. Native Area**

21-7-2021

**Constructor**

**1. Constructor is a set of instructions used for initialization (Assigning) and Instantiation (Object Creation).**

**2. Constructor Name and Class Name should always be same.**

**3. Constructors will not have return type.**

**4. Constructors will get executed at the time of object creation.**

**syntax: AccessSpecifier ClassName(optional arguments)**

**{**

**// Set of Instructions**

**}**

**5. Constructors are categorized into 2 types:**

**a. Default Constructor**

**b. Custom/User-Defined Constructor**

**Default Constructor**

**1. If a constructor is not explicitly present in a class, then the compiler will automatically generate a constructor and that constructors are called as Default Constructor.**

**2. Default constructor neither accepts any arguments nor has any implementation.**

**Custom/User-Defined Constructor**

**1. If a constructor is explicitly defined inside a class by the user or the programmer, then we refer it as custom/user-defined constructor.**

**2. They are further categorized into 2 types:**

**i.  Non Parameterized Custom Constructor**

**ii. Parameterized Custom Constructor**

**NOTE: WHEN THERE IS DEFAULT CONSTRUCTOR, THEN CUSTOM CONSTRUCTOR CANNOT BE PRESENT AND VICE VERSA.**

**Global/Member Variable and Local variable**

**1. Global/Member variables are those variables which are declared in the class limit/Scope.**

**2. They can be accessed globally ie. through out the class.**

**3. Global/Member variables are categorized into**

**a. static**

**b. Non-static**

**1. Local Variables are those variables which are declared within a specific scope or limit such as method, constructor, etc......**

**2. Local variables are accessible within that specific scope.**

**this keyword**

**1. In java, we can have both member/global and local variable names same, then always the local variables will dominate the member variables.**

**2. In order to avoid the dominating part we make use of "this" keyword.**

**3. this is keyword which is used to point to the current object/instance.**

22-7-2021

**INHERITANCE**

**-----------**

**1. Inheritance is a process of one class acquiring the properties of another class.**

**2. A class which gives or shares the properties are called as Super Class, Base Class or Parent Class.**

**3. A class which acquires or accepts the properties are called as Sub Class, Derived Class or Child Class.**

**4. In java, we achieve inheritance with the help of 'extends' keyword.**

**5. Inheritance is also referred as "IS-A" Relationship.**

**6. In java, Only Variables and methods are inherited whereas blocks and constructors are not INHERITED.**

**Types of Inheritance**

**---------------------**

**1. Single Level Inheritance**

**2. Multi-Level Inheritance**

**3. Hierarchical Inheritance**

**4. Multiple Inheritance**

**5. Hybrid Inheritance**

23-7-2021

**super keyword**

**1. super is a keyword which is used to access the super class properties.**

**syntax: super.variableName or super.methodName()**

**Note:**

**1. this -> points to current object**

**2. super -> points to super class object**

**Method Overriding**

**1. The process of Inheriting the method and changing the implementation/Definition of the inherited method is called as method overriding.**

**2. In order to achieve method overriding, we have to follow the below rules:**

**i.   Method Name must be same.**

**ii.  Arguments should be same**

**iii. returntype should also be same.**

**Note:**

**1. Access Specifier should be same or of Higher Visibility.**

**2. While overriding a method we can optionally use annotation ie. @Override**

**3. annotation was introduced from JDK 1.5**

**final Keyword**

* **final is a keyword which can be used for a variable, method and class**

**- final variable acts as a constant, whose value cannot be re-initialized.**

**- final method can be inherited but cannot be overridden.**

**- final class cannot be Inherited.**

28-7-2021

**CONSTRUCTOR CHAINING**

**1. The Process of one constructor calling another constructor is called as constructor chaining.**

**2. Constructor Chaining can be achieved only in case of constructor overloading.**

**3. Constructor Chaining can happen in two ways:**

**a. Constructor Chaining in same class can be achieved using this calling statement ie. this().**

**b. Constructor Chaining in another class can be achieved using super calling statement ie. super() & IS-A relationship.**

**Note :**

**1. this() or super() should always be the first executable line within the constructor.**

**2. Recursive Chaining is not possible, Therefore if there are 'n' constructors we can have a maximum of 'n-1' calling statements.**

**3. We cannot have multiple calling statements inside a single constructor.**

**1. super() is used to invoke or call constructor of another class.**

**a. IS-A (Inheritance) --> extends**

**b. super() ie. super calling statement.**

**2. super() can be used in 2 ways:**

**- Implicit super()**

**- Explicit super()**

**i.  implicitly**

**--------------**

**When we create an object of a class, and if that class has a super class, and if that super class has a non-parameterized constructor, then the sub class constructor will invoke the super class constructor implicitly.**

