

1. What will happen when we try to add duplicate key in HashMap?

Ans: -

public V put (K key, value)

It stores the data in (Key, Value) pairs, and you can access them by an index of another type (e.g., an Integer). One object is used as a key (index) to another object (value). If you try to insert the duplicate key, it will replace the element of the corresponding key.

It means if I have key values as put (1,5) then if key 1 is duplicate key then my value will be replaced by value 5 as a key.

2. Does Map implement Collection interface?

Ans:- The answer is no.

Collection interface provides add, remove, search or iterate while Map has clear, get, put, remove, etc.

3. Difference between SOAP and REST

What is SOAP?

SOAP is a protocol which was designed before REST and came into the picture. The main idea behind designing SOAP was to ensure that programs built on different platforms and programming languages could exchange data in an easy manner. SOAP stands for Simple Object Access Protocol.

What is REST?

REST was designed specifically for working with components such as media components, files, or even objects on a particular hardware device. Any web service that is defined on the principles of REST can be called a Restful web service. A Restful service would use the normal HTTP verbs of GET, POST, PUT and DELETE for working with the required components. REST stands for Representational State Transfer.

Below is the main difference between SOAP and REST API

SOAP	REST
<ul style="list-style-type: none">• SOAP stands for Simple Object Access Protocol	<ul style="list-style-type: none">• REST stands for Representational State Transfer
<ul style="list-style-type: none">• SOAP is a protocol. SOAP was designed with a specification. It includes a WSDL file which has the required information on what the	<ul style="list-style-type: none">• REST is an Architectural style in which a web service can only be treated as a RESTful service if it follows the constraints of being<ol style="list-style-type: none">1. Client Server2. Stateless3. Cacheable

web service does in addition to the location of the web service.	4. Layered System 5. Uniform Interface
<ul style="list-style-type: none"> SOAP cannot make use of REST since SOAP is a protocol and REST is an architectural pattern. 	<ul style="list-style-type: none"> REST can make use of SOAP as the underlying protocol for web services, because in the end it is just an architectural pattern.
<ul style="list-style-type: none"> SOAP uses service interfaces to expose its functionality to client applications. In SOAP, the WSDL file provides the client with the necessary information which can be used to understand what services the web service can offer. 	<ul style="list-style-type: none"> REST use Uniform Service locators to access to the components on the hardware device. For example, if there is an object which represents the data of an employee hosted on a URL as http://demo.guru99 , the below are some of URI that can exist to access them.
<ul style="list-style-type: none"> SOAP requires more bandwidth for its usage. Since SOAP Messages contain a lot of information inside of it, the amount of data transfer using SOAP is generally a lot. <pre><?xml version="1.0"?> <SOAP-ENV:Envelope xmlns:SOAP-ENV ="http://www.w3.org/2001/12/soap-envelope" SOAP-ENV:encodingStyle =" http://www.w3.org/2001/12/soap-encoding"> <soap:Body> <Demo.guru99WebService xmlns="http://tempuri.org/"> <EmployeeID>int</EmployeeID> </Demo.guru99WebService> </soap:Body> </SOAP-ENV:Envelope></pre>	<ul style="list-style-type: none"> REST does not need much bandwidth when requests are sent to the server. REST messages mostly just consist of JSON messages. Below is an example of a JSON message passed to a web server. You can see that the size of the message is comparatively smaller to SOAP. <pre>{"city":"Mumbai","state":"Maharastra"}</pre>
<ul style="list-style-type: none"> SOAP can only work with XML format. As seen from SOAP messages, all data passed is in XML format. 	<ul style="list-style-type: none"> REST permits different data format such as Plain text, HTML, XML, JSON, etc. But the most preferred format for transferring data is JSON.

How you handle exception in spring boot project?

How you deploy your application on server?

Answers for above question will be merged after done with project understating.

5) How/when can we rollback our transaction?

You can use ROLLBACK TRANSACTION to erase all data modifications made from the start of the transaction or to a savepoint. It also frees resources held by the transaction. This does not include changes made to local variables or table variables. These are not erased by this statement.

You just have to write the statement ROLLBACK TRANSACTION, followed by the name of the transaction that you want to rollback. Now, try to run the AddBook transaction to insert the record where the name is Book15 (make sure that no book with this name already exists in the Books table).

6) How we handle merging conflict?

<https://www.simplilearn.com/tutorials/git-tutorial/merge-conflicts-in-git>

7) Have you written any test cases?

Yes,

How to write test cases for software:

Use a Strong Title

A good test case starts with a strong title. As a best practice, it's good to name the test case along the same lines as the module that you are testing. For example, if you're testing the login page, include "Login Page" in the title of the test case. In some cases, if the tool you're using doesn't already do this, it might make sense to include a unique identifier in the title of the test case as well, so the identifier can be referenced instead of a long title

Include a Strong Description

The description should tell the tester what they're going to test. Sometimes this section might also include other pertinent information such as the test environment, test data, and preconditions/assumptions. A description should be easy to read and immediately communicate the high-level goal of the test.

Include Assumptions and Preconditions

You should include any assumptions that apply to the test and any preconditions that must be met prior to the test being executed. This information can include which page the user should start the test on, dependencies on the test environment, and any special setup requirements that must be done before running the test. This information also helps keep the test steps short and concise.

Keep the Test Steps Clear and Concise

Test cases should be simple. Keep in mind, the person who wrote the test case might not be the same person who executes the test itself. The test steps should include the necessary data and information on how to execute the test. This is perhaps the most important part of a test case. Keep this section clear and concise, but don't leave out any necessary details. Write the test case so that anyone can go in and perform the test.

Include the Expected result

The expected result tells the tester what they should experience as a result of the test steps. This is how the tester determines if the test case is a "pass" or "fail".

Make it Reusable

A good test case is reusable and provides long-term value to the software testing team. When writing a test case, keep this in mind. You can save time down the road by re-using the test case instead of re-writing it.

Sample of a Test Case

Here is an example of a test case:

Title: Login Page – Authenticate Successfully on gmail.com

Description: A registered user should be able to successfully login at gmail.com.

Precondition: the user must already be registered with an email address and password.

Assumption: a supported browser is being used.

Test Steps:

1. Navigate to gmail.com
2. In the 'email' field, enter the email address of the registered user.
3. Click the 'Next' button.
4. Enter the password of the registered user
5. Click 'Sign In'

Expected Result: A page displaying the Gmail user's inbox should load, showing any new message at the top of the page.

8) Can we write test case for static method?

Static methods don't need an instance of the class to be invoked — they can be called on the class itself. Although testing a non-static method (at least one that doesn't call a static method or interact with external dependencies) is straightforward, testing a static method is not an easy task at all.

Answer is too long in Command prompt mode.

For detailed explanation click on link below.

<https://blog.codecentric.de/en/2011/11/testing-and-mocking-of-static-methods-in-java/>

9) Do you have any front end exp?

Ans- No,

10.What is join?

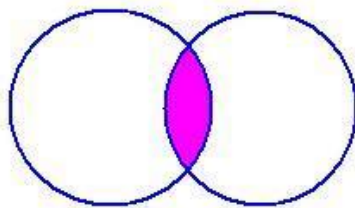
We have three referral answers here one is from sql topic other from thread and from string topic.

From Sql join>>>A SQL Join statement is used to combine data or rows from two or more tables based on a common field between them. Different types of Joins are:

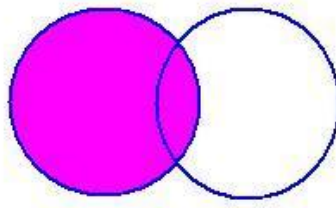
- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- FULL JOIN

Consider following are two different table we have

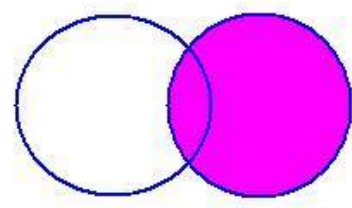
JOINS AND SET OPERATIONS IN RELATIONAL DATABASES



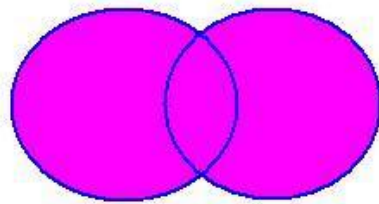
Inner join (result similar to Intersect)



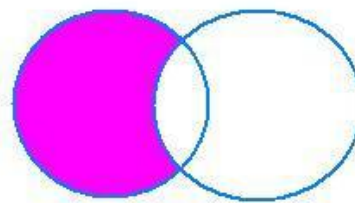
Left outer join



Right outer join



Full outer join



Minus

#Join from thread topic >>>The `join()` method waits for a thread to die. In other words, it causes the currently running threads to stop executing until the thread it joins with completes its task.

#Join from string topic >>>The `java.lang.string.join()` method concatenates the given elements with the delimiter and returns the concatenated string. Note that if an element is null, then null is added. The `join()` method is included in java string since JDK 1.8. There are two types of `join()` methods in java string.

Syntax:

```
public static String join(CharSequence deli, CharSequence... ele)
```

and

```
public static String join  
(CharSequence deli, Iterable<? extends CharSequence> ele)
```

Parameters:

deli- delimiter to be attached with each element

ele- string or char to be attached with delimiter

Returns : string joined with delimiter.

```
// Java program to demonstrate
```

```
// working of join() method
```

```
class Gfg1 {
```

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

```

{

    // delimiter is "<" and elements are "Four", "Five", "Six", "Seven"

    String gfg1 = String.join(" < ", "Four", "Five", "Six", "Seven");

    System.out.println(gfg1);

}

}

```

Output:

Four < Five < Six < Seven

11.What is polymorphism?

Ans>>**Real life example of polymorphism:** A person at the same time can have different characteristic. Like a man at the same time is a father, a husband, an employee. So the same person posses different behavior in different situations. This is called polymorphism. Polymorphism is considered one of the important features of Object-Oriented Programming. Polymorphism allows us to perform a single action in different ways. In other words, polymorphism allows you to define one interface and have multiple implementations. The word “poly” means many and “morphs” means forms, So it means many forms.

In Java polymorphism is mainly divided into two types:

- Compile time Polymorphism
- Runtime Polymorphism

1.Compile-time polymorphism: It is also known as static polymorphism. This type of polymorphism is achieved by function overloading or operator overloading. But **Java doesn't support the Operator Overloading.**

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

2. Runtime polymorphism: It is also known as Dynamic Method Dispatch. It is a process in which a function call to the overridden method is resolved at Runtime. This type of polymorphism is achieved by Method Overriding.

Method overriding, on the other hand, occurs when a derived class has a definition for one of the member functions of the base class. That base function is said to be **overridden**.

