BU INTERVIEW

1. **Tell me about yourself ?**
   -

2. **What have you learnt - Stage 1 to 4?**

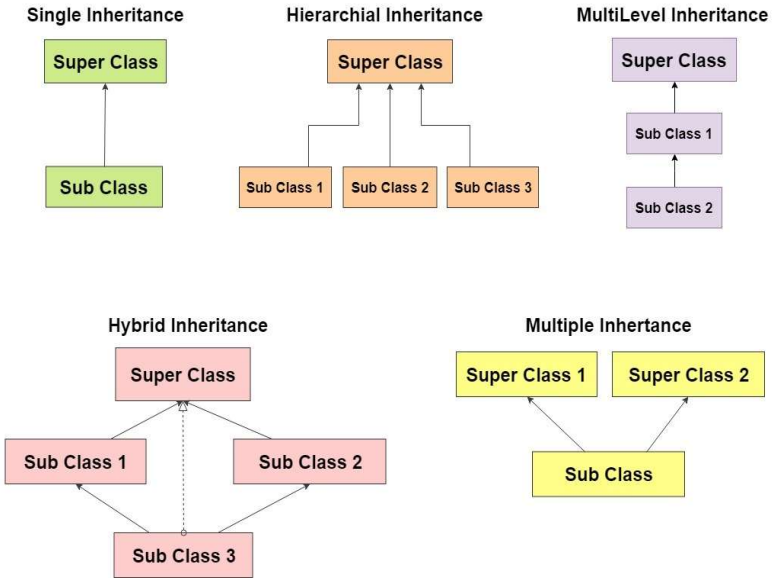
   - User Interface Design, SQL Programming, Unix and Shell Scripting, Java, Design Patter and Principles, Data Structure and Algorithms, Spring Core & Maven, Unit Testing, Spring MVC & Spring Boot, React, Microservices, Application debugging, Jenkins, GIT and Jira, Cloud and AWS.

3. **OOPs Concept ?**

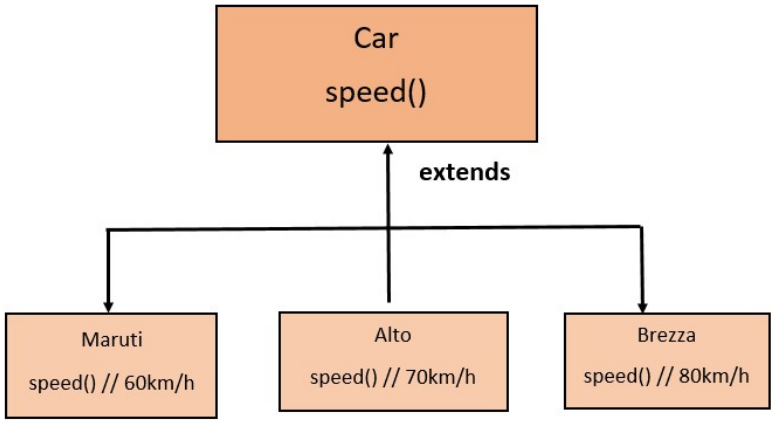
   - Object-oriented programming aims to implement real-world entities like inheritance, hiding, polymorphism etc in programming.
   - The main aim of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function.
   - **Class -** A class is a user defined blueprint or prototype from which objects are created. It represents the set of properties or methods that are common to all objects of one type.
   - **Object -** Object is a basic unit of Object-Oriented Programming and represents the real-life entities. Object can be defined as an instance of a class, which contains both the data and the function, which operates on the data.

   Let us now discuss 4 pillars of OOPS:

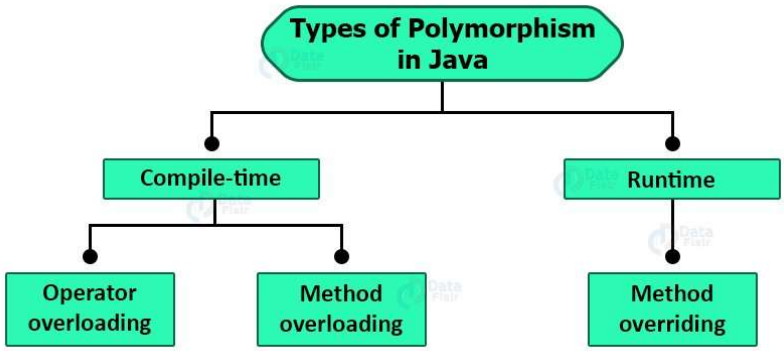
   - **Inheritance -** It is the mechanism in java by which one class is allowed to inherit the features(fields and methods) of another class. Inheritance supports the concept of “reusability”.



- **Polymorphism -** The ability of a variable, object, or function to take on multiple forms.



In Java polymorphism is mainly divided into two types:



i. **Compile-time Polymorphism –** Static Polymorphism

a) **Method Overloading:** When there are multiple functions with the same name but different parameters then these functions are said to be overloaded. Functions can be overloaded by change in the number of arguments or/and a change in the type of arguments.

b) Java doesn't support the Operator Overloading.

ii. **Runtime Polymorphism -** Dynamic Polymorphism

- **Method Overriding:**
  - Overriding is a feature that allows a subclass or child class to provide a specific implementation of a method that is already provided by one of its super-classes or parent classes.
  - When a method in a subclass has the same name, same parameters or signature, and same return type(or sub-type) as a method in its super-class, then the method in the subclass is said to *override* the method in the super-class.

Overriding

```
class Dog{
    public void bark(){
        System.out.println("woof ");
    }
}
class Hound extends Dog{
    public void sniff(){
        System.out.println("sniff ");
    }

    public void bark(){
        System.out.println("bowl");
    }
}
```

Same Method Name, Same parameter

Overloading

```
class Dog{
    public void bark(){
        System.out.println("woof ");
    }

    //overloading method
    public void bark(int num){
        for(int i=0; i<num; i++)
            System.out.println("woof ");
    }
}
```

Same Method Name, Different Parameter

Note -> Method hiding means subclass has defined a class method with the same signature as a class method in the superclass. In that case the method of superclass is hidden by the subclass.