**ii. explicitly**

**-------------**

**When we create an object of a class, and if that class has a super class, and if that super class has a parameterized constructor, then the sub class constructor should invoke the super class constructor explicitly, otherwise we get compile time error.**

2-8-2021

**Packages**

**1. Packages are nothing but Folder or Directory.**

**2. Packages are used to store classes and interfaces.**

**3. Searching becomes easy.**

**4. Better Maintenance of the Programs.**

**package com.google.gmail;**

**class Inbox {**

**}**

**Access Modifiers or Access Specifiers**

**1. Access Modifiers or Access Specifiers is used to specify the accessibility (boundary or range) of a specific member (class, variable, method and Constructor).**

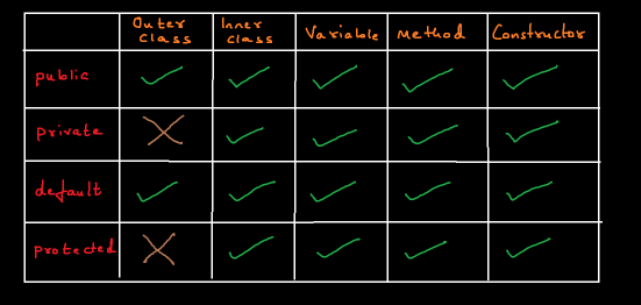
**2. The different Access Modifiers are as follows:**

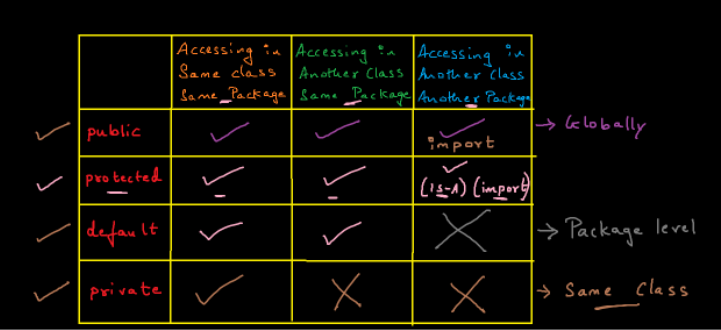
**i.   public**

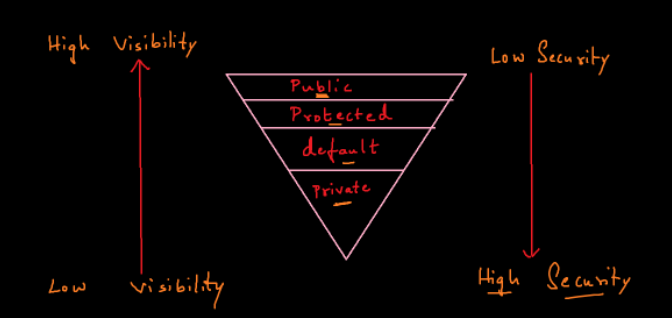
**ii.  private**

**iii. protected**

**iv.  default**







3-8-2021

**Encapsulation**

**1. Encapsulation is a Process of Wrapping/Binding/Grouping Data Members with its related Member Functions in a Single Entity/Unit called as class.**