12..How will you fetch 1 lakh employee data from database in sorted order?

The first way is: -

```

SELECT *
FROM Employee

```

Where SIZE > 100000
ORDER BY NAME ASC

But it will take more time for loading all 10000 records from the databas
The efficient way is fetching the data while it is continuously loading on our result page

Note- set fetch size as 500;

```
SELECT *  
FROM (SELECT ename, empid,salary,address  
      FROM Employee  
      WHERE SIZE > 100000 ORDER BY NAME DESC)  
WHERE ROWNUM <= 100
```

13.Diff bet string buffer n builder?

1)	StringBuffer is <i>synchronized</i> i.e. thread safe. It means two threads can't call the methods of StringBuffer simultaneously.	StringBuilder is <i>non-synchronized</i> i.e. not thread safe. It means two threads can call the methods of StringBuilder simultaneously.
2)	StringBuffer is <i>less efficient</i> than StringBuilder	StringBuilder is <i>more efficient</i> than StringBuffer

```
StringBuffer buffer=new StringBuffer("hello");  
"hello");
```

```
buffer.append("java");
```

```
StringBuilder builder=new StringBuilder(
```

```
builder.append("java");
```

14.Difference between HashMap & Hashtable

HashMap	Hashtable
1) HashMap is non synchronized . It is not-thread safe and can't be shared between many threads without proper synchronization code.	Hashtable is synchronized . It is thread-safe and can be shared with many threads.
2) HashMap allows one null key and multiple null values .	Hashtable doesn't allow any null key or value
3) HashMap is a new class introduced in JDK 1.2 .	Hashtable is a legacy class .
4) HashMap is fast .	Hashtable is slow .
5) We can make the HashMap as synchronized by calling this code Map m = Collections.synchronizedMap(hashMap);	Hashtable is internally synchronized and can't be unsynchronized.
6) HashMap is traversed by Iterator .	Hashtable is traversed by Enumerator and Iterator .

7) Iterator in HashMap is fail-fast .	Enumerator in Hashtable is not fail-fast .
8) HashMap inherits AbstractMap class.	Hashtable inherits Dictionary class.

15.What is string constant pool?

String pool is nothing but a storage area in **Java heap** where string literals stores. It is also known as **String Intern Pool** or **String Constant Pool**. It is just like object allocation. By default, it is empty and privately maintained by the **Java String** class. Whenever we create a string the string object occupies some space in the heap memory. Creating a number of strings may increase the cost and memory too which may reduce the performance also.

16.What is immutable?

The term Mutable means "can change" and Immutable means "cannot change". An Immutable Object means **that the state of the Object cannot change after its creation**. ... Once string object is created its data or state can't be changed but a new string object is created. string is called as immutable that holds same properties. We can also create a class as immutable.

17.When we use string buffer n builder?

If the Object value can change and will only be accessed from a single thread, use a **StringBuilder** because **StringBuilder** is unsynchronized. In case the Object value can change, and will be modified by multiple threads, use a **StringBuffer** because **StringBuffer** is synchronized.

18.Diff bet iterator n list iterator?

Iterator	ListIterator
Can traverse elements present in Collection only in the forward direction.	Can traverse elements present in Collection both in forward and backward directions.
Helps to traverse Map, List and Set.	Can only traverse List and not the other two.
Indexes cannot be obtained by using Iterator.	It has methods like <code>nextIndex()</code> and <code>previousIndex()</code> to obtain indexes of elements at any time while traversing List.
Cannot modify or replace elements present in Collection	We can modify or replace elements with the help of <code>set(E e)</code>

Iterator	ListIterator
Cannot add elements and it throws <code>ConcurrentModificationException</code> .	Can easily add elements to a collection at any time.
Certain methods of Iterator are <code>next()</code> , <code>remove()</code> and <code>hasNext()</code> .	Certain methods of ListIterator are <code>next()</code> , <code>previous()</code> , <code>hasNext()</code> , <code>hasPrevious()</code> , <code>add(E e)</code> .

19.What is interface n abstract class?

An **abstract class** allows you to create functionality that subclasses can implement or override. An interface only allows you to define functionality, not implement it. And whereas a class can extend only one abstract class, it can take advantage of multiple interfaces.

20.What is check exception?

1) Checked Exception: are the exceptions that are checked at compile time. If some code within a method throws a checked exception, then the method must either handle the exception or it must specify the exception using *throws* keyword.

For example, consider the following Java program that opens file at location “C:\test\a.txt” and prints the first three lines of it. The program doesn’t compile, because the function `main()` uses `FileReader()` and `FileReader()` throws a checked exception *FileNotFoundException*. It also uses `readLine()` and `close()` methods, and these methods also throw checked exception *IOException*

```
import java.io.*;

class Main {

    public static void main(String[] args) {

        FileReader file = new FileReader("C:\\test\\a.txt");

        BufferedReader fileInput = new BufferedReader(file);

        // Print first 3 lines of file "C:\test\a.txt"
```

```

        for (int counter = 0; counter < 3; counter++)

            System.out.println(fileInput.readLine());

        fileInput.close();

    }

}

```

Output:

Exception in thread "main" java.lang.RuntimeException: Uncompilable source code - unreported exception java.io.FileNotFoundException; must be caught or declared to be thrown

at Main.main(Main.java:5)

21.How to provide security to Spring boot project?

Ans not found..

22.write sql query to to fetch all data from specific column?

Select coloum_name from table_name;

e.g.-- SELECT city from employee;

tis query will give all data from specific coloum.

23.If given no is Even divide by 2, If odd multiply by 3 & add 1 and given number is from (10-1000).

```
package com.creative;
```

```
import java.util.Scanner;
```

```
//If given no is Even divide by 2, If add multiply by 3 & add 1 and given number is from (10-1000).
```

```
public class Oddeven {
```

```
    public void EvenOdFun(int a) {
```

```
        if (a < 10 || a > 1000) {
            System.out.println("please enter valid input");
```

```
        } else if (a % 2 == 0) {
            System.out.println("this is even number");
```

```
            a = a / 2;
            System.out.println("the value after performing given logic>>" + a);
```

```
        } else {
```

```
        System.out.println(" this is odd number");
        a = a * 3;
        a = a + 1;
        System.out.println("result after odd logic>>>" + a);
    }
}

public static void main(String[] args) {

    Scanner sc = new Scanner(System.in);
    System.out.println("please enter the no");
    int a = sc.nextInt();
    Oddeven ov = new Oddeven();
    ov.EvenOdFun(a);    }}
```

24) What is spring actuator ?

Spring Actuator is a cool feature of Spring Boot with the help of which you can see what is happening inside a running application. So, whenever you want to debug your application, and need to analyze the logs you need to understand what is happening in the application right? In such a scenario, the Spring Actuator provides easy access to features such as identifying beans, CPU usage, etc. The Spring Actuator provides a very easy way to access the production-ready REST points and fetch all kinds of information from the web. These points are secured using Spring Security's content negotiation strategy.

25) What is java ?

Java is a programming language and a platform. Java is a high level, robust, object-oriented and secure programming language. Java was developed by *Sun Microsystems* in the year 1995.

There are many devices where Java is currently used. Some of them are as follows:

1. Desktop Applications such as acrobat reader, media player, antivirus, etc.
2. Web Applications such as irctc.co.in, javatpoint.com, etc.
3. Enterprise Applications such as banking applications.
4. Mobile
5. Embedded System
6. Smart Card
7. Robotics
8. Games, etc.

26)What is Decode and Case in oracle ?

Could not find answer of that question

27)What is connection ?

A Connection is the session between java application and database. The Connection interface is a factory of Statement, PreparedStatement, and DatabaseMetaData i.e. object of Connection can be used to get the object of Statement and DatabaseMetaData. The Connection interface provide many methods for transaction management like commit(), rollback() etc.

28)what annotations you have used?

@Required

This annotation is applied on bean setter methods. Consider a scenario where you need to enforce a required property. The @Required annotation indicates that the affected bean must be populated at configuration time with the required property. Otherwise an exception of type BeanInitializationException is thrown.

@Autowired

This annotation is applied on fields, setter methods, and constructors. The @Autowired annotation injects object dependency implicitly.

When you use @Autowired on fields and pass the values for the fields using the property name, Spring will @Qualifier

This annotation is used along with @Autowired annotation. When you need more control of the dependency injection process, @Qualifier can be used. @Qualifier can be specified on individual constructor arguments or method parameters. This annotation is used to avoid confusion which occurs when you create more than one bean of the same type and want to wire only one of them with a property. automatically assign the fields with the passed values.

@Configuration

This annotation is used on classes which define beans. @Configuration is an analog for XML configuration file – it is configuration using Java class. Java class annotated with @Configuration is a configuration by itself and will have methods to instantiate and configure the dependencies.

@Bean

This annotation is used at the method level. @Bean annotation works with @Configuration to create Spring beans. As mentioned earlier, @Configuration will have methods to instantiate and configure dependencies. Such methods will be annotated with @Bean. The method annotated with this annotation works as bean ID and it creates and returns the actual bean.

@Controller

The @Controller annotation is used to indicate the class is a Spring controller. This annotation can be used to identify controllers for Spring MVC or Spring WebFlux.

29) What is driver manager ?

The DriverManager class acts as an interface between the user and drivers. It keeps track of the drivers that are available and handles establishing a connection between a database and the appropriate driver. It contains several methods to keep the interaction between the user and drivers.

In this various method are as follows:

Public static Connection getConnection(String url): is used to establish the connection with the specified url

```
public static Connection getConnection(String  
userName,String password)
```

30) What is public static void main ?

main() in Java is the entry point for any Java program. It is always written as **public static void main(String[] args).**

- **public:** Public is an access modifier, which is used to specify who can access this method. Public means that this Method will be accessible by any Class.
- **static:** It is a keyword in java which identifies it is class-based. main() is made static in Java so that it can be accessed without creating the instance of a Class. In case, main is not made static then the compiler

will throw an error as **main()** is called by the JVM before any objects are made and only static methods can be directly invoked via the class.

- **void:** It is the return type of the method. Void defines the method which will not return any value.
- **main:** It is the name of the method which is searched by JVM as a starting point for an application with a particular signature only. It is the method where the main execution occurs.
- **String args[]:** It is the parameter passed to the main method.

31) What is application server?

Application server contains Web and EJB containers. It can be used for servlet, jsp, struts, jsf, ejb etc. It is a component based product that lies in the middle-tier of a server centric architecture.

It provides the middleware services for state maintenance and security, along with persistence and data access. It is a type of server designed to install, operate and host associated services and applications for the IT services, end users and organizations.

The Example of Application Servers are:

1. **JBoss:** Open-source server from JBoss community.
2. **Glassfish:** Provided by Sun Microsystem. Now acquired by Oracle.
3. **Weblogic:** Provided by Oracle. It more secured.
4. **Websphere:** Provided by IBM.

