Java

> OOP language

-Encapsulation: Wrapping up of data and functions into a single unit.

-Abstraction: Displaying only essentials and hiding the details.

-Polymorphism: Ability of a message to be displayed in more than one form.

-Inheritance: Ablity of a class to derive features from another class.

>Class-A class is a template or blueprint from which objects are created.

>Object-An object is the instance(result) of a class .An object is a real-world entity.

>Datatypes

-Non Primitive datatypes-Class, object, array, string, and interface are called non-primitive data types in Java. These data types are not predefined in Java.

Primitive datatypes-There are 7 primitive data types: string, number, bigint, boolean, undefined, symbol, and null.

>Access modifiers

-Private: Can be accessed only within the class. It cannot be accessed from outside the class.

-Default: Can be accessed only within the package. Cannot be accessed from outside the package.

-Protected: The access level of a protected modifier is within the package and outside the package through child class.

-Public: The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

>Loops

-for loop:

for(initialization; condition; increment/decrement){

//statement or code to be executed

}

-while loop:

while (condition){

//code to be executed

Increment / decrement statement

}

-do while loop:

do{

//code to be executed / loop body

//update statement

}while (condition);

>Decision making statements:

-if:

if(condition)

{

// Statements to execute if

// condition is true

}

-if else:

if (condition)

{

// Executes this block if

// condition is true

}

else

{

// Executes this block if

// condition is false

}

-switch:

switch(expression){

case value1:

//code to be executed;

break; //optional

case value2:

//code to be executed;

break; //optional

......

default:

code to be executed if all cases are not matched;

}

>Conditional operators

-Logical AND:&&

-Logical OR-||

>Increment and Decrement operator

-If you use the ++ operator as a prefix, ++a, the value of a is incremented by 1; then it returns the value.

-If you use the ++ operator as a postfix,a++, the original value of a is returned first; then var is incremented by 1.

>Array

-It is an object which contains elements of a similar data type.

-Arrays work our w.r.t.zero.

eg:A[5]={6,7,8,9,1}

-2 types of array:1D & 2D

>Constructor

-Used to create an object.

-By default, any class will be supplied with zero argument constructor.

-types:

Default, No-argument constructor and Parameterized.

>This operator:

-It is a reference variable which points to current objects.

-It can be used in instance variables, passed as an argument, can be used with constructor.

>Super:

-It is a reference variable which point the super class or parent object class.

-Can be used with methods, constructors, variables.

>Static:

-Not a instance variable.

-It is a class variable.

-Static can be global variable,classes,objects,blocks,methods.

>Blocks:

-A block in Java is a set of code enclosed within curly braces { } within any class, method, or constructor.

-It begins with an opening brace ( { ) and ends with an closing braces ( } ).

>Methods:

-A method is a block of code or collection of statements or a set of code grouped together to perform a certain task or operation.

-method declaration provides information about method attributes, such as visibility, return-type, name, and arguments.

>Method Overloading:

-If a class has multiple methods having same name but different in parameters, it is known as Method Overloading.

-We do not consider return type.

-There are two ways to overload the method in java or we only consider the following things in method overloading:-

\*By changing number of arguments

\*By changing the data type

>Method Overriding:

-If subclass (child class) has the same method as declared in the parent class, it is known as method overriding in Java.

>Abstraction:

-Hiding implementation to outside world.

-Use keyword abstract.

-In abstract class, there should be atleast one method which is abstract.

>Interfaces:

-The interface in Java is a mechanism to achieve abstraction.

-There can be only abstract methods in the Java interface, not method body.

-It is used to achieve abstraction and multiple inheritance in Java.

>Runtime errors:

-The runtime errors are the errors which are not generated by the compiler and produce an unpredictable result at the execution time.

>Exception:

-Handle runtime errors so that normal flow of the application can be maintained.

-Types of exception:-

\*ArithmeticException

It is thrown when an exceptional condition has occurred in an arithmetic operation.

\*ArrayIndexOutOfBoundsException

It is thrown to indicate that an array has been accessed with an illegal index. The index is either negative or greater than or equal to the size of the array.

\*ClassNotFoundException

This Exception is raised when we try to access a class whose definition is not found

\*FileNotFoundException

This Exception is raised when a file is not accessible or does not open.

\*IOException

It is thrown when an input-output operation failed or interrupted

\*InterruptedException

It is thrown when a thread is waiting, sleeping, or doing some processing, and it is interrupted.

\*NoSuchFieldException

It is thrown when a class does not contain the field (or variable) specified

\*NoSuchMethodException

It is thrown when accessing a method which is not found.

\*NullPointerException

This exception is raised when referring to the members of a null object. Null represents nothing

\*NumberFormatException

This exception is raised when a method could not convert a string into a numeric format.

\*RuntimeException

This represents any exception which occurs during runtime.

\*StringIndexOutOfBoundsException

It is thrown by String class methods to indicate that an index is either negative or greater than the size of the string.

>FILE HANDLING I/O:-

-A named location used to store related information is known as a File.

-There are several File Operations like creating a new File, getting information about File, writing into a File, reading from a File and deleting a File.

>Stream:

-A series of data is referred to as a stream.