**2. The process of grouping and protecting data members and member functions in a single entity called as class.**

**3. The Best Example for Encapsulation is JAVA BEAN CLASS.**

**Specifications of JAVA BEAN CLASS (1, 3, 4)**

**1. Class should be public non abstract class.**

**2. Class should have public non parameterized constructor.**

**3. Variables or Data Members should be declared as private.**

**4. Those Data Members should have respective public getter method and public setter method.**

**5. Class should implement a marker interface called as Serializable.**

**Advantages of Encapsulation**

**1. Validation can be done. (Protecting -> data security)**

**2. Flexible ie. Data can be made either write only or read only.**

**3. Readability of the Code is Improved.**

**Assignment:**

**-----------**

**1. Create a non-abstract class called as Doctor.**

**2. declare 3 private attributes called as name, age and salary.**

**3. Declare respective getter() and setter().**

**4. Access all those methods from another class.**

4-8-2021

**TYPE CASTING**

**1. Type Casting is a process of converting/storing one type of value/data into another type of value/data.**

**2. They are classified into 2 types:**

**i. Primitive Type Casting**

**ii. Non-Primitive Type Casting (Derived Casting)**

**3. Primitive Type Casting is further divided into 2 types:**

**i. Widening:**

**- Converting Smaller type of data into Bigger type of data.**

**- Widening happens Implicitly/Automatically.**

**ii. Narrowing:**

**- Converting Bigger type of data into Smaller type of data.**

**- Narrowing happens Explicitly.**

**Non-Primitive Casting or Derived Casting or Class Type Casting**

**1. Non-Primitive Casting or Derived Casting or Class Type Casting can be divided into 2 types:**

**i.  Upcasting**

**ii. Down-casting**

**Upcasting**

**1. Creating an object of sub class, and storing it's address into a reference of type Superclass.**

**2. With Upcasted Reference we can access only superclass Members/Properties.**

**3. In order to achieve upcasting, IS-A Relationship mandatory.**

**4. Upcasting will have implicitly/Automatically.**

**5. Superclass reference, Subclass Object.**

**Down-casting**

**1. The process of converting the upcasted reference back to Subclass type reference is called as Down-casting.**

**2. With the Subclass/Down-casted reference we can access both superclass and subclass members properties.**

**3. In order to achieve down-casting, upcasting is mandatory.**

**4. Down-casting has to be done explicitly.**

**syntax : (SubClassName) SuperClassReference;**

5-8-2021

**ClassCastException**

**1. If a object has been upcasted we have to downcast to same type else we get ClassCastException.**

**2. In other words, if one type of reference is upcasted and downcasted to some other type of reference we get ClassCastException.**

**3. If we Downcast, without upcasting even then we get ClassCastException.**

**4. In order to avoid ClassCastException we make use of instanceof operator.**

**instanceof**

**1. instanceof is an operator in order to check if an object is an instance of a specific class type or not.**

**2. In other words, instanceof is an operator in order to check if an object is having the properties of a specific class type or not.**

**3. instanceof will return boolean value.**

**syntax: object instanceof ClassName**

**6-8-2021**

**Method Binding**

**1. Associating or Mapping the Method Caller to it's Method Implementation or Definition is called Method Binding.**

**Polymorphism**

**1. Polymorphism means many forms.**

**2. The ability of a method to behave differently, when different objects are acting upon it.**

**3. The ability of a method to exhibit different forms, when different objects are acting upon it.**

**4. Different types of polymorphism are as follows:**

**i.  Compile time polymorphism.**

**ii. Run time polymorphism.**

**Compile time Polymorphism**

**1. Compile time Polymorphism is achieved with the help of Method Overloading.**

**2. Compile time Polymorphism is also referred as Early Binding or Static Binding.**

**3. Method Binding is happening at compile time, Hence we call Method Overloading as Early Binding or Static Binding or Compile time Polymorphism.**

**4. Out of so many overloaded methods, which method implementation should get executed is decided by the compiler during compile time.**

**Run time Polymorphism**

**1. Run time Polymorphism is achieved with the help of**

**i. Inheritance (IS-A Relationship)**

**ii. Method Overriding**

**iii. Upcasting**

**2. When we call an Overridden method on the superclass reference, the method implementation which gets executed is dependent on the subclass acting upon it.**

**3. Out of so many Overridden method, which method implementation should get executed is decided by the JVM at runtime.**

**4. Runtime Polymorphism is also called as Late Binding, Dynamic Binding or Dynamic Method Dispatch.**

**Note:**

**If we call a overridden method on the superclass reference, always the overridden method implementation only gets executed.**

9-8-2021

**abstract**

**1. abstract is a keyword which can be used with class and method.**

**2. A class which is not declared using abstract keyword is called as Concrete class or non-abstract class.**

**3. Concrete class can allow only concrete methods.**

**4. A class which is declared using abstract keyword is called as Abstract class.**

**5. Abstract class can allow both abstract and concrete methods.**

**6. Concrete method has both declaration and implementation/definition.**

**7. Abstract method has only declaration but no implementation.**

**8. All Abstract methods should be declared using abstract keyword.**

**Contract of Abstract or What should we do when a class extends abstract class:**

**1. When a class Inherits an abstract class, override all the abstract methods.**

**2. When a class Inherits an abstract class and if we do not want to override the inherited abstract method, then make the sub class as abstract class.**

**1. Can abstract class have constructors?**

**Yes. But we cannot invoke directly, it has to be invoked by the sub class constructor either implicitly or explicitly using super().**

**NOTE:**

**1. Can a class inherit an abstract class? -> YES -> Contract of Abstract**

**2. We cannot create an object of abstract class.**

**3. Abstract methods cannot be private.**

**4. Abstract methods cannot be static.**

**5. Abstract methods cannot be final.**

10-8-2021

Interface

---------

1. Interface is a Java Type Definition which has to be declared using interface keyword.

2. Interface is a media between 2 systems, wherein 1 system is the client/user and another system is object with resources/services.

syntax: interface InterfaceName

{

}

3. Interface can have variables, those variables are automatically

public, static and final.