32) What is diff bet c, c++ and java ?

C	C++	JAVA
It is procedural language	It is an object-oriented programming language.	It is a pure object-oriented programming language
It uses the top-down approach.	It uses the bottom-up approach.	It also uses the bottom-up approach.
It is a static programming language.	It is also a static programming language.	It is a dynamic programming language.
It uses a compiler only to translate the code into	It also uses a compiler only to translate the code	java uses both compiler and interpreter and it is

machine language.	into machine language.	also known as an interpreted language
There are 32 keywords in the C language.	There are 60 keywords in the C++ language.	There are 52 keywords in the Java language.
It supports pointer	It also supports pointer.	Java does not support the pointer concept because of security.
It is platform dependent.	It is platform dependent.	it is platform-independent because of byte code.

33) What is Schema ?

A Schema in [SQL](#) is a collection of database objects associated with a [database](#). The username of a database is called a Schema owner (owner of logically grouped structures of data). Schema always belong to a single database whereas a database can have single or multiple schemas. Also, it is also very similar to separate namespaces or containers, which stores database objects. It includes various database objects including your tables, views, procedures, index, etc.

Advantages of Schema

- ❑ You can apply security permissions for separating and protecting database objects based on user access rights.
- ❑ A logical group of database objects can be managed within a database. Schemas play an important role in allowing the database objects to be organized into these logical groups.
- ❑ The schema also helps in situations where the database object name is the same. But these objects fall under different logical groups.
- ❑ A single schema can be used in multiple databases.
- ❑ The schema also helps in adding security.
- ❑ It helps in manipulating and accessing the objects which otherwise is a complex method.
- ❑ You can also transfer the ownership of several schemas.
- ❑ The objects created in the database can be moved among schemas.

34) How you change your application schemas ?

Related to project

35) Jsp servlet life cycle ?

A Java Server Page life cycle is defined as the process that started with its creation which later translated to a servlet and afterward servlet lifecycle comes into play. This is how the process goes on until its destruction.

o Following steps are involved in the JSP life cycle:

- ☐ Translation of JSP page to Servlet
- ☐ Compilation of JSP page(Compilation of JSP into test.java)
- ☐ Classloading (test.java to test.class)
- ☐ Instantiation(Object of the generated Servlet is created)
- ☐ Initialization(jspInit() method is invoked by the container)
- ☐ Request processing(_jspService() is invoked by the container)
- ☐ JSP Cleanup (jspDestroy() method is invoked by the container)

36) Interface and abstract class diff ?

Abstract class	Interface
Abstract class can have abstract and non-abstract methods.	Interface can have only abstract methods. Since Java 8, it can have default and static methods also.
Abstract class doesn't support multiple inheritance.	Interface supports multiple inheritance.
Abstract class can have final, non-final, static and non-static variables	Interface has only static and final variables.

37) What is multiple inheritance?

One class has many super classes called as multiple inheritance.

Why multiple inheritance not supported in java in case of classes?

Class base has test () method and class derived has also test () method. Class test extends Base, Derived, which test method It will called, so it create the ambiguity so that's why multiple inheritance does not supports in java.

```
package com.multiple.inheritance;
```

```
public class A {  
  
    void m1() {  
  
    }  
}
```

```
package com.multiple.inheritance;
```

```
public class B {  
  
    void m1() {  
  
    }  
}
```

```
package com.multiple.inheritance;
```

```
class C extends A,B {  
  
    public static void main(String[] args) {  
  
        C c= new C();  
        c.m1();  
    }  
}
```

Note- it will get the compile time error.

38) Is there any full form of java ?

There no full form of java

39) what is @qualifier, @Bean ?

40) what is singleton answer factory design pattern?

It means define the class which has single instance that provide the global point of access to it called as singleton design pattern.

When?

If you have business requirement in which only one object is created then you should go for singleton design pattern.

Factory design pattern

Design the factory method in which multiple objects are stored called as factory design pattern.

Why?

Sometime our application or frameworks don't know what kind of object has to create at run time. It only knows when it has to create.

Another issue is that class will contain object of another classes, this can be easily achieved by just using the new keyword and the class constructor. The problem with this approach is that it is very hard coded approach to create the object as this creates dependency between two classes.

To overcome this above situation, we should go for factory pattern.

Example:-

Suppose we have requirement to book AC ticket for First tier, second tier and Third tier.

Step 1 create the interface Booking.

Step 2- create the concrete class First tier will implement the same interface

Step 3- create the concrete class Second tier will implement the same interface

Step 4- create the concrete class Third tier will implement the same interface

Step 5- create class BookingFactory to generate the object of concrete class.

Step 6- Create the factory class to get the object of concrete class by passing the data

41) Stored Procedure and function difference

Following are the important differences between SQL Function and SQL Procedure.

Sr. No.	Key	Function	Procedure
1	Definition	A function is used to calculate result using given inputs.	A procedure is used to perform certain task in order.
2	Call	A function can be called by a procedure.	A procedure cannot be called by a function.
3	DML	DML statements cannot be executed within a function.	DML statements can be executed within a procedure.
4	SQL, Query	A function can be called within a query.	A procedure cannot be called within a query.
5	SQL, Call	Whenever a function is called, it is first compiled before being called.	A procedure is compiled once and can be called multiple times without being compiled.
6	SQL, Return	A function returns a value and control to calling function or code.	A procedure returns the control but not any value to calling function or code.
7	try-catch	A function has no support for try-catch	A procedure has support for try-catch blocks.
8	SELECT	A select statement can have a	A select statemnt can't have a

		function call.	procedure call.
9	Explicit Transaction Handling	A function can not have explicit transaction handling.	A procedure can use explicit transaction handling.

42) Why use hibernate?

Open Source and Lightweight

Hibernate framework is open source under the LGPL license and lightweight.

Fast Performance

The performance of hibernate framework is fast because cache is internally used in hibernate framework. There are two types of cache in hibernate framework first level cache and second level cache. First level cache is enabled by default.

Database Independent Query

HQL (Hibernate Query Language) is the object-oriented version of SQL. It generates the database independent queries. So you don't need to write database specific queries. Before Hibernate, if database is changed for the project, we need to change the SQL query as well that leads to the maintenance problem.

Automatic Table Creation

Hibernate framework provides the facility to create the tables of the database automatically. So there is no need to create tables in the database manually.

Simplifies Complex Join

Fetching data from multiple tables is easy in hibernate framework.

Provides Query Statistics and Database Status

Hibernate supports Query cache and provides statistics about query and database status.

43) What is the status code?

Status codes are issued by a server in response to a client's request made to the server.

44) What is 409 status code?

The HTTP Status Code 409 means that the client's request could not be processed because of conflict in the current state of the resource.

Examples of conflicts that could result in an HTTP Status Code 409 include an edit conflict on a given resource as a result of multiple simultaneous updates.

45) What is Predicate?

It is a functional interface which represents a predicate (boolean-valued function) of one argument. It is defined in the `java.util.function` package and contains `test()` a functional method.

Methods	Description
<code>boolean test(T t)</code>	It evaluates this predicate on the given argument.
<code>default Predicate<T> and(Predicate<? super T> other)</code>	It returns a composed predicate that represents a short-circuiting logical AND of this predicate and another. When evaluating the composed predicate, if this predicate is false, then the other predicate is not evaluated.
<code>default Predicate<T> negate()</code>	It returns a predicate that represents the logical negation of this predicate.
<code>default Predicate<T> or(Predicate<? super T> other)</code>	It returns a composed predicate that represents a short-circuiting logical OR of this predicate and another. When evaluating the composed predicate, if this predicate is true, then the other predicate is not evaluated.
<code>static <T> Predicate<T></code>	It returns a predicate that tests if two arguments are equal according to <code>Objects.equals(Object, Object)</code> .

isEqual(Object targetRef)	
---------------------------	--

46) Difference between Predicate and Function Interface?

Function interface is used to do the transformation. It can accept one argument and produces a result. On the other side, Predicate can also accept only one argument but it can only return boolean value. It is used to test the condition.

Sr. No.	Key	Function	Predicate
1	Basic	It can take 2 type parameters First one represents input type argument type and second one represents return type.	It can take one type parameter which represents input type or argument type.
2	Return Type	It can return any type of value.	It can only return boolean value
3	Method	It has an abstract method apply().	It has an abstract method test().
4.	Use Case	It can be used to implement conditional checks	It can be used for the transformation and to the return values.

47) Filter() vs. Map() (java8)

- The map returns the transformed object value.
For example, transform the given Strings from Lower case to Upper case by using the map().

```
myList.stream().map(s -> s.toUpperCase()).forEach(System.out::println);
```

- The filter returns the stream of elements that satisfies the predicate.
For example, find the strings which are starting with 'o' by using the filter.

```
myList.stream().filter(s -> s.startsWith("o")).forEach(System.out::println);
```

```
import java.util.ArrayList;
import java.util.List;
```

```
public class MapAndFilterExample {
    static List<String> myList = null;
```

```
    static {
        myList = new ArrayList<String>();
        myList.add("okay");
        myList.add("omg");
        myList.add("kk");
    }
```

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

```
        System.out.println("Converting the given Strings from Lower case to Upper case
        by using map");
```

```
        myList.stream().map(s -> s.toUpperCase()).forEach(System.out::println);
```

```
        System.out.println("filter which is starting with 'o' by using filter");
```

```
        myList.stream().filter(s -> s.startsWith("o")).forEach(System.out::println);
```

```
    }
}
```

48) Difference between Spring and Spring boot ?

Spring	Spring Boot
Spring Framework is a widely used Java EE framework for building applications.	Spring Boot Framework is widely used to develop REST APIs .

It aims to simplify Java EE development that makes developers more productive.	It aims to shorten the code length and provide the easiest way to develop Web Applications .
The primary feature of the Spring Framework is dependency injection .	The primary feature of Spring Boot is Autoconfiguration . It automatically configures the classes based on the requirement.
It helps to make things simpler by allowing us to develop loosely coupled applications.	It helps to create a stand-alone application with less configuration.
The developer writes a lot of code (boilerplate code) to do the minimal task.	It reduces boilerplate code.
To test the Spring project, we need to set up the sever explicitly.	Spring Boot offers embedded server such as Jetty and Tomcat , etc.
It does not provide support for an in-memory database.	It offers several plugins for working with an embedded and in-memory database such as H2 .
Developers manually define dependencies for the Spring project in pom.xml .	Spring Boot comes with the concept of starter in pom.xml file that internally takes care of downloading the dependencies JARs based on Spring Boot Requirement.

49) Difference between Join and UNION ?