-Stream is classified into two types, i.e., Byte Stream and Character Stream.

>Input Stream and Output Stream::

-Input Stream:-

Types of InputStream are:

\*FileInputStream

\*ByteArrayInputStream

\*FilterInputStream

\*ObjectInputStream

-Output Stream:-

Types of OutputStream are:

\*FileOutputStream

\*ByteArrayOutputStream

\*FilterOutputStream

\*ObjectOutputStream

>BufferedReader-

BufferedReader is a class in Java that reads text from a character-input stream, buffering characters so as to provide for the efficient reading of characters, lines and arrays.

>BufferedWriter-

A BufferedWriter on the other hand is a java class that writes text to a character-output stream, while buffering characters so as to provide for the efficient writing of single characters, strings and arrays.

>InputStreamReader-

InputStream Read data from the source once at a time.

>OutputStreamReader-

OutputStream Write Data to the destination once at a time.

>FileReader and FileWriter-

- FileWriter and FileReader classes are used to write and read data from text files (they are Character Stream classes).

- Do not use the FileInputStream and FileOutputStream classes if you have to read and write any textual information as these are Byte stream classes.

>Collections:

-It is a class in java.

-It can contain heterogenous elements and can modify the size dynamically.

-Collection is an interface.

>Interfaces:

-An interface in Java is a blueprint of a class.

-It has static constants and abstract methods.

>Java Collections:

These are:

\*List:

-List contains your ordered elements but can have duplicate also.

-The list is an ordered collection, also known as a sequence.

-Because List is organized, you have complete control over where list items are placed in a List.

-One thing to note here is that the Java List collection can only contain objects.

-Arraylist,Linkedlist

\*Set:

Java Set is a collection construct that, by definition, contains unique elements — that is, no duplicates.

- The Java Set collection can only contain objects, and it is not an ordered list, that means it does not care about the order of the elements.

- Because Set is an interface, you cannot instantiate it directly.

\*Queue:

Java Queue is a collection that works on FIFO (First In First Out) principle.

-The elements that are added first will be removed first from the queue.

-LinkedList and Priority Queue are the most common types of Queue implementations in Java.

\*Map:

-The map is a convenient collection construct that you can use to associate one object (key) with another object (value).

-As you can imagine, the key of the Map must be unique and can be used later to retrieve values.

-Different implementations of the Map are HashMap, TreeMap, LinkedHashMap, etc.

-HashMap Java is the common Map type used by programmers.

>Wrapper class:

-The wrapper class implements the technique to convert the primitive into object and object into primitive.

-There is a concept of autoboxing and unboxing in the wrapper class, which transform the primitives into objects and objects into primitives automatically.

-A Wrapper class is a class whose object wraps or contains primitive data types.

-When we create an object to a wrapper class, it contains a field and in this field, we can store primitive data types.

-Classes of this type like Integer,Float,Double,Long,Character,Boolean.

>Generics:

-Generics enable types (classes and interfaces) to be parameters when defining classes, interfaces and methods .

-<T> is the data type parameter. <T> , <N> , and <E> are some of the letters used for data type parameters according to Java conventions.

-In the above example, you can pass it a specific data type when creating a GenericClass object.

>Threads:

-A thread, in the context of Java, is the path followed when executing a program.

-It is a class that allows to perform multitasking.

InnerClass:

-Java inner class or nested class is a class that is declared inside the class or interface.

-We use inner classes to logically group classes and interfaces in one place to be more readable and maintainable.

syntax:-

class Java\_Outer\_class{

//code

class Java\_Inner\_class{

//code

}

}

JUNIT:-Its a unit testing framework.

RTM-

-Requirement Traceability Matrix (RTM) is a document that maps and traces user requirement with test cases.

-The main purpose of Requirement Traceability Matrix is to validate that all requirements are checked via test cases such that no functionality is unchecked during Software testing.

\*Test Plan:

-A Test Plan is defined as a document which outlines the scope, objective, method and weight on a software testing task

\*Test Stratergy:

-Test Strategy in software testing is defined as a set of guiding principles that determines the test design & regulates how the software testing process will be done.

\*Test Case:

-A test case holds a description of a set of steps to be executed by a tester in a bid to validate a scenario.

\*Test Scenario:

-A test scenario normally contains multiple test cases which contain details on how to test the scenario.

-The test scenarios are those derived from the use case and give the one-line information about what to test.

\*Test Script:

-Test Script refers to a set of instructions or short program which tests some portion of the functionality of a software product.

\*Test Data:

-Test Data in Software Testing is the input given to a software program during test execution.

>Several phases of testing includes:

-Integration testing, Acceptance testing, Regression testing, Whitebox testing, Blackbox testing, Unit testing, User acceptance testing, Preproduction testing, Production testing, Smock testing, Performance production testing

>Annotations for Junit testing:

1. @Test: It is used to specify the test method.

2. @BeforeClass: It is used to specify that method will be called only once, before starting all the test cases.

3. @AfterClass: It is used to specify that method will be called only once, after finishing all the test cases.

4. @Before: It is used to specify that method will be called before each test case.

5. @After: It is used to specify that method will be called after each test case.

6. @Ignore: It is used to ignore the test case.