4. Interface can allow only abstract methods, and those methods are automatically public and abstract.

5. class can achieve IS-A Relationship with an interface using implements keyword.

6. When a class implements an interface, mandatorily override the abstract method or make the class as abstract. (CONTRACT OF ABSTRACT)

7. While Overriding a method, Access Specifier/Modifier should be same or of Higher Visibility.

8. A class can implement any number of Interfaces (Multiple Interfaces).

9. A class can extend 1 class and implement any number of interfaces.

10. Interfaces does not contain Constructors.

11. We cannot create an object of interface.

11-8-2021

**Abstraction**

**1. The process of Hiding the Implementation details (unneccessary details) and showing only the functionalities (Behaviour) to the user with the help of an abstract class or interface is called as Abstraction.**

**2. The process of Hiding the Implementation and showing only the functionality is called as Abstraction.**

**3. Abstraction can be achieved by following the below rules:**

**i.   Abstract class or Interface with abstract methods.**

**ii.  Is-A (Inheritance).**

**iii. Method Overriding.**

**iv.  Upcasting.**

**4. Hide Implementation and Show Functionality.**

12-8-2021

**Singleton Design Pattern or Singleton Class**

**It is a Design Patten wherein we can create only a single instance or a single object for a class.**

**Rules in order to Develop Singleton Class/Singleton Design Pattern**

**1. Declare a Private Constructor.**

**2. Have a public static helper/Factory method to create a single Object.**

**3. Declare a private static non primitive reference variable.**

Java liberary

17-8-2021

**Java Libraries**

**1. It is a collection of pre-defined packages.**

**2. Each package/folder is collection of pre-defined classes and pre-defined interfaces.**

**3. Each class or interface is a collection of variables and methods.**

**All the pre-defined classes and interfaces are present inside a jar file called as**

**rt.jar (rt->runtime)**

**or**

**zip file -> src.zip**

**Write Any 6 pre-defined package Names.**

**1.java.lang**

**2.java.util**

**3.java.io**

**4.java.sql**

**5.java.applet**

**6.java.awt ..........**

**---------------------------------------**

**1. java.lang package**

**- This package is implicitly or automatically imported in all java classes and interfaces.**

**2. Object Class**

**- Object is a pre-defined class present in java.lang package.**

**- Object class is referred as super-most class in java.**

**- Object class is implicitly inherited by all java classes.**

**methods present in Object Class**

**1. protected Object clone()**

**2. public boolean equals(Object o)**

**3. public int hashCode()**

**4. public String toString()**

**5. public void wait()**

**6. public void wait(long a)**

**7. public void wait(long a, int b)**

**8. public void notify()**

**9. public void notifyAll()**

**10. public Class getClass()**

**11. protected void finalize()**

**----------------------------------------**

**1. toString()**

**syntax: public String toString()**

**- toString() returns the String representation of an Object in the below format.**

**FullyQualifiedClassName @ HexadecimalValueOfTheHashCode**

18-8-2021

**2. hashCode()**

**syntax: public int hashCode()**

**- hashCode() returns a unique id for an Object.**

**- In order to identify an Object Uniquely we make use of hashCode().**

**3. equals()**

**syntax: public boolean equals(Object obj)**

**- equals() is used for content comparison.**

**-equals() is used to compare the attributes of 2 objects.**

20-8-2021

**String**

**------**

**-> String is pre-defined final class present in java.lang package.**

**-> String Objects Immutable in Nature.**

**-> String is a Collection/Set of Characters.**

**-> String is also a Non-Primitive Datatype and default value is null.**

**-> String implements Serializable, Comparable, CharSequence**

**-> String Objects can be created in 2 ways.**

**1. literal (" ") double quotations**

**2. using new Operator**

**-------------------------**

**1. String s = "java";**

**2. String s = new String("java");**

**-> String Objects are stored inside a memory location called as String pool.**

**-> String Pool is further divided into 2 types**

**1. Constant Pool**

**2. Non-Constant Pool**

**-> Literal Objects are stored inside constant pool and constant pool does not allow duplicates.**

**-> String Objects created using new operator are stored inside Non-Constant Pool and Non-Constant Pool allows duplicates.**

**-> String class has automatically overridden 3 methods from Object class**

**1. toString()**

**2. hashCode()**

**3. equals()**

**-> toString() of the Object Class is Overridden in String Class to return the actual data passed to the Constructor during object creation.**