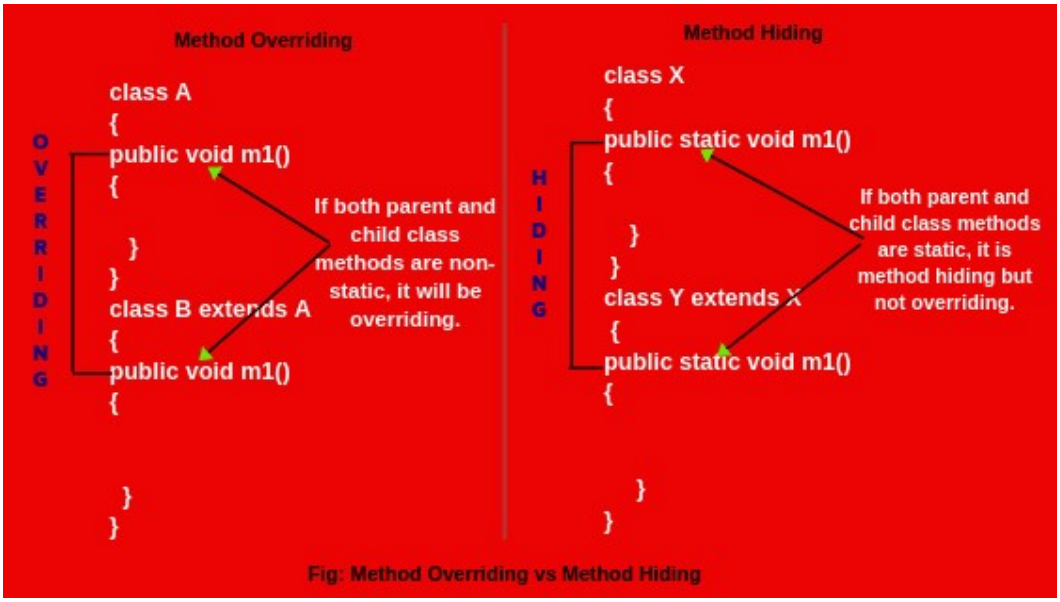
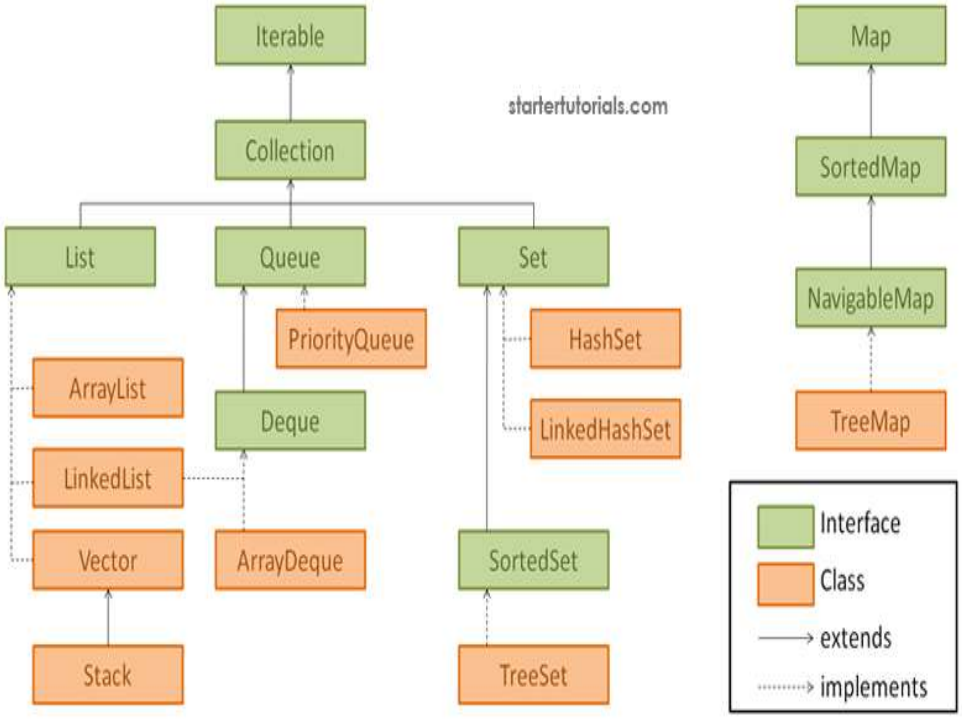


Fig: Method Overriding vs Method Hiding

- **Abstraction –**
    - Data Abstraction is the property by virtue of which only the essential details are displayed to the user. The trivial or the non-essentials units are not displayed to the user. Ex: A car is viewed as a car rather than its individual components.
    - In java, abstraction is achieved by interfaces and abstract classes. We can achieve 100% abstraction using interfaces.
- **Encapsulation -** It is defined as the wrapping up of data under a single unit. It is the mechanism that binds together code and the data it manipulates. Another way to think about encapsulation is, it is a protective shield that prevents the data from being accessed by the code outside this shield.

4. Collections -> Types of collections

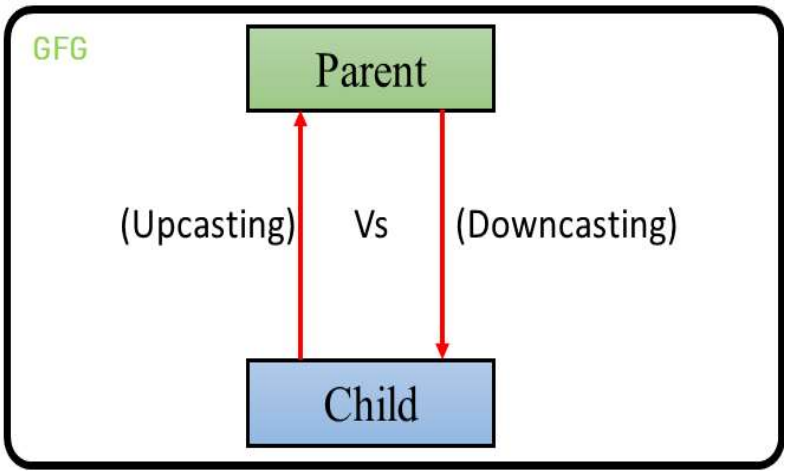
- Java defines a collection as *an object that represents a group of objects*.
- The Collection interface (**java.util.Collection**) and Map interface (**java.util.Map**) are the two main “root” interfaces of Java collection classes.
- The utility package, (java.util) contains all the classes and interfaces that are required by the collection framework.

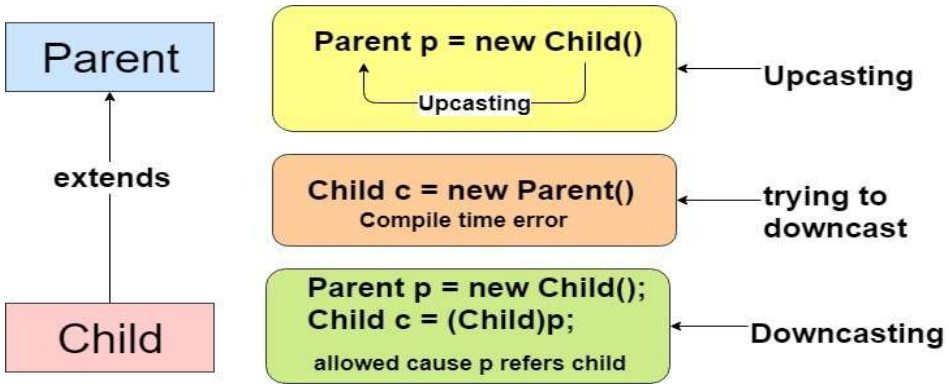


5. **Inheritance type**

   Already Answered. Click here to see.