JOIN	UNION
Join combines the data into new Columns	Union's combines the data into new Rows.

The Output of JOIN is a new horizontal set of rows having the same number of Rows but can have different numbers of Columns.	The Output of UNION is a new vertical set of rows having the same number of Columns but can have different numbers of Rows.
The JOIN clause in SQL is applicable only when the two Tables involved have at least one Attribute Common in both Tables.	The UNION in SQL is applicable when the two Tables have the same number of Attributes and the domains of corresponding Attributes are also Same.
Types of JOIN are SELF, INNER, LEFT OUTER JOIN, RIGHT OUTER JOIN, FULL OUTER JOIN	Types of UNION are UNION and UNION ALL

Example of Union

```
SELECT 1 AS table1
UNION
SELECT 2 AS table1;
```

```
Output:
table1
-----
1
2
```

Example of Join

```
SELECT * FROM
(SELECT 1 AS table1) AS table1
JOIN
(SELECT 2 AS table2) AS table2
ON (1=1);
```

```
Output:
table1      table2
-----
1          2
```

@SpringBootApplication -is the combination of **@EnableAutoConfiguration**, **@Configuration** and **@ComponentScan** annotations with their default attributes.

- **@EnableAutoConfiguration** - enables auto-configuration. It means that Spring Boot looks for auto-configuration beans on its classpath and automatically applies them
- **@ComponentScan** - enable **@Component** scan on the package where the application is located
- **@Configuration** - allow to register extra beans in the context or import additional configuration classes

51) @Controller and @RestController

- The @Controller is a annotation to mark class as Controller Class in Spring While @RestController is used in REST Web services and similar to @Controller and @ResponseBody.
- The @Controller annotation indicates that the class is controller like web Controller while @RestController annotation indicates that the class is controller where @RequestMapping Method assume @ResponseBody by Default(i.e REST APIs).
- The key difference is that you do not need to use @ResponseBody on each and every handler method once you annotate the class with @RestController.
- @Controller create a Map of Model Object and find a view while @RestController simply return object and object data directly written into http response as JSON orXML.

This can be also done with @Controller annotation and @ResponseBody

annotation but since this is the default behaviour of RESTful Web Services. Spring introduced @RestController which combines the behaviour of @Controller and @ResponseBody together.

52) Find occurrence of word in a string

```
// Java Program to find the occurrence
// of words in a String using HashMap.
import java.io.*;
import java.util.HashMap;
import java.util.Map;

class CheckOccurance {
    public static void main(String[] args)
    {

        // Declaring the String
        String str = "Alice is girl and Bob is boy";
        // Declaring a HashMap of <String, Integer>
        Map<String, Integer> hashMap = new HashMap<>();

        // Splitting the words of string
        // and storing them in the array.
        String[] words = str.split(" ");

        for (String word : words) {

            // Asking whether the HashMap contains the
            // key or not. Will return null if not.
            Integer integer = hashMap.get(word);

            if (integer == null)
                // Storing the word as key and its
                // occurrence as value in the HashMap.
                hashMap.put(word, 1);

            else {
                // Incrementing the value if the word
                // is already present in the HashMap.
                hashMap.put(word, integer + 1);
            }
        }
        System.out.println(hashMap);
    }
}
```

53) Do you know about views in MySQL?

A view is a database object that has no values. Its contents are based on the base table. It contains rows and columns similar to the real table. In MySQL, the View is a virtual table created by a query by joining one or more tables. It is operated similarly to the base table but does not contain any data of its own. The View and table have one main difference that the views are definitions built on top of other tables (or views). If any changes occur in the underlying table, the same changes are reflected in the View also.

Syntax: CREATE [OR REPLACE] VIEW view_name AS SELECT columns FROM tables	Example: CREATE VIEW trainer AS SELECT course_name, trainer FROM courses;
--	---

54) Stored Procedure and triggers

Stored procedures are pieces of the code written in PL/SQL to do some specific task. Stored procedures can be invoked explicitly by the user. It's like a java program , it can take some input as a parameter then can do some processing and can return values.

On the other hand, trigger is a stored procedure that runs automatically when various events happen (eg update, insert, delete). Triggers are more like an event handler they run at the specific event. Trigger can not take input and they can't return values.

Sr. No.	Key	Triggers	Stored procedures
1	Basic	trigger is a stored procedure that runs automatically when various events happen (eg update, insert, delete)	Stored procedures are a pieces of the code in written in PL/SQL to do some specific task

2	Running Methodology	It can execute automatically based on the events	It can be invoked explicitly by the user
3	Parameter	It can not take input as parameter	It can take input as a parameter
4	Transaction statements	we can't use transaction statements inside a trigger	We can use transaction statements like begin transaction, commit transaction, and rollback inside a stored procedure
5	Return	Triggers can not return values	Stored procedures can return values

55) Query to get max salary

`SELECT max(Salary) FROM EmpDetails`

56) Query to get max salary from each department

```
SELECT DeptId, max(Salary) FROM EmpDetails GROUP BY DeptID
```

57) Query to get 2nd highest salary from each department

```
SELECT max(salary)
FROM employees
WHERE salary < (SELECT Max(salary)
                FROM employee);
```

```
SELECT salary FROM employees ORDER BY salary desc LIMIT 2 , 1;
```

58) Query create a table from existing table

By Copying all columns from another table

Syntax:	Example:
<pre>CREATE TABLE new_table AS (SELECT * FROM old_table);</pre>	<pre>CREATE TABLE suppliers AS (SELECT * FROM companies);</pre>

Create a SQL table from another table without copying any values from the old table?

Syntax:	Example:
<pre>CREATE TABLE new_table AS (SELECT * FROM old_table WHERE <false condition>);</pre>	<pre>CREATE TABLE suppliers AS (SELECT * FROM companies WHERE 1 = 2);</pre>

59) What is 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.

As JPA is just a specification, it doesn't perform any operation by itself. It requires an implementation. So, ORM tools like Hibernate, TopLink and iBatis implements JPA specifications for data persistence.

60) Data Structure where insertion order is preserved and duplication is not allowed?

linkedHashSet and LinkedHashMap

61)Java 8 Features

Lambda Expression - only use on Single Abstract Method Interface

Streams - Way of processing collection object in functional manner

Functional interface - predicate, function, consumer, supplier

ForEach loop

Default Method and Static methods in interface

62) Constructor Chaining

Constructor chaining is the process of calling one constructor from another constructor with respect to the current object.

Constructor chaining can be done in two ways:

- Within same class: It can be done using this() keyword for constructors in same class
- From base class: by using super() keyword to call the constructor from the base class.

63) What is Serialization?

Serialization in Java is a mechanism of *writing the state of an object into a byte-stream*. It is mainly used in Hibernate, RMI, JPA, EJB and JMS technologies.

The reverse operation of serialization is called *deserialization* where byte-stream is converted into an object. The serialization and deserialization process is platform-independent, it means you can serialize an object on one platform and deserialize it on a different platform.

64)How to achieve Serialization in java?

To make a Java object serializable we implement the **java.io.Serializable** interface.

The ObjectOutputStream class contains a writeObject() method for serializing an Object.

Example:

Student.java

```
import java.io.Serializable;
```

```
public class Student implements Serializable{
```

```
    int id;
```

```
    String name;
```

```
public Student(int id, String name) {
```



```
        this.id = id;

        this.name = name;
    }
}
```

Test.java

```
import java.io.*;

class Test{

    public static void main(String args[]){

        try{

            //Creating the object
            Student s1 =new Student(211,"ravi");

            //Creating stream and writing the object
            FileOutputStream fout=new FileOutputStream("f.txt");
            ObjectOutputStream out=new ObjectOutputStream(fout);
            out.writeObject(s1);

            //closing the stream
            out.close();

            System.out.println("success");

        }catch(Exception e){

            System.out.println(e);

        }

    }

}
```

65) What are methods in serialization?

There are no methods in serializable interface

To make a Java object serializable we implement the **java.io.Serializable** interface. The ObjectOutputStream class contains a writeObject() method for serializing an Object.

public final void writeObject(Object obj) throws IOException

The `ObjectInputStream` class contains a `readObject()` method for deserializing an object.
`public final Object readObject() throws IOException, ClassNotFoundException`

66) Can we override the static methods?

NO, We cannot override the static methods

But we can give the same method name and method signature.

Then it is known as method hiding.

67) Difference between `StringBuffer` and `StringBuilder`?

Java provides three classes to represent a sequence of characters: `String`, `StringBuffer`, and `StringBuilder`. The `String` class is an immutable class whereas `StringBuffer` and `StringBuilder` classes are mutable. There are many differences between `StringBuffer` and `StringBuilder`. The `StringBuilder` class was introduced in JDK 1.5.

A list of differences between `StringBuffer` and `StringBuilder` are given below:

No.	<code>StringBuffer</code>	<code>StringBuilder</code>
1)	<code>StringBuffer</code> is <i>synchronized</i> i.e. thread safe. It means two threads can't call the methods of <code>StringBuffer</code> simultaneously.	<code>StringBuilder</code> is <i>non-synchronized</i> i.e. not thread safe. It means two threads can call the methods of <code>StringBuilder</code> simultaneously.
2)	<code>StringBuffer</code> is <i>less efficient</i> than <code>StringBuilder</code> .	<code>StringBuilder</code> is <i>more efficient</i> than <code>StringBuffer</code> .

68) Q- what is the difference between `sleep ()` & `wait ()`?

<code>Wait()</code>	<code>Sleep()</code>
<code>Wait()</code> method belongs to <code>Object</code> class.	<code>Sleep()</code> method belongs to <code>Thread</code> class.
<code>Wait()</code> method releases lock during <code>Synchronization</code> .	<code>Sleep()</code> method does not release the lock on object during

	Synchronization.
Wait() should be called only from Synchronized context.	There is no need to call sleep() from Synchronized context.
Wait() is not a static method.	Sleep() is a static method.
Wait() Has Three Overloaded Methods: <ul style="list-style-type: none"> • wait() • wait(long timeout) • wait(long timeout, int nanos) 	Sleep() Has Two Overloaded Methods: <ul style="list-style-type: none"> • sleep(long millis) millis: milliseconds • sleep(long millis, int nanos) nanos: Nanoseconds
public final void wait(long timeout)	public static void sleep(long millis) throws InterruptedException

69) Q- What is marker interface?

An **Interface** that does not contain methods, fields, and constants is known as **marker interface**.

In other words, an empty interface is known as **marker interface** or **tag interface**.

Types of marker interface

- Cloneable Interface
- Serializable Interface
- Random access Interface

70) Q- what is deep & shallow?

Clone () method:

- The process of creating exactly duplicate object is called cloning.
- The main objective of cloning is to maintain backup.
- That is if something goes wrong we can recover the situation by using backup copy.
- We can perform cloning by using clone() method of Object class.