**-> hashCode() of the Object Class is overridden in String class to return a number based on the ASCII value.**

**-> equals() of the Object Class is Overridden in String Class to compare the contents of both the objects.**

**1. How String Objects are Immutable? Explain String Immutability Concept.**

**- When we re-initialize a String object, rather than modifying the same object, a new object is created and the reference pointing to the old object gets de-referenced and starts pointing to the newly created object.**

**This is String Immutability Concept.**

**Mutable Version of String**

**1. StringBuffer**

**2. StringBuilder**

**Exception Handling**

**Exception Handling**

**1. Exception is an event or an interruption which stops the execution of a program, wherein the below lines of code won't get executed.**

**2. In other words, Exception is a Runtime Interruption which can be HANDLED.**

**3. ERROR: Error is also a Mistake, Runtime Interruption which cannot be HANDLED (We have to Debug).**

**- Errors can occur in 2 ways**

**i.  Compilation error -> Syntax Mistakes**

**ii. Runtime Error -> Execute a class without main()**

**1.The Process of handling an Exception is called as Exception Handling.**

**2. Typically an Exception is Handled using Try Block and Catch Block.**

**try and catch block**

**1. The critical lines of code which gives an exception should be written inside the try block.**

**2. If there is try block, mandatorily catch block should be present and vice versa.**

**3. The Solution should always be written within the catch block.**

**4. catch block will be executed only if an exception occurs.**

**syntax:**

**try**

**{**

**// set of instructions**

**}**

**catch(ExceptionName referenceVariable)**

**{**

**// set of instructions**

**}**

**4. One try block can have any number of catch blocks.**

**5. There should not be any executable lines of code between try and catch block.**

**6. It is always a good practice to handle the superclass exception as the last catch block.**

**/\* internally**

**\* 1. An object of ArithmeticException is created**

**\* 2. the object is thrown**

**\* 3. it is caught by the catch block**

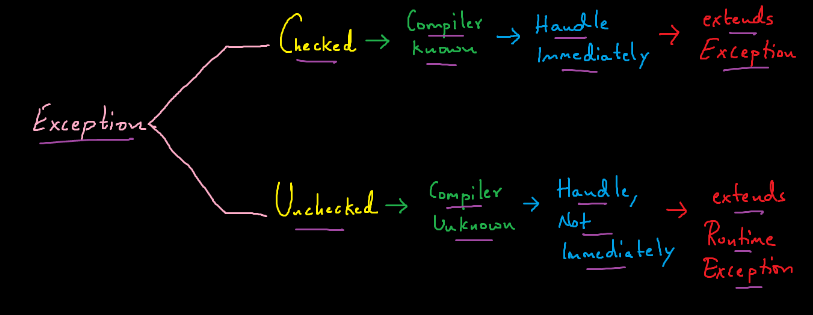
**\*/**

**Important methods present in Throwable Class:**

**1. printStackTrace(): This method is used to get the complete info about the Exception.**

**2. getMessage(): This method is used to return a small message about the Exception occurred.**

**Checked Exception and Unchecked Exception**



**throws**

**1. throws is an indication to the caller about the possibility of an Exception.**

**2. throws is used to propagate an Exception.**

**3. throws is generally used with Checked Exceptions.**

**4. Typically we use throws with methods, and we can use throws wrt Constructors as well.**

**Custom Exception or User-Defined Exception**

**------------------------------------------**

**1. Based on the project, it is sometimes necessary to create our own Exception and those exceptions which the user/Programmer creates are called as Custom Exception or User-Defined Exception.**

**Rules for working with Custom Exception:**

**----------------------------------------**

**1. Create a class with the Exception Name.**

**2. The Exception class which we created should either inherit Exception(Checked) or RuntimeException(Unchecked) class.**

**{Optionally Override getMessage()}**

**3. Create an object of the exception class created and invoke/throw the object of that Exception.**

**4. Handle it using try and catch block.**

**throw**

**-----------------------------------------------------**

**1. throw is a keyword in order to invoke an object of Exception.**

**2. throw is generally used with custom exception.**

**syntax: throw objectOfExceptionType ; // throw new ExceptionName();**

**finally block**

**-------------**

**1. The Set of Instructions which has to be executed all the time has to written within the finally block.**

**2. Finally Block is a block of code which gets executed all the time.**

**ie. irrespective of exception occurs or not.**

**syntax :  finally**

**{**

**}**

**void eat() throws ArithmeticException, ClassCastException, InvalidAgeException {**

**}**

**or**

**void eat() throws Exception {**

**}**

**throw -> Only 1 Object can be Thrown**

**Note :**

**1. In java we can have nested try and catch block.**

**2. We can have try and catch block within finally block as well.**