6. **Upcasting & downcasting**





- **Typecasting -** The conversion of one data type to another datatype implicitly or explicitly. Just like the data types, the objects can also be type casted.

- **Upcasting -** Upcasting can be done implicitly. Upcasting gives us the flexibility to access the parent class members, but it is not possible to access all the child class members using this feature.

- **Downcasting -** It means the typecasting of a parent object to a child object. Downcasting cannot be implicit. Downcasting is used when we need to develop a code that accesses behaviours of the child class.

Note – In Upcasting, we can access some specified methods of the child class and all method of parent class while in Downcasting, all the methods and variables of both classes can be accessed.

7. Difference b/w LL and ArrayList

ArrayList	LinkedList
ArrayList internally uses a dynamic array to store the elements	LinkedList internally uses a doubly linked list to store the elements
Manipulation with ArrayList is slow because it internally uses an array. If any element is removed from the array, all the bits are shifted in memory.	Manipulation with LinkedList is faster than ArrayList because it uses a doubly linked list, so no bit shifting is required in memory.
ArrayList consumes less memory than LinkedList	A LinkedList consumes more memory than an ArrayList because it also stores the next and previous references along with the data.
An ArrayList class can **act as a list** only because it implements List only.	LinkedList class can **act as a list and queue** both because it implements List and Deque interfaces.
ArrayList is better for storing and accessing data.	LinkedList is **better for manipulating** data.

8. Default Size of Map

The default initial capacity of the *HashMap* is 2^4 i.e., **16**. The capacity of the *HashMap* is doubled each time it reaches the threshold. i.e., the capacity is increased to 2^5=32, 2^6=64, 2^7=128…… when the threshold is reached.

Load factor is the measure which decides when to increase the capacity of the HashMap. The default load factor is 0.75f.

For example, if the *HashMap* is created with initial capacity of 16 and load factor of 0.75f, then threshold will be,

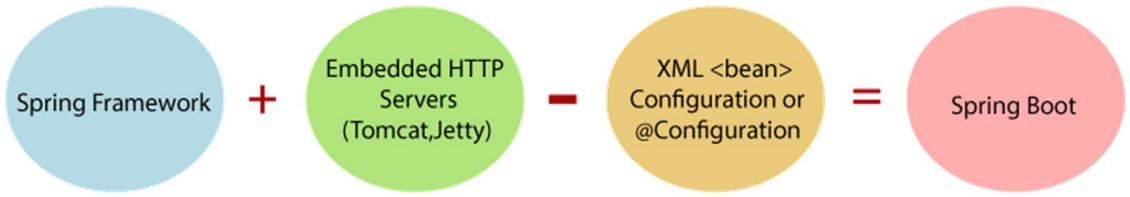
Threshold = 16 * 0.75 = 12

That means, the capacity of the *HashMap* is increased from 16 to 32 after the 12th element (key-value pair) is added into the *HashMap*.

9. ArrayList default size

Whenever an instance of ArrayList in Java is created then by default the capacity of Arraylist is 10. Since ArrayList is a growable array, it automatically resizes itself whenever a number of elements in ArrayList grow beyond a threshold.

10. Spring & Spring boot overview



What is Spring ->

- The Spring Framework (Spring) is an open-source application framework that provides infrastructure support for developing Java applications.

Why Spring ->

- Spring removes tedious configuration work so that developers can focus on writing business logic. Spring handles the infrastructure so developers can focus on the application.

How Spring Works ->

A web application (layered architecture) commonly includes three layers:

- **Presentation/view layer (UI) -** This is the outermost layer which handles the presentation of content and interaction with the user.
- **Business logic layer -** The central layer that deals with the logic of a program.
- **Data access layer -** The deep layer that deals with data retrieval from sources.

Each layer is dependent on the other for an application to work. Without a Spring Framework, application code tends to be tightly coupled (interdependent), which is not considered good coding practice.

Spring's core logic is dependency injection. Dependency injection is a programming pattern that allows developers to build more decoupled architectures. Dependency injection means that Spring understands the different Java annotations that a developer puts on top of classes. Spring knows that the developer wants to create an instance of a class and that Spring should manage it. Spring also understands the dependency and makes sure that all instances created have properly populated dependencies.

What is Spring Boot ->

- Spring Boot is a project that is built on the top of the Spring Framework. It provides an easier and faster way to set up, configure, and run both simple and web-based applications.
- It is a Spring module that provides the RAD (Rapid Application Development) feature to the Spring Framework. It is used to create a stand-alone Spring-based application that you can just run because it needs minimal Spring configuration.

Why Spring Boot ->

- To avoid complex XML configuration in Spring.
- To develop a production ready Spring applications in an easier way
- It provides a powerful batch processing and manages REST endpoints.
- To reduce the development time and run the application independently

How Spring Boot Works ->

- Spring Boot automatically configures your application based on the dependencies you have added to the project by using **@EnableAutoConfiguration** annotation. For example, if MySQL database is on your classpath, but you have not configured any database connection, then Spring Boot auto-configures an in-memory database.
- The entry point of the spring boot application is the class contains **@SpringBootApplication** annotation and the main method.

11. JDBC template

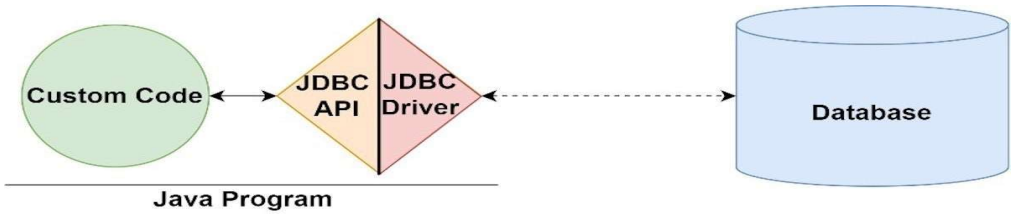
JDBC(Java Database Connectivity) is an API(Application programming interface) used in java programming to interact with databases.

Usage -

- Making a connection to a database.
- Creating SQL or MySQL statements.
- Executing SQL or MySQL queries in the database.
- Viewing & modifying the resulting records.

Component of JDBC –

- **JDBC drivers** are client-side adapters (installed on the client machine, not on the server) that convert requests from Java programs to a protocol that the DBMS can understand.
- **ResultSet** – These objects hold data retrieved from a database after you execute an SQL query using Statement objects. It acts as an iterator to allow you to move through its data.



- JDBC Template is a central class in the JDBC core package that simplifies the use of JDBC and helps to avoid common errors. It internally uses JDBC API and eliminates a lot of problems with JDBC API.
- It executes SQL queries or updates, initiating iteration over ResultSets and catching JDBC exceptions and translating them to the generic.

Statement - It is used for accessing your database. Statement interface cannot accept parameters and useful when you are using static SQL statements at runtime. If you want to run SQL query only once, then this interface is preferred over PreparedStatement.