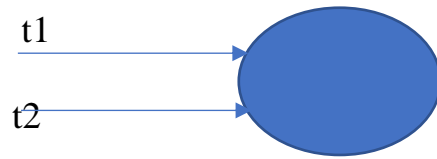
protected native object clone() throws CloneNotSupportedException;

Shallow Cloning

The process of creating just duplicate reference variable but not duplicate object is called shallow cloning.

Example: Test t1=new Test();
Test t2=t1;

Diagram: •



Deep Cloning

The process of creating exactly independent duplicate object is called deep cloning.

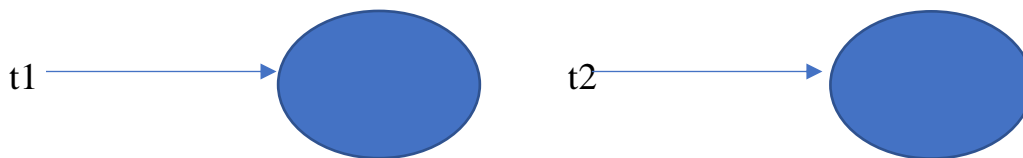
Example: `Test t1=new Test();`

`Test t2=(Test)t1.clone();`

`System.out.println(t1==t2);//false`

`System.out.println(t1.hashCode()==t2.hashCode());//false` Diagram:

- Cloning by default deep cloning.



71) Q- What is reflection?

Java reflection is useful because it supports dynamic retrieval of information about classes and data structures by name, and allows for their manipulation within an executing Java program. This feature is extremely powerful and has no equivalent in other conventional languages such as C, C++, Fortran, or Pascal.

72) Q- is java reference or by value?

Java and C is always call **by value**.

The term call by **reference** strictly applies to C++

73) Q-What is collection?

If we want to represent group of individual objects as single entity called as collection.

74) Q- What is the difference between hashset & hashtable?

Hashset	Hashtable
It uses add method to insert the new element	It uses put method to insert the new element
It can have only single null value	It doesn't allow null for key as well as for value
Like Hashmap ,it is not thread safe	It is thread safe
It is not synchronized	It is synchronized

75) WHAT IS PREDICATE

Java Predicate Interface

It is a functional interface which represents a predicate (boolean-valued function) of one argument. It is defined

in the java.util.function package and contains test() a functional method.

Java Predicate Interface Methods

METHODS	DESCRIPTION
boolean test(T t)	It evaluates this predicate on the given argument.
default Predicate<T> and(Predicate<? super T> other)	It returns a composed predicate that represents a short-circuiting logical AND of this predicate and another. When evaluating the composed predicate, if this predicate is false, then the other predicate is not evaluated.
default Predicate<T> negate()	It returns a predicate that represents the logical negation of this predicate.
default Predicate<T> or(Predicate<? super T> other)	It returns a composed predicate that represents a short-circuiting logical OR of this predicate and another. When evaluating the composed predicate, if this predicate is true, then the other predicate is not evaluated.
static <T> Predicate<T> isEqual(Object targetRef)	It returns a predicate that tests if two arguments are equal according to Objects.equals(Object, Object).

Example 1

```
import java.util.function.Predicate;
public class PredicateInterfaceExample {
    public static void main(String[] args) {
        Predicate<Integer> pr = a -> (a > 18); // Creating predicate
        System.out.println(pr.test(10)); // Calling Predicate method
    }
}
```

Output:

```
false
```

Example 2

```
import java.util.function.Predicate;
public class PredicateInterfaceExample {
    static Boolean checkAge(int age){
        if(age>17)
            return true;
        else return false;
    }
    public static void main(String[] args){
        // Using Predicate interface
        Predicate<Integer> predicate = PredicateInterfaceExample::checkAge;
        // Calling Predicate method
    }
}
```

```
        boolean result = predicate.test(25);
        System.out.println(result);
    }
}
```

Output:

```
true
```

76) Design Patterns in Java

A design patterns are **well-proved solution** for solving the specific problem/task..

Problem

Suppose you want to create a class for which only a single instance (or object) should be created and that single object can be used by all other classes.

Solution:

Singleton design pattern is the best solution of above specific problem. So, every design pattern has **some specification or set of rules** for solving the problems. What are those specifications, you will see later in the types of design patterns.

But remember one-thing, design patterns are programming language independent strategies for solving the common object-oriented design problems. That means, a design pattern represents an idea, not a particular implementation.

By using the design patterns you can make your code more flexible, reusable and maintainable. It is the most important part because java internally follows design patterns.

Advantage of design pattern:

1. They are reusable in multiple projects.
2. They provide the solutions that help to define the system architecture.
3. They capture the software engineering experiences.
4. They provide transparency to the design of an application.
5. They are well-proved and testified solutions since they have been built upon the knowledge and experience of expert software developers.
6. Design patterns don't guarantee an absolute solution to a problem. They provide clarity to the system architecture and the possibility of building a better system.

When should we use the design patterns?

We must use the design patterns **during the analysis and requirement phase of SDLC**(Software Development Life Cycle).

Design patterns ease the analysis and requirement phase of SDLC by providing information based on prior hands-on experiences.

Categorization of design patterns:

Basically, design patterns are categorized into two parts:

1. Core Java (or JSE) Design Patterns.
2. JEE Design Patterns.

Core Java Design Patterns

In core java, there are mainly three types of design patterns, which are further divided into their sub-parts:

1.Creational Design Pattern

1. Factory Pattern
2. Abstract Factory Pattern
3. Singleton Pattern
4. Prototype Pattern
5. Builder Pattern.

2. Structural Design Pattern

1. Adapter Pattern
2. Bridge Pattern
3. Composite Pattern
4. Decorator Pattern
5. Facade Pattern
6. Flyweight Pattern
7. Proxy Pattern

3. Behavioral Design Pattern

1. Chain Of Responsibility Pattern
2. Command Pattern
3. Interpreter Pattern
4. Iterator Pattern
5. Mediator Pattern

6. Memento Pattern
7. Observer Pattern
8. State Pattern
9. Strategy Pattern
10. Template Pattern
11. Visitor Pattern

Singleton design pattern in Java

1. Singleton design pattern in Java
2. Advantage of Singleton Pattern
3. Usage of Singleton Pattern
4. Example of Singleton Pattern

Singleton Pattern says that just "**define a class that has only one instance and provides a global point of access to it**".

In other words, a class must ensure that only single instance should be created and single object can be used by all other classes.

There are two forms of singleton design pattern

- **Early Instantiation:** creation of instance at load time.
- **Lazy Instantiation:** creation of instance when required.

Advantage of Singleton design pattern

- Saves memory because object is not created at each request. Only single instance is reused again and again.

Usage of Singleton design pattern

- Singleton pattern is mostly used in multi-threaded and database applications. It is used in logging, caching, thread pools, configuration settings etc.

How to create Singleton design pattern?

To create the singleton class, we need to have static member of class, private constructor and static factory method.

- **Static member:** It gets memory only once because of static, it contains the instance of the Singleton class.
- **Private constructor:** It will prevent to instantiate the Singleton class from outside the class.
- **Static factory method:** This provides the global point of access to the Singleton object and returns the instance to the caller.

Understanding early Instantiation of Singleton Pattern

In such case, we create the instance of the class at the time of declaring the static data member, so instance of the class is created at the time of classloading.

Let's see the example of singleton design pattern using early instantiation.

File: A.java

```
1. class A{
2.     private static A obj=new A();//Early, instance will be created at load time
3.     private A(){}
4.
5.     public static A getA(){
6.         return obj;
7.     }
8.
9.     public void doSomething(){
10.        //write your code
11.    }
12. }
```

Understanding lazy Instantiation of Singleton Pattern

In such case, we create the instance of the class in synchronized method or synchronized block, so instance of the class is created when required.

Let's see the simple example of singleton design pattern using lazy instantiation.

File: A.java

```
1. class A{
2.     private static A obj;
3.     private A(){}
4.
5.     public static A getA(){
6.         if (obj == null){
7.             synchronized(Singleton.class){
8.                 if (obj == null){
```

```
9.         obj = new Singleton();//instance will be created at request time
10.     }
11. }
12. }
13. return obj;
14. }
15.
16. public void doSomething(){
17. //write your code
18. }
19. }
```

Significance of Classloader in Singleton Pattern

If singleton class is loaded by two classloaders, two instance of singleton class will be created, one for each classloader.

Significance of Serialization in Singleton Pattern

If singleton class is Serializable, you can serialize the singleton instance. Once it is serialized, you can deserialize it but it will not return the singleton object.

To resolve this issue, you need to override the **readResolve()** method that enforces the singleton. It is called just after the object is deserialized. It returns the singleton object.

```
1. public class A implements Serializable {
2.     //your code of singleton
3.     protected Object readResolve() {
4.         return getA();
5.     }
6.
7. }
```

Example of Builder Design Pattern

To create simple example of builder design pattern, you need to follow 6 following steps.

1. Create Packing interface
 2. Create 2 abstract classes CD and Company
 3. Create 2 implementation classes of Company: Sony and Samsung
 4. Create the CDType class
 5. Create the CDBuilder class
 6. Create the BuilderDemo class
-

1) Create Packing interface

File: Packing.java

1. **public interface** Packing {
2. **public** String pack();
3. **public int** price();
4. }

2) Create 2 abstract classes CD and Company

Create an abstract class CD which will implement Packing interface.

File: CD.java

1. **public abstract class** CD **implements** Packing{
2. **public abstract** String pack();
3. }

File: Company.java

1. **public abstract class** Company **extends** CD{
2. **public abstract int** price();
3. }

3) Create 2 implementation classes of Company: Sony and Samsung

File: Sony.java

1. **public class** Sony **extends** Company{
2. @Override
3. **public int** price(){
4. **return** 20;
5. }
6. @Override
7. **public** String pack(){
8. **return** "Sony CD";
9. }
10. }//End of the Sony class.

File: Samsung.java

```
1. public class Samsung extends Company {
2.     @Override
3.     public int price(){
4.         return 15;
5.     }
6.     @Override
7.     public String pack(){
8.         return "Samsung CD";
9.     }
10. }//End of the Samsung class.
```

4) Create the CDType class

File: CDType.java

```
1. import java.util.ArrayList;
2. import java.util.List;
3. public class CDType {
4.     private List<Packing> items=new ArrayList<Packing>();
5.     public void addItem(Packing packs) {
6.         items.add(packs);
7.     }
8.     public void getCost(){
9.         for (Packing packs : items) {
10.             packs.price();
11.         }
12.     }
13.     public void showItems(){
14.         for (Packing packing : items){
15.             System.out.print("CD name : "+packing.pack());
16.             System.out.println(", Price : "+packing.price());
17.         }
18.     }
19. }//End of the CDType class.
```

5) Create the CDBuilder class

File: CDBuilder.java

```
1. public class CDBuilder {
2.     public CDType buildSonyCD(){
3.         CDType cds=new CDType();
4.         cds.addItem(new Sony());
```

```

5.         return cds;
6.     }
7.     public CDType buildSamsungCD(){
8.         CDType cds=new CDType();
9.         cds.addItem(new Samsung());
10.        return cds;
11.    }
12. }// End of the CDBuilder class.