PreparedStatement - It is used when you want to use SQL statements many times. The PreparedStatement interface accepts input parameters at runtime.

CallableStatement - It is used when you want to use the database stored procedures. CallableStatement can accept runtime input parameters.

12. HTML vs HTML5

HTML	HTML5
➢ No support for audio and video ➢ Does not have any standard process to handle the codes that have structural error ➢ It does not allow JavaScript to run in the browser ➢ Makes the use of vector graphics possible when concurrently used with Flash, Silverlight or some other third party plugins ➢ HTML is not a mobile friendly markup language ➢ Doctype declaration and character encoding are way too long ➢ Low storage efficiency	➢ Supports high quality audio and video ➢ Have support for persistent error handler through improvised error handling process ➢ It allows JavaScript to run in the background ➢ Scalable Vector Graphic(SVG) comes as an integral part of this version without any third party plugins ➢ HTML5 is quite mobile friendly ➢ Doctype declaration and character encoding are very short and simple ➢ High storage efficiency

13. CSS

- CSS stands for Cascading Style Sheets
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media
- CSS saves a lot of work. It can control the layout of multiple web pages all at once
- External stylesheets are stored in .css files.

14. What is Multithreading

Multithreading is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilization of CPU. Each part of such program is called a thread. So, threads are light-weight processes within a process.

Threads can be created by using **two** mechanisms :

(i). Extending the Thread class

We create a class that extends the java.lang.Thread class. This class overrides the run() method available in the Thread class. A thread begins its life inside run() method.

temp.java > Main > main

```java
class MyThread extends Thread {
    public void run()
    {
        try {
            // Displaying the thread that is running
            System.out.println("Thread " + Thread.currentThread().getId()+ " is running");
        }
        catch (Exception e) {
            // Throwing an exception
            System.out.println("Exception is caught");
        }
    }
}

// Main Class
public class Main {
    public static void main(String[] args)
    {
        MyThread t1 = new MyThread();
        MyThread t2 = new MyThread();

        t1.start() // will start the thread
        t2.start();
    }
}
```

Output

```
Thread 10 is running
Thread 11 is running
```

(ii). Implementing the Runnable Interface

We create a new class which implements java.lang.Runnable interface and override run() method. Then we instantiate a Thread object and call start() method on this object.

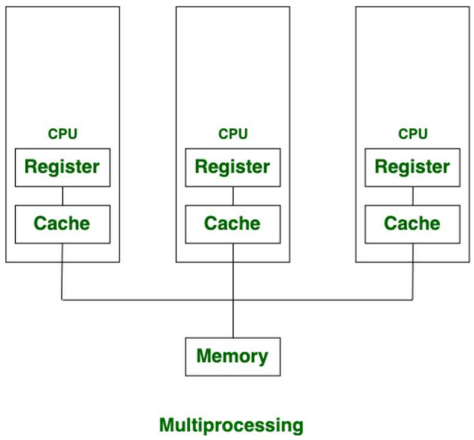
```java
class MyThread implements Runnable {
    public void run()
    {
        try {
            // Displaying the thread that is running
            System.out.println("Thread " + Thread.currentThread().getId()+ " is running");
        }
        catch (Exception e) {
            // Throwing an exception
            System.out.println("Exception is caught");
        }
    }
}

// Main Class
public class Main {
    public static void main(String[] args)
    {
        Thread t1 = new Thread(new MyThread());
        Thread t2 = new Thread(new MyThread());

        t1.start() // will start the thread
        t2.start();
    }
}
```

Output

```
Thread 10 is running
Thread 11 is running
```

15. Multi-processing

Multiprocessing is a system that has more than one or two processors. In Multiprocessing, CPUs are added for increasing computing speed of the system. Because of Multiprocessing, there are many processes are executed simultaneously.



Multiprocessing

Difference between Multiprocessing and Multithreading:

S.NO	Multiprocessing	Multithreading
1.	In Multiprocessing, CPUs are added for increasing computing power.	While In Multithreading, many threads are created of a single process for increasing computing power.
2.	In Multiprocessing, Many processes are executed simultaneously.	While in multithreading, many threads of a process are executed simultaneously.
3.	Multiprocessing are classified into **Symmetric** and **Asymmetric**.	While Multithreading is not classified in any categories.
4.	In Multiprocessing, Process creation is a time-consuming process.	While in Multithreading, process creation is according to economical.
5.	In Multiprocessing, every process owned a separate address space.	While in Multithreading, a common address space is shared by all the threads.

16. DDL vs DML

DDL	DML
It stands for Data Definition Language.	It stands for Data Manipulation Language.
It is used to create database schema and can be used to define some constraints as well.	It is used to add, retrieve or update the data.
It basically defines the column (Attributes) of the table.	It add or update the row of the table. These rows are called as tuple.
It doesn't have any further classification.	It is further classified into Procedural and Non-Procedural DML.
Basic command present in DDL are CREATE, DROP, RENAME, ALTER etc.	BASIC command present in DML are UPDATE, INSERT, MERGE etc.
DDL does not use WHERE clause in its statement.	While DML uses WHERE clause in its statement.

DDL commands can't be rolled back but DML can be.

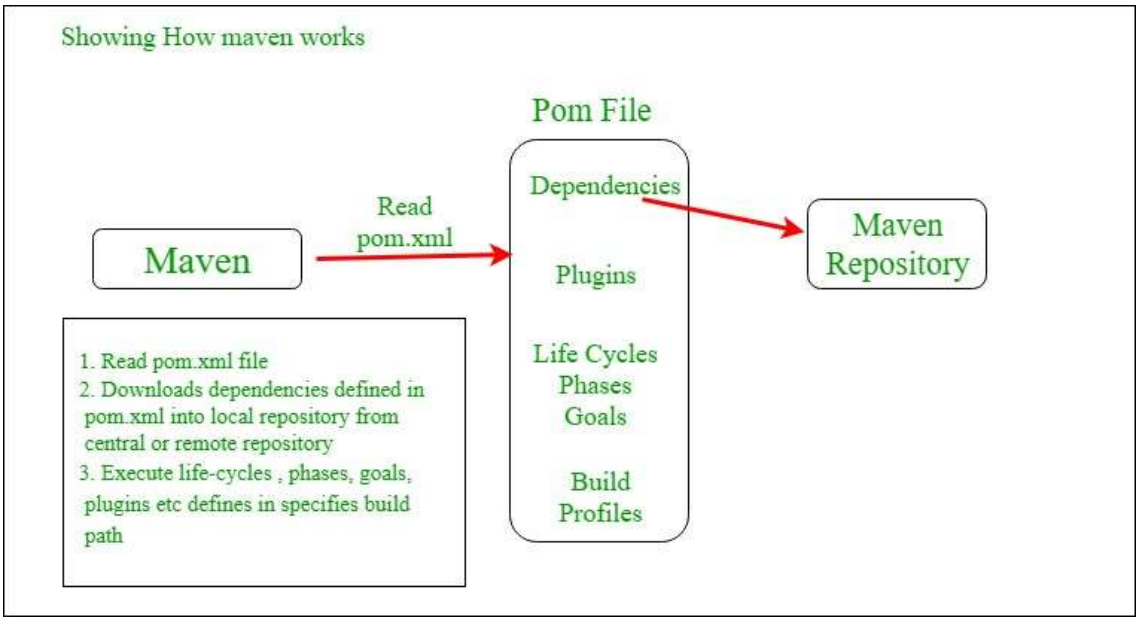
17. Truncate and Drop Table

DROP is a DDL(Data Definition Language) command and is used to remove table definition and indexes, data, constraints, triggers etc for that table. Performance-wise the DROP command is quick to perform but slower than TRUNCATE. Can't be rolled back. In the DROP command, table space is freed from memory.

TRUNCATE is a DDL(Data Definition Language) command. It is used to delete all the rows from the table. Like the DROP command, the TRUNCATE command also does not contain a WHERE clause. The TRUNCATE command is faster than both the DROP. Can't be rolled back. It does not free the table space from memory.

18. Maven

- Maven is a powerful project management tool that is based on POM (project object model). It is used for projects build, dependency and documentation. It simplifies the build process.
- In short terms we can tell maven is a tool that can be used for building and managing any Java-based project. With the help of Maven, we can build any number of projects into output types like the JAR, WAR etc without doing any scripting.
- When working on a java project and that project contains a lot of dependencies, builds, requirement, then handling all those things manually is very difficult and tiresome. Thus, using some tool which can do these works is very helpful.
- Maven is such a build management tool which can do all the things like adding dependencies, managing the classpath to project, generating war and jar file automatically and many other things.



19. Dependency Injection

- Classes often require references to other classes. For example, a Car class might need a reference to an Engine class. These required classes are called dependencies, and in this example the Car class is dependent on having an instance of the Engine class to run.
- When class A uses some functionality of class B, then it's said that class A has a dependency of class B. In Java, before we can use methods of other classes, we first need to create the object of that class (i.e., class A needs to create an instance of class B).
- **So, transferring the task of creating the object to someone else and directly using the dependency is called dependency injection.**

Role of DI :-

i. Create the objects

ii. Know which classes require those objects

iii. And provide them all those objects

iv. If there is any change in objects, then DI looks into it and it should not concern the class using those objects. This way if the objects change in

the future, then its DI's responsibility to provide the appropriate objects to the class.

20. **Loose Coupling -** Loose coupling means that the degree of dependency between two components is very low.

21. **Tight Coupling -** Tight coupling means that the degree of dependency between two components is very high.

22. **Why is there no multiple inheritance in Java, but implementing multiple interfaces is allowed?**

    Because interfaces specify only what the class is doing, not how it is doing it. The problem with multiple inheritance is that two classes may define different ways of doing the same thing, and the subclass can't choose which one to pick.

23. **What is difference between OBJECT ORIENTED PROGRAMMING and OBJECT BASED PROGRAMMING?**

Object based Programming	Object oriented Programming
• Object-based language doesn't support all the features of OOPs like Polymorphism and Inheritance. • Object-based language has built-in object like JavaScript has window object. • Examples : JavaScript, VB etc.	• Object-oriented language supports all the features of OOPs . • Object-oriented language doesn't have built-in object. • Examples : C++, C#, Java etc.

24. **Where is OBJECT BASED PROGRAMMING used ?**

OBP languages can build actual objects from a constructor function, and it has almost any feature that any object could have such as Constructor, functions (Methods), Properties,

Instances etc. So, they are used where most of the work is needed to be done by object but doesn't require much feature of OOPs language.

25. Difference between HTML4 and HTML5 ?

Parameters	HTML4	HTML5
Structure	Uses common structure like header and footer	Uses new structures like drag and drop.
DOCTYPE declaration	Very lengthy and refers to an external resource	Simple and in one line
Applets display	Applet tag used to display applets was removed	Object tag introduced to display applet type items.
Inaccurate syntax handling	Cannot handle	Can handle
Acronym tag	Removed	A new tag <abbr> introduced
Compatibility	Compatible with almost all browsers	Not compatible with all browsers
Multimedia supporting tags	Not present	Present
Storage	Browser cache can be used as temporary storage	Application cache, Web Sql DB and Web storage can be used as client storage.
Use of cookies	Yes	No. Local storage instead of cookies
Drawing of shapes	Not possible	Possible
2D drawing	Flash is required	Flash not required

26. What is new in Java 8?

Lambda expression – Adds functional processing capability to Java.

Stream API – New stream API to facilitate pipeline processing.

Date Time API – Improved date time API.

Default method – Interface to have default method implementation.

He did not ask me what you have learnt. He started with project directly. I told him we haven't done that yet completely; we just have the topic. He asked to explain your part in the project. He's not asking direct stuff. He's asking ques like why, not what. If you don't know something, just tell him you don't know.

27. Stored procedure ->

- The stored procedure is created and stored in the Java DB database as a database object.
- The procedure is invoked (or called) using a SQL command, or from a Java program using JDBC API.

- A stored procedure is a set of Structured Query Language (SQL) statements with an assigned name, which are stored in a relational database management.

- A stored procedure is a prepared SQL code that you can save, so the code can be reused repeatedly. So if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.

Example

```
CREATE PROCEDURE SelectAllCustomers
AS
SELECT * FROM Customers
GO;
```

Execute the stored procedure above as follows:

Example

```
EXEC SelectAllCustomers;
```

28. Spring framework ->

Already answered here.

29. HashSet ->

- HashSet contains only unique elements.
- It implements the Set interface.

```java
    public static void main(String[] args) {

        // Creating hashset
        HashSet<Integer> set = new HashSet<>();

        // Adding element add() method
        set.add(2);
        set.add(4);
        set.add(6);

        // Checking if elements exist
        set.contains(4) // returns True

        // remove a element
        set.remove(2) // will remove 2

        // Loop through hashset
        for (Integer i : set) {
            System.out.println(i);
        }
    }
```

30. HTML CSS

```html
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="styles.css">
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

31. Junit

- JUnit is a unit testing open-source framework for the Java programming language. Java Developers use this framework to write and execute automated tests.
- Annotations used in Junit :-

@Test	Tells JUnit which public void method can be run as a test case
@Before	To execute some statement before each test case
@After	To execute some statement after each test case
@Ignores	Used to ignore some statements during test execution
@BeforeClass	Used to execute a statement before all the test cases
@AfterClass	Used to execute a statement after all the test cases
@Test (time out =500)	Used to set some timeout while executing the test
@Test (expected=Illegal ArgumentException.class)	Used to handle some exception during test execution

- Assertions in Junit :-

```
//Check that two objects are equal
assertEquals(str1, str2);

//Check that a condition is true
assertTrue (val1 < val2);

//Check that a condition is false
assertFalse(val1 > val2);

//Check that an object isn't null
assertNotNull(str1);

//Check that an object is null
assertNull(str3);

//Check if two object references point to the same object
assertSame(str4,str5);

//Check if two object references not point to the same object
assertNotSame(str1,str3);

//Check whether two arrays are equal to each other.
assertArrayEquals(expectedArray, resultArray);
```

32. Code quality tool
https://snyk.io/learn/code-review/java-tools/

33. Abstraction and interface

- Interface provides 100% abstraction while abstract class between 0 to 100%.
- Use abstract class when you must provide some common functionalities to the child classes. Like in case of banks, every bank has same logic for withdraw, deposit money and there might be some functions that differ, but most of them are same.