```

6) Create the BuilderDemo class

File: BuilderDemo.java

```

1. public class BuilderDemo{
2.     public static void main(String args[]){
3.         CDBuilder cdBuilder=new CDBuilder();
4.         CDType cdType1=cdBuilder.buildSonyCD();
5.         cdType1.showItems();
6.
7.         CDType cdType2=cdBuilder.buildSamsungCD();
8.         cdType2.showItems();
9.     }
10. }

```

Output of the above example

1. CD name : Sony CD, Price : 20
2. CD name : Samsung CD, Price : 15

77) Covariant Return Type

The covariant return type specifies that the return type may vary in the same direction as the subclass.

Before Java5, it was not possible to override any method by changing the return type. But now, since Java5, it is possible to override method by changing the return type if subclass overrides any method whose return type is Non-Primitive but it changes its return type to subclass type. Let's take a simple example:

FileName: B1.java

```

1. class A{
2.     A get(){return this;}
3. }
4.
5. class B1 extends A{
6.     @Override

```

```

7. B1 get(){return this;}
8. void message(){System.out.println("welcome to covariant return type");}
9.
10. public static void main(String args[]){
11. new B1().get().message();
12. }
13. }

```

Output:

```
welcome to covariant return type
```

As you can see in the above example, the return type of the get() method of A class is A but the return type of the get() method of B class is B. Both methods have different return type but it is method overriding. This is known as covariant return type.

Advantages of Covariant Return Type

Following are the advantages of the covariant return type.

- 1) Covariant return type assists to stay away from the confusing type casts in the class hierarchy and makes the code more usable, readable, and maintainable.
- 2) In the method overriding, the covariant return type provides the liberty to have more to the point return types.
- 3) Covariant return type helps in preventing the run-time *ClassCastException* on returns.

Let's take an example to understand the advantages of the covariant return type.

78) Q.what are functional interfaces?

Answer:

A functional interface is an interface that contains only one abstract method.

From Java 8 onwards, [lambda expressions](#) can be used to represent the instance of a functional interface.

A functional interface can have any number of default,static methods.

examples : ***Runnable, ActionListener, Comparable***

@FunctionalInterface annotation is used to ensure that the functional interface can't have more than one abstract method. In case more than one abstract method is present, the compiler flags an 'Unexpected @FunctionalInterface annotation' message. However, it is not mandatory to use this annotation.

example:

```

public interface MyFunctionalInterface {
    public void execute();
}

```

another example:

```

public interface MyFunctionalInterface2{
    public void execute();
    public default void print(String text) {
        System.out.println(text);
    }
    public static void print(String text, PrintWriter writer) throws IOException {
        writer.write(text);
    }
}

```

The above interface also a functional interface in because it only contains a single Abstract method.

79) What is solid principle?

Answer:

SOLID principles are object-oriented design concepts relevant to software development. SOLID is an acronym for five other class-design principles:

Single Responsibility Principle,
 Open-Closed Principle,
 Liskov Substitution Principle,
 Interface Segregation Principle, and
 Dependency Inversion Principle.

Principle	Description
Single Responsibility Principle	Each class should be responsible for a single part or functionality of the system.
Open-Closed Principle	Software components should be open for extension, but not for modification.
Liskov Substitution Principle	Objects of a superclass should be replaceable with objects of its subclasses without breaking the system.
Interface Segregation Principle	No client should be forced to depend on methods that it does not use.
Dependency Inversion Principle	High-level modules should not depend on low-level modules, both should depend on

Principle	Description
	abstractions.

80) @Qualifier, @bean, @componentscan, @RestController, @controller,

@Qualifier Annotation -

To eliminate the issue of which bean needs to be injected.

Ex.

```
public class FooService {

    @Autowired
    @Qualifier("fooFormatter")
    private Formatter formatter;
}
```

By including the @Qualifier annotation, provide the name of the specific implementation we want to use to avoid ambiguity when Spring finds multiple beans of the same type.

@bean

@Bean Annotation is applied on a method to specify that it returns a bean to be managed by Spring context. Spring Bean annotation is usually declared in Configuration classes methods. In this case, bean methods may reference other @Bean methods in the same class by calling them directly.

EX.

```
public class MyDAOBean {

    @Override
    public String toString() {
        return "MyDAOBean"+this.hashCode();
    }
}
```

Here is a Configuration class where we have defined a @Bean method for MyDAOBean class.

```
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class MyAppConfiguration {
```



```

    @Bean
    public MyDAOBean getMyDAOBean() {
        return new MyDAOBean();
    }
}

```

@ComponentScan

1. What is component scan?

The process of discovering Spring Components within classpath of the application is called Component Scanning in Spring. Later spring will use them to add in application context.

The classes annotated

with **@Component**, **@Controller**, **@Service**, **@Repository**, **@Configuration**, **@RestController** etc will be treated as Spring Components.

@ComponentScan annotation tells **Spring** that where to look for Spring Components explicitly. **@ComponentScan** annotation is used with **@Configuration** annotation.

Using @ComonentScan without attribute

@ComonentScan without attribute tells that Spring to discover components in current package and within it's sub packages

Ex.

@ComponentScan

@Configuration

```

public class ComponentScanNoAttributeConfig {
}

```

@ComonentScan with attribute

We can tell Spring to which packages to scan specifically by using **basePackages** attribute. Let's see few examples.

5.1. Scanning Base Packages using @ComonentScan

Following example demonstrates how to scan components only within **reptiles** and its sub packages specifically.

@Configuration

@ComponentScan(basePackages = "com.javabydeveloper.spring.bean.animal.reptiles")

```

public class ReptilesConfig {
}

```

5.2. Scanning Multiple Packages

We can specify multiple packages using **basePackages** attribute. Let's see an example.

@Configuration

@ComponentScan(basePackages = {"com.javabydeveloper.spring.bean.animal.amphibians", "com.javabydeveloper.spring.bean.animal.reptiles.crocodles"})

```

public class ScanMultiplePackageConfig {
}

```

5.3. Using basePackageClasses attribute

basePackageClasses is type-safe alternative to **basePackages** for specifying the packages to scan for annotated components. The package of each class specified will be scanned. In following example **BlackMamba** belongs to **snakes** package and **Frog** belongs to **amphibians** package.

@Configuration

```
@ComponentScan(basePackageClasses = {BlackMamba.class, Frog.class})
```

```
public class ScanBasePackageClassesConfig {  
}
```

@RestController

annotation in order to simplify the creation of RESTful web services. **It's a convenient annotation that combines @Controller and @ResponseBody**, which eliminates the need to annotate every request handling method of the controller class with the @ResponseBody annotation.

@Controller

We can annotate classic controllers with the @Controller annotation. This is simply a specialization of the @Component class, which allows us to auto-detect implementation classes through the classpath scanning.

We typically use @Controller in combination with a @RequestMapping annotation for request handling methods.

81) What is the difference between fetch & pull?

Git Fetch	Git Pull
Git fetch fetches the required information only to the local repository.	Git pull fetches the required information not only to the local repository but also to the workspace that you are currently working in.
In Github fetch, the content of the specified branch is only downloaded.	In Github pull, the content of the specified branch is downloaded, and also the changes are committed to the local repository.
Its main function is to fetch the content.	Its main function is a combination of fetch and merges the content.

It has only command-line syntax.	It has command-line syntax as well as a pull request to post the changes.
Command used: git fetch <remote> <branch>	Command used: git pull <branch>

82) what is the difference between Push & pull?

PUSH

The `git push` command is used to transfer or push the commit, which is made on a local branch in your computer to a remote repository like GitHub. The command used for pushing to GitHub is given below.

```
git push 'remote_name' 'branch_name'
```

PULL

If you make a change in a repository, GIT PULL can allow others to view the changes. It is used to acknowledge the change that you've made to the repository that you're working on. Or also called a target repository.

The simple command to PULL from a branch is:

```
git pull 'remote_name' 'branch_name'.
```

The `git pull` command is a combination of `git fetch` which fetches the recent commits in the local repository and `git merge`, which will merge the branch from a remote to a local branch also 'remote_name' is the repository name and 'branch_name' is the name of the specific branch.

83) What is the save and persist method?

Sr. No.	Key	save()	persist()
1	Basic	It stores object in database	It also stores object in database
2	Return Type	It return generated id and return type is serializable	It does not return anything. Its void return type.
3	Transaction Boundaries	It can save object within boundaries and outside boundaries	It can only save object within the transaction boundaries
4.	Detached Object	It will create a new row in the table for detached object	It will throw persistence exception for detached object
5.	Supported by	It is only supported by Hibernate	It is also supported by JPA

84) what is the difference between get & post method?

GET REQUEST	POST REQUEST
GET retrieves a representation of the specified resource.	POST is for writing data, to be processed to the identified resource.
It typically has relevant information in the URL of the request.	It typically has relevant information in the body of the request.
It is limited by the maximum length of the URL supported by the browser and web server.	It does not have such limits.
It is the default HTTP method.	In this we need to specify the method as POST to send a request with the POST method.
You can bookmark GET request.	You cannot bookmark POST request.
It is less secure because data sent is part of	It is a little safer because the parameters are

the URL	not stored in browser history or in web server logs.
It is cacheable.	It is not cacheable.
Ex. GET the page showing a particular question.	Ex. Send a POST request by clicking the “Add to cart” button.

85) Q.Access modifiers in java

which are used for setting the access level to classes, variables, methods, and constructors.

Simply put, there are four access modifiers: *public*, *private*, *protected* and *default* (no keyword).

Before we begin let's note that a top-level class can use *public* or *default* access modifiers only. At the member level, we can use all four.

Default -**all members are visible within the same package**

Public - all other classes in all packages will be able to use it

Private

-**is accessible from the same class only**

Protected

- **same package and in addition from all subclasses of its class**

Modifier	Class	Package	Subclass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	N
default	Y	Y	N	N
private	Y	N	N	N

86) When to use arraylist and linkedlist

1. Underlying Data Structure

The first difference between ArrayList and LinkedList comes with the fact that ArrayList is backed by Array while LinkedList is backed by LinkedList.