```java
abstract class Bank {
    public void deposit(int m) {
        System.out.println("Some common logic");
    }

    public void withdraw(int m) {
        System.out.println("Some commn logic");
    }

    public abstract void calROI(int m);
}
```

Abstract Class

```java
class Axis extends Bank {
    @Override
    public void calROI(int m) {
        System.out.println("Axis Bank calROI() logic");
    }
}

class Paytm extends Bank {

    @Override
    public void calROI(int m) {
        System.out.println("Paytm Bank calROI() logic");
    }
}

public class Main {

    public static void main(String[] args) {
        Bank axis = new Axis();
        Bank paytm = new Paytm();

        axis.deposit(100);
        paytm.deposit(100);

        axis.calROI(100);
        paytm.calROI(100);
    }
}
```

Implementation of Abstract Class

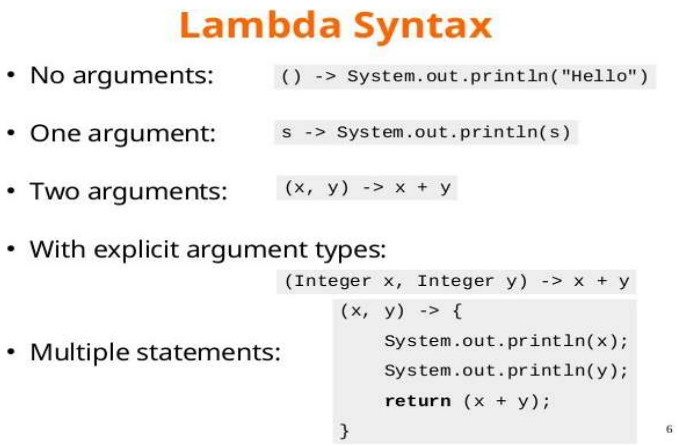
- Use interfaces when we just want to give child classes, some set of contracts to must implement the methods. Here methods might be same but the logic inside them differs in most of the child class implementation.

34. About Project

35. Lambda expression

- A lambda expression is a short block of code which takes in parameters and returns a value. Lambda expressions are like methods, but they do not need a name and they can be implemented right in the body of a method.

- It provides a clear and concise way to represent one method interface using an expression. It is very useful in collection library. It helps to iterate, filter and extract data from collection.



36. Streams in java – It is a sequence of data.

- **map:** The map method is used to returns a stream consisting of the results of applying the given function to the elements of this stream.

- **filter:** The filter method is used to select elements as per condition from the stream.

- **forEach:** The forEach method is used to iterate through every element of the stream.

- **reduce:** The reduce method is used to reduce the elements of a stream to a single value.

```java
public static boolean isEven(int n) {
    if (n % 2 == 0)
        return true;
    return false;
}

public static void main(String[] args) {

    List<Integer> number = Arrays.asList(1, 2, 3, 4, 5);
    int[] arr = {1,2,3,4,5};

    // map with Collection object like list
    List<Integer> square = number.map(x -> x * x).toList();
    System.out.println(square); // will print [1,4,9,16,25]

    // map with arrays
    int[] temp = Arrays.stream(arr).map(x-> x*x).toArray();
    for(int i=0; i<arr.length; i++)
        System.out.print(temp[i] + " ");  // will print 1 4 9 16 25

    // filter with collection object like list
    List<Integer> even = number.filter(x -> isEven(x)).toList();
    System.out.println(even); // will print [2,4]

    //foreach with collection object like list
    number.forEach(x -> System.out.print(x*x*x + " "));  // will print [1, 8, 27, 64, 125]

    //reduce with array
    System.out.println();
    int sum = Arrays.stream(arr).reduce(0,(acc,i) -> acc+i);
    System.out.println(sum);  // will print 15
```

37. Collections (and what all collection we used in project, how and where)

38. Reading a complete folder by streams

```java
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.stream.Stream;

public class SimpleTesting {

    public static void main(String[] args) throws IOException {
        try (Stream<Path> paths = Files.walk(Paths.get("/home/folder/src"))) {
            paths
            .filter(Files::isRegularFile)
            .forEach(System.out::println);
        }
    }
}
```

39. Equals and hashcode methods –

- Java.lang.object has two very important methods defined:
  public Boolean equals(Object obj) and public int hashCode().

- **equals() ->** By default, two objects are equal if and only if they are referring to the same memory location. Most Java classes override this method to provide their own comparison logic.

- **Hashcode() ->** Java Object hashCode() is a native method and returns the integer hash code value of the object.

- If **two objects are equal** according to the **equals()** method, then calling the **hashCode()** on each of the two objects must produce the **same** integer result.

- However, if **two objects are unequal** according to the **equals**(), then calling the hashCode() on each of both objects **may produce same or distinct integer results.**

- Overriding the hashCode() is generally necessary whenever equals() is overridden, to maintain the general contract for the hashCode() method, which states that equal objects must have equal hash codes.

40. How to get div in JS?

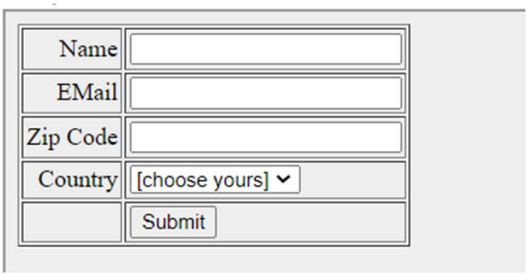
```html
<body>

    <h1>The div element</h1>

    <div id="div1" class="myDiv">
      <p>This is some text in a div element.</p>
    </div>


    <div id="div1" class="myDiv">
      <p>This is some text in a div element.</p>
    </div>

</body>
```

HTML Code containing div

```javascript
const myDiv = document.getElementById("div1")

// or we can use other methods like
const myDiv = document.getElementsByClassName("div1")[0]
const myDiv = document.querySelector("#div1")
```

Getting div through JS

41. Form validation



Basic Form

```
<script type = "text/javascript">
   <!--
      // Form validation code will come here.
      function validate() {

         if( document.myForm.Name.value == "" ) {
            alert( "Please provide your name!" );
            document.myForm.Name.focus() ;
            return false;
         }
         if( document.myForm.EMail.value == "" ) {
            alert( "Please provide your Email!" );
            document.myForm.EMail.focus() ;
            return false;
         }
         if( document.myForm.Zip.value == "" || isNaN( document.myForm.Zip.value ) ||
            document.myForm.Zip.value.length != 5 ) {

            alert( "Please provide a zip in the format #####." );
            document.myForm.Zip.focus() ;
            return false;
         }
         if( document.myForm.Country.value == "-1" ) {
            alert( "Please provide your country!" );
            return false;
         }
         return( true );
      }
   //-->
</script>
```

Form Validation using JS

42. SQL queries(to get top 3) – Both queries will return same result.

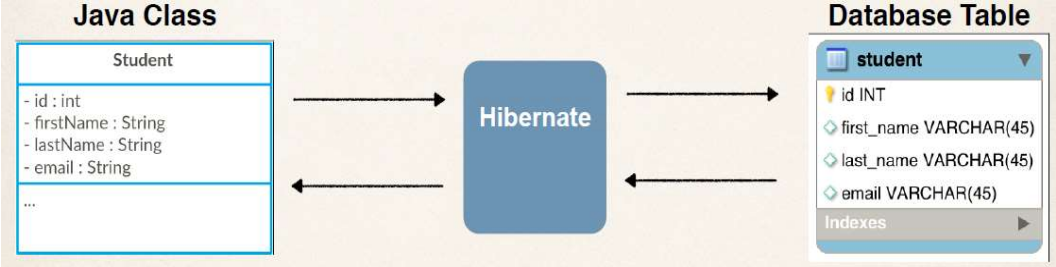
```
SELECT TOP 3 * FROM Customers;
```

```
SELECT * FROM Customers
LIMIT 3;
```

43. Spring core - Core (spring-core) is the core of the framework that power features such as Inversion of Control and dependency injection.

44. Spring JDBC - It provides a JDBC abstraction layer that eliminates the need to separate JDBC coding when dealing with databases.

45. JPA and hibernate –



JPA -

- The Java Persistence API (JPA) is a specification of Java. It is used to persist data between Java object and relational database. JPA acts as a bridge between object-oriented domain models and relational database systems.
- JPA is only a specification, it is not an implementation.

Hibernate –

- It is used to save the Java objects in the relational database system.
- The main feature of Hibernate is to map the Java classes to database tables.
- It helps in mapping Java data types to SQL data types.

The major difference between Hibernate and JPA is that Hibernate is a framework while JPA is API specifications. Hibernate is the implementation of all the JPA guidelines.

46. AWS –

- AWS (Amazon Web Services) is a comprehensive, evolving cloud computing platform provided by Amazon that includes a mixture of infrastructure as a service (IaaS), platform as a service (PaaS) and packaged software as a service (SaaS) offering. AWS services can offer an organization tool such as compute power, database storage and content delivery services.

- Famous Services of AWS :-
    - **EC2(Elastic Compute Cloud)** : It offers various instance types to developers so that they can choose required resources such as CPU, memory, storage, and networking capacity based on their application requirements.
    - **AWS Lambda : I**t is a serverless compute service. It is also responsible for executing code for applications. It helps you execute a program without the hassle of managing servers.
    - **Amazon (Simple Storage Service) S3 :** It is an open cloud-based storage service that is utilized for online data backup.

47.

i. Review about your project
ii. Challenges faced
iii. What have you done to overcome that

48. Difference b/w abstract class and interface and why we use interface when we have abstract class - even after explaining he said learn about diamond problem

Interfaces are more flexible because a class can implement multiple interfaces. Since Java does not have multiple inheritance, using abstract classes prevents your users from using any other class hierarchy. In short, interfaces are used to implement multiple inheritance.

```java
// Class 1
class GrandParent {

void fun() {
    System.out.println("Grandparent");
}
}

// Class 2
class Parent1 extends GrandParent {
void fun() {
    System.out.println("Parent1");
}
}

// Class 3
class Parent2 extends GrandParent {
void fun() {
    System.out.println("Parent2");
}
}

// Class 4
class Child extends Parent1, Parent2 {

// Main driver method
public static void main(String args[]) {

    // Creating object of this class in main() method
    Child c = new Child();

    // Now calling fun() method from its parent classes
    // which will throw compilation error because it is not clear which fun()
    // method should run
    t.fun();
}
}
```

Diamond Problem

49. How do you achieve multi-threading. Ways to do that. How would you write that in code?

Already answered here.

50. Features of Spring framework

These Spring Framework features are as follow

i. Lightweight

The Spring Framework is very lightweight with respect to its size and functionality. It is due to its POJO implementation which doesn't force to inherit any class or implement any interfaces.

ii. Aspect Oriented Programming(AOP)

It is an important part of Spring Framework. Aspect Oriented Programming is used for separating cross-cutting concerns (for example logging, security etc.) from the business logic of the application.

iii. Transaction Management

Transaction Management use for unify several transaction management APIs and is used to coordinate transactions for Java object. Also, not tie to the J2EE environment and use with containerless environments.

iv. Container

The Spring Framework designs and manages the lifecycle and configurations of application objects.

v. Dependency Injection

Dependency Injection is a feature of Spring Framework allows you to develop loosely coupled applications. Therefore, the unit testing of these loosely coupled applications becomes easier. This also allows the developer to swap out some of the modules according to its need.

vi. Integration with other frameworks

A great thing about this framework is that it doesn't try to solve the problems have already solved. It just tries to integrate them with its framework which provides a solution to greater problems. Example IBATIS, Hibernate, Toplink etc.

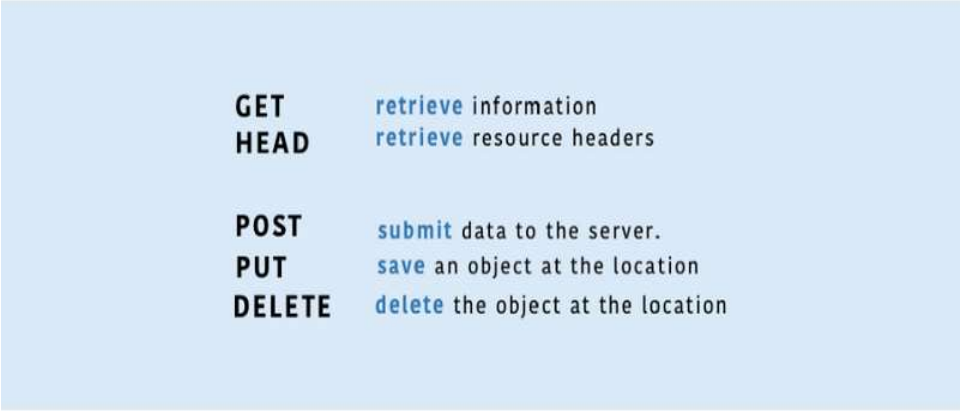
51. Autowired –

- The @Autowired annotation in spring automatically injects the dependent beans into the associated references of a POJO class. This annotation will inject the dependent beans by matching the datatype**.**

- Autowiring makes the container to search the bean configurations and do the collaboration among beans, without the developer specifically mentioning these.

52. How to use REST

REST determines how the API looks like. It stands for "Representational State Transfer". It is a set of rules that developers follow when they create their API.



53. How to achieve MVC

- **Model:** It represents the business layer of application. It is an object to carry the data that can also contain the logic to update controller if data is changed.
- **View:** It represents the presentation layer of application. It is used to visualize the data that the model contains.
- **Controller:** It works on both the model and view. It is used to manage the flow of application, i.e. data flow in the model object and to update the view whenever data is changed.

Example – For Employee Management System :

> Implementation of MVC using Java
>
> To implement MVC pattern in Java, we are required to create the following three classes.
>
> - **Employee Class**, will act as model layer
> - **EmployeeView Class**, will act as a view layer
> - **EmployeeContoller Class**, will act a controller layer

For more information Read here - [MVC Architecture in Java]

54. How to integrate and use Hibernate
https://www.baeldung.com/spring-boot-hibernate

55. What is repository ?

- Repositories are classes or components that encapsulate the logic required to access data sources. They centralize common data access functionality,

providing better maintainability and decoupling the infrastructure or technology used to access databases from the domain model layer.