2. LinkedList implements Deque

Another difference between ArrayList and LinkedList is that apart from the List interface, LinkedList also implements the Deque interface, which provides first in first out operations.

Also, LinkedList is implemented as a doubly-linked list and for index-based operation,

navigation can happen from either end

3. Adding elements in ArrayList

Adding an element in ArrayList is $O(1)$ operation if it doesn't trigger re-size of Array, in which case it becomes $O(\log(n))$, On the other hand, appending an element in LinkedList is $O(1)$ operation

4. Removing an element from a position

In order to remove an element from a particular index e.g. by calling `remove(index)`, ArrayList performs a copy operation which makes it close to $O(n)$ while LinkedList needs to traverse to that point which also makes it $O(n/2)$, as it can traverse from either direction .

5. Retrieving element from a position

The `get(index)` operation is $O(1)$ in ArrayList while its $O(n/2)$ in LinkedList, as it needs to traverse till that entry. Though, in Big O notation $O(n/2)$ is just $O(n)$ because we ignore constants there.

87) Spring Scopes

There are five types of [spring bean](#) scopes:

1. **singleton** – only one instance of the spring bean will be created for the spring container. This is the default spring bean scope. While using this scope, make sure bean doesn't have shared instance variables otherwise it might lead to data inconsistency issues.
2. **prototype** – A new instance will be created every time the bean is requested from the spring container.
3. **request** – This is same as prototype scope, however it's meant to be used for web applications. A new instance of the bean will be created for each HTTP request.
4. **session** – A new bean will be created for each HTTP session by the container.
5. **global-session** – This is used to create global session beans for Portlet applications.

88) Servletcontext and ServletConfig

Servlet Config	Servlet Context
Servlet config object represent single servlet	It represent whole web application running on particular JVM and common for all the servlet
Its like local parameter associated with particular servlet	Its like global parameter associated with whole application
It's a name value pair defined inside the servlet section of web.xml file so it has servlet wide scope	ServletContext has application wide scope so define outside of servlet tag in web.xml file.
<code>getServletConfig()</code> method is used to get the config object	<code>getContext()</code> method is used to get the context object.
for example shopping cart of a user is a specific to particular user so here we can use servlet config	To get the MIME type of a file or application session related information is stored using servlet context object.

89) Best practice used for Collection

Ans: Here's the list:

1. Choosing the right collections

2. Always using interface type when declaring a collection
3. Use generic type and diamond operator
4. Specify initial capacity of a collection if possible
5. Prefer isEmpty() over size()
6. Do not return null in a method that returns a collection
7. Do not use the classic for loop
8. Favor using forEach() with Lambda expressions
9. Overriding equals() and hashCode() properly
10. Implementing the Comparable interface properly
11. Using Arrays and Collections utility classes
12. Using the Stream API on collections
13. Prefer concurrent collections over synchronized wrappers
14. Using third-party collections libraries
15. Eliminate unchecked warnings
16. Favor generic types
17. Favor generic methods
18. Using bounded wildcards to increase API flexibility

90) .How to change embedded server in spring boot ?

Ans : You will need to **update pom. xml** add the dependency for spring-boot-starter-jetty . Also, you will need to exclude default added spring-boot-starter-tomcat dependency.

91) .Hibernate annotations

Ans :

Hibernate JPA Annotations - Contents:

Annotation	Package Detail/Import statement
@Entity	import javax.persistence.Entity;
@Table	import javax.persistence.Table;
@Column	import javax.persistence.Column;
@Id	import javax.persistence.Id;
@GeneratedValue	import javax.persistence.GeneratedValue;
@Version	import javax.persistence.Version;
@OrderBy	import javax.persistence.OrderBy;
@Transient	import javax.persistence.Transient;
@Lob	import javax.persistence.Lob;
Hibernate Association Mapping Annotations	
@OneToOne	import javax.persistence.OneToOne;


@ManyToOne	import javax.persistence.ManyToOne;
@OneToMany	import javax.persistence.OneToMany;
@ManyToMany	import javax.persistence.ManyToMany;
@PrimaryKeyJoinColumn	import javax.persistence.PrimaryKeyJoinColumn;
@JoinColumn	import javax.persistence.JoinColumn;
@JoinTable	import javax.persistence.JoinTable;
@MapsId	import javax.persistence.Id;
Hibernate Inheritance Mapping Annotations	
@Inheritance	import javax.persistence.Inheritance;
@DiscriminatorColumn	import javax.persistence.DiscriminatorColumn;
@DiscriminatorValue	import javax.persistence.DiscriminatorValue;

92) . Java version u have used ?

Ans : I am using java version "1.8.0_241"

93) .Difference between method and constructor

Ans :

CONSTRUCTOR	VERSUS	METHOD
Constructor		Method
Constructors are special type of methods used to initialize objects of its class.		Methods are a set of instructions that are invoked at any point in a program to return a result.
The purpose of a constructor is to create an instance of a class.		The purpose of a method is to execute Java code.
They are called implicitly immediately upon object creation.		They can be called directly without even creating an instance of that class.
They are used to initialize objects that don't exist.		They perform operations on already created objects.
The name of the constructor must be same as the name of the class.		They can have arbitrary names in Java.
They are not inherited by subclasses.		They are inherited by subclasses.
		

94) . Normalization

Ans : Normalization is the process of organizing the data in the database.

Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate the undesirable characteristics like Insertion, Update and Deletion Anomalies.

A normalization defines rules for the relational table as to whether it satisfies the normal form. A **normal form** is a process that evaluates each relation against defined criteria and removes the multivalued, joins, functional and trivial dependency from a relation. If any data is updated, deleted or inserted, it does not cause any problem for database tables and help to improve the relational table' integrity and efficiency.

95) .why string is immutable ?

Ans : The String is immutable in Java because **of the security, synchronization and concurrency, caching, and class loading**. The reason of making string final is to destroy the immutability and to not allow others to

extend it. The String objects are cached in the String pool, and it makes the String immutable.

96) .features of REST full web services ?

Ans **These are the features of REST services:**

- Client-Server. REST services must be based on a Client-Server architecture. ...
- No condition. ...
- Cache-enabled information. ...
- Consistent interface. ...
- Resource access by name. ...
- Related resources. ...
- Answer in a known format.

97) .Dependency injection methods

Ans : A class is no longer responsible for creating the objects it requires, and it does not have to delegate instantiation to a factory object as in the Abstract Factory design pattern. There are three types of dependency injection —

**constructor injection,
method injection and
property injection**

98) .Spring hibernate integration

Ans : We can simply integrate **hibernate application with spring application**.

In hibernate framework, we provide all the database information hibernate.cfg.xml file.

But if we are going to integrate the hibernate application with spring, we don't need to create the hibernate.cfg.xml file. We can provide all the information in the applicationContext.xml file.

Advantages :

The Spring framework provides **HibernateTemplate** class, so you don't need to follow so many steps like create Configuration, BuildSessionFactory, Session, beginning and committing transaction etc.

99) Roles and responsibilities

Ans : The roles and responsibilities of a Java developer/engineer will vary greatly depending on the company and specific position. Here are some typical responsibilities:

- Designing, implementing, and maintaining Java applications that are often high-volume and low-latency, required for mission-critical systems
- Delivering **high availability** and performance
- Contributing in all phases of the development lifecycle
- Writing well-designed, efficient, and testable code
- Conducting software analysis, programming, testing, and debugging
- Managing Java and Java EE application development
- Ensuring designs comply with specifications
- Preparing and producing releases of software components
- Transforming requirements into stipulations
- Support continuous improvement
 - Investigating alternatives and technologies

- Presenting for architectural review

100) .load and get difference

Ans :

	get ()	load ()
1	Return value null is possible	Never returns null
2	Fast if record exists	Slow if record exists
3	Used to retrieve object (record)	Used for delete etc. operations
4	Eager fetching	Lazy fetching
5	Always hits the database	Not always hits
6	Does not return proxy object	Always returns a proxy object
7	Performance-wise is slow as it may have to make number of rounds to database to get data	Better performance. If already in the cache, the record is available to much difference
8	As it returns null if no record exist, the execution continues	It throws ObjectNotFoundException, if record not found. Execution terminates if not handled successfully

101) .Eager and lazy difference

Lazy vs. Eager Learning

Lazy: Wait for query before generalizing

- k -nearest neighbor, case-based reasoning

Eager: Generalize before seeing query

- ID3, FOIL, Naive Bayes, neural networks, ...

Does it matter?

- Eager learner must create global approximation
- Lazy learner can create many local approximations
- If they use same H , lazy can represent more complex functions (e.g., consider H = linear functions)

102) .java 8 features

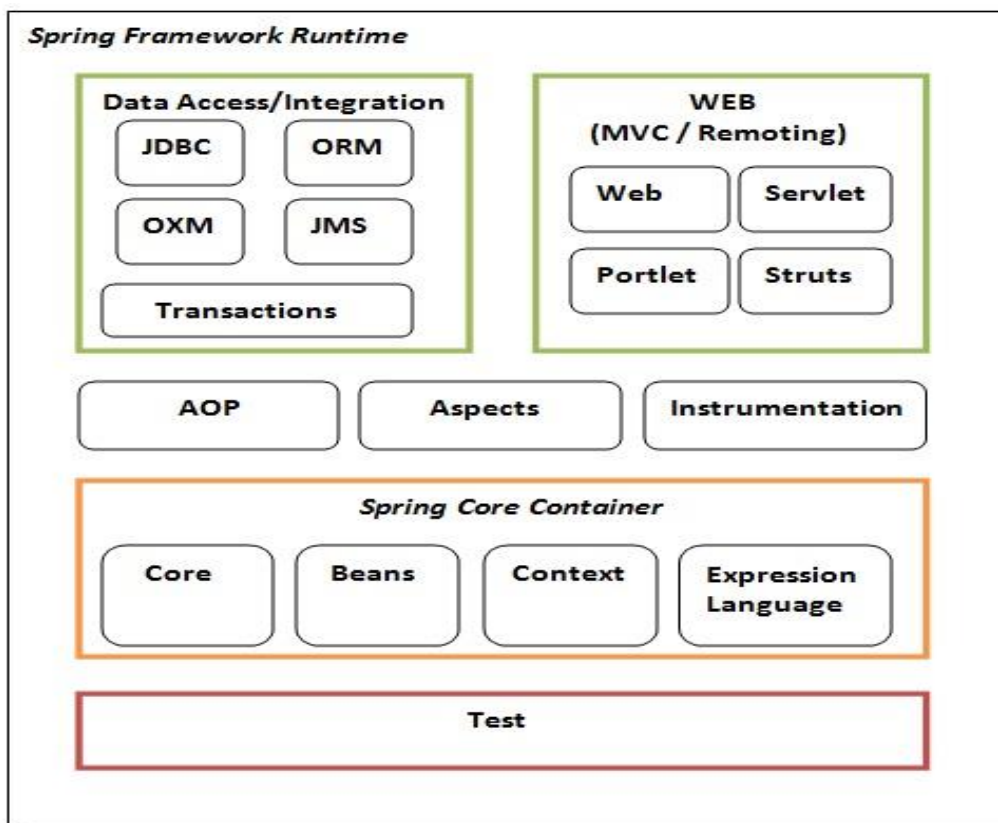
Ans : java 8 provides following features for Java Programming:

- Lambda expressions,
- Method references,

- Functional interfaces,
- Stream API,
- Default methods,
- Base64 Encode Decode,
- Static methods in interface,
- Optional class,
- Collectors class,
- ForEach() method,
- Parallel array sorting,
- Nashorn JavaScript Engine,
- Parallel Array Sorting,
- Type and Repating Annotations,
- IO Enhancements,
- Concurrency Enhancements,
- JDBC Enhancements etc.

103) .modules of spring framework

Ans : The Spring framework comprises of many modules such as core, beans, context, expression language, AOP, Aspects, Instrumentation, JDBC, ORM, OXM, JMS, Transaction, Web, Servlet, Struts etc. These modules are grouped into Test, Core Container, AOP, Aspects, Instrumentation, Data Access / Integration, Web (MVC / Remoting) as displayed in the following diagram.



104) .scopes of spring beans

Ans : The Spring Framework supports the following five scopes, three of which are available only if you use a web-aware ApplicationContext.

Sr.No.	Scope & Description
1	singleton This scopes the bean definition to a single instance per Spring IoC container (default).
2	prototype This scopes a single bean definition to have any number of object instances.
3	request This scopes a bean definition to an HTTP request. Only valid in the context of a web-aware Spring ApplicationContext.
4	session This scopes a bean definition to an HTTP session. Only valid in the context of a web-aware Spring ApplicationContext.
5	global-session This scopes a bean definition to a global HTTP session. Only valid in the context of a web-aware Spring ApplicationContext.

105) .how to implements joins in hibernate

Ans : **Following is a step by step guide to show hibernate join statement.**

1. 4.1 Create a Maven project. Create a new maven project in eclipse. ...
2. 4.2 Add Hibernate Dependencies. ...
3. 4.3 Create Hibernate Configuration File. ...
4. 4.4 Create entities. ...
5. 4.5 Map Java objects to database. ...—
6. 4.6 Hibernate Test Program.

106) .collection type in hibernate

Ans : The persistent collections injected by Hibernate behave like **HashMap , HashSet , TreeMap , TreeSet or ArrayList** , depending on the interface type. Collections instances have the usual behavior of value types.

16.to design rest full web services what points should consider

Ans : **10 things you should do to write effective RESTful Web services**

- 1: Don't look for an official "REST standard" ...
- 2: Adhere to standards anyway. ...
- 3: Make sure your documentation is flawless. ...
- 4: Provide JSON output. ...
- 5: Don't forget XML. ...

- 6: Understand the verbs. ...
 - 7: Understand the importance of URI routing.
- 8: Do not make changes without versioning the service
- 9: Stay in contact with your users
- 10: Provide sample code

107) how to select unique record from table-which keyword is use

Ans : The **SQL DISTINCT keyword** is used in conjunction with the SELECT statement to eliminate all the duplicate records and fetching only unique records.

108) how u handle exception in spring boot

Ans : Exception Handler

The **@ExceptionHandler** is an annotation used to handle the specific exceptions and sending the custom responses to the client. Define a class that extends the RuntimeException class. You can define the @ExceptionHandler method to handle the exceptions as shown.

109) what is functional interface

Ans : A functional interface is **an interface that contains only one abstract method**. They can have only one functionality to exhibit. ... A functional interface can have any number of default methods. Runnable, ActionListener, Comparable are some of the examples of functional interfaces.

110) why lambda function come into picture

Ans :

lambda expressions are added in Java 8 and provide below functionalities. **Enable to treat functionality as a method argument, or code as data**. A function that can be created without belonging to any class. A lambda expression can be passed around as if it was an object and executed on demand.

111) reverse array without using other array program.

```
-> package com.devglan.set2;
import java.util.Arrays;
public class ArrayReverse {
    public int[] reverse(int [] array){
        if(array == null || array.length <= 1){
            System.out.println("Invalid array.");
        }
        for (int i = 0; i < array.length / 2; i++) {
            int temp = array[i];
            array[i] = array[array.length - 1 - i];
            array[array.length - 1 - i] = temp;
        }
        return array;
    }
    public static void main(String[] args){
        ArrayReverse arrayReverse = new ArrayReverse();
        int[] input = {1, 2, 3, 4, 5, 6, 7, 8};
```

```

System.out.println("Original array" + Arrays.toString(input));
System.out.println("Reversed array" + Arrays.toString(arrayReverse.reverse(input)));
}
}

```

112) find out duplicate from string in java

-> Define a string.

Two loops will be used to find the duplicate characters. Outer loop will be used to select a character and initialize variable count by 1.

Inner loop will compare the selected character with rest of the characters present in the string.

If a match found, it increases the count by 1 and set the duplicates of selected character by '0' to mark them as visited.

After inner loop, if count of character is greater than 1, then it has duplicates in the string.

Solution

```

public class DuplicateCharacters {
    public static void main(String[] args) {
        String string1 = "Great responsibility";
        int count;
        //Converts given string into character array
        char string[] = string1.toCharArray();
        System.out.println("Duplicate characters in a given string: ");
        //Counts each character present in the string
        for(int i = 0; i < string.length; i++) {
            count = 1;
            for(int j = i+1; j < string.length; j++) {
                if(string[i] == string[j] && string[i] != ' ') {
                    count++;
                }
                //Set string[j] to 0 to avoid printing visited character
                string[j] = '0';
            }
            //A character is considered as duplicate if count is greater than 1
            if(count > 1 && string[i] != '0')
                System.out.println(string[i]);
        }
    }
}

```

Output:

Duplicate characters in a given string:

```

r
e
t
s
i

```

113) what is immutable in java.

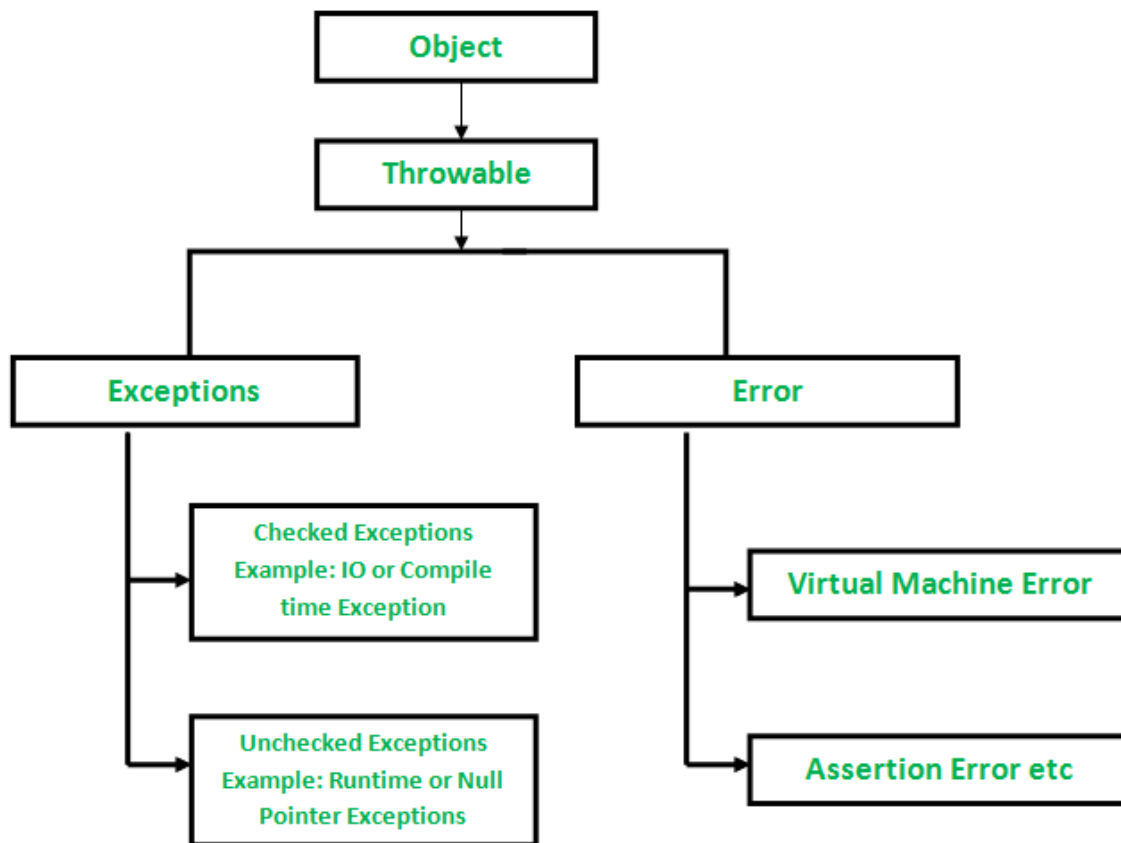
-> Immutable class in java means that once an object is created, we cannot change its content.

In Java, all the wrapper classes (like Integer, Boolean, Byte, Short) and String class is immutable. ... Data members in the class must be declared as final so that we can't change the value of it after object creation.

114> Exception hierarchy.

-> An exception is an unwanted or unexpected event, which occurs during the execution of a program i.e at run time, that disrupts the normal flow of the program's instructions.

All exception and errors types are sub classes of class Throwable, which is base class of hierarchy. One branch is headed by Exception. This class is used for exceptional conditions that user programs should catch. NullPointerException is an example of such an exception. Another branch, Error are used by the Java run-time system (JVM) to indicate errors having to do with the run-time environment itself (JRE). StackOverflowError is an example of such an error.



115> What is spring and spring boot.

Spring is an open-source lightweight framework widely used to develop enterprise applications. Spring Boot is built on top of the conventional spring framework, widely used to develop REST APIs. ... Spring Boot provides embedded servers such as Tomcat and Jetty etc

116> what is @RestController and @ResponseBody

@RestController annotation in order to simplify the creation of RESTful web services. It's a convenient annotation that combines @Controller and @ResponseBody, which eliminates the

need to annotate every request handling method of the controller class with the `@ResponseBody` annotation.

117> Collection hierarchy.

The **Collection in Java** is a framework that provides an architecture to store and manipulate the group of objects.

Java Collections can achieve all the operations that you perform on a data such as searching, sorting, insertion, manipulation, and deletion.