```java
// Java Program to illustrate StudentRepository File

package com.example.demo.repository;

import com.example.demo.entity.Student;
import org.springframework.stereotype.Repository;

import java.util.HashMap;
import java.util.Map;

@Repository
public class StudentRepository implements DemoRepository<Student> {

    // Using an in-memory Map
    // to store the object data
    private Map<Long, Student> repository;

    public StudentRepository() {
        this.repository = new HashMap<>();
    }

    // Implementation for save method
    @Override
    public void save(Student student) {
        repository.put(student.getId(), student);
    }

    // Implementation for findStudentById method
    @Override
    public Student findStudentById(Long id) {
        return repository.get(id);
    }
}
```

In this **StudentRepository.java** file, you can notice that we have added the @Repository annotation to indicate that the class provides the mechanism for storage, retrieval, update, delete and search operation on objects.

56. Explain about Spring Cloud?

- Spring Cloud is a framework for building robust cloud applications. The framework facilitates the development of applications by providing solutions to many of the common problems faced when moving to a distributed environment.
- The major use-case for Spring Cloud is the ready-to-use solution that it provides to common problems observed in distributed environments like load balancing, service discovery, circuit breaking, etc., which can easily be integrated in an existing Spring project.

57. How to store a data in client machine ?

- All your web storage data is contained within two object-like structures inside the browser: **sessionStorage** and **localStorage**. The first one persists data for as long as the browser is open (the data is lost when the browser is closed) and the second one persists data even after the browser is closed and then opened again.
- **localstorage.setItem()** method allows you to save a data item in storage — it takes two parameters: the name of the item, and its value.

```
localStorage.setItem('name','Chris');
```

- The **localstorage.getItem()** method takes one parameter — the name of a data item you want to retrieve — and returns the item's value.

```
let myName = localStorage.getItem('name');
```

58. If I want to declare a div and want it to be at the top of the website how to achieve that?

```
div {
  position: absolute;
  top: 0;
}
```

59. How to hold the divs vertically in the center of the website?

```
<!DOCTYPE html>
<html>
<head>
<style>
.container {

  border: 3px solid green;
}

.vertical-center {
border:2px solid red;
 width:80%;
 margin:auto;
}

</style>
</head>
<body>

<div class="container">
  <div class="vertical-center">
    <p>I am vertically centered.</p>
  </div>
  <div class="vertical-center">
    <p>I am vertically centered.</p>
  </div>
  <div class="vertical-center">
    <p>I am vertically centered.</p>
  </div>
</div>

</body>
</html>
```

60. How to merge two objects in JavaScript?

Spread operator (...) can be used to merge two or more objects and create a new one that has properties of the merged objects.

```
let obj1 = {"name1":"Voldemort", "age1":21}
let obj2 = {"name2": "Lois Lane", "age2":19}

let obj3 = {...obj1, ...obj2} // merging two objects

console.log(obj3);
```

Output->

```
{ name1: 'Voldemort', age1: 21, name2: 'Lois Lane', age2: 19 }
```

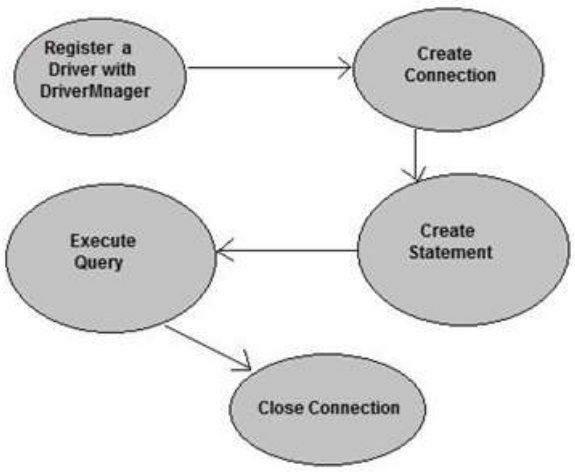
61. Display the student details whose count of marks is over 1000

62. Triggers – A trigger is a stored procedure in database which automatically invokes whenever a special event in the database occurs. For example, a trigger can be invoked when a row is inserted into a specified table or when certain table columns are being updated.

63. How to connect to database

```java
public static Connection getConnection(String[] args) {

        Connection conn1 = null;

        try {
            String url1 = "jdbc:mysql://localhost:3306/test1";
            String user = "root";
            String password = "secret";

            conn1 = DriverManager.getConnection(url1, user, password);
            if (conn1 != null) {
                System.out.println("Connected to the database test1");
            }
            return conn1;
        }

        catch (SQLException ex) {
            System.out.println("An error occurred. Maybe user/password is invalid");
            ex.printStackTrace();
        }
}
```

64. Steps involved in jdbc



65. How to load the driver?

To connect to a database, we must get an instance of a JDBC driver. We can obtain it through the DriverManager by specifying the JDBC URL connection string.

```
Class.forName("oracle.jdbc.driver.OracleDriver");
```

66. What is type of Connection class?

- A Connection is a session between a Java application and a database. It helps to establish a connection with the database.
- Connection interface resides in java.sql package.

67. How to do testing?

Using Junit.

68. How to validate a function that shows a max of an Array?

69. How to get keys from a map?

```
12      public static void main(String[] args) {
13
14          HashMap<Integer, Integer> map = new HashMap<>();
15          map.put(1,2);
16          map.put(2,3);
17          map.put(3,4);
18
19          // Getting keys - way1
20          for(Map.Entry<Integer, Integer> m : map.entrySet()){
21              System.out.println(m.getKey());
22          }
23
24          // Getting keys - way2
25          for ( Integer key : map.keySet() ) {
26              System.out.println( key );
27          }
28
29      }
30  }
31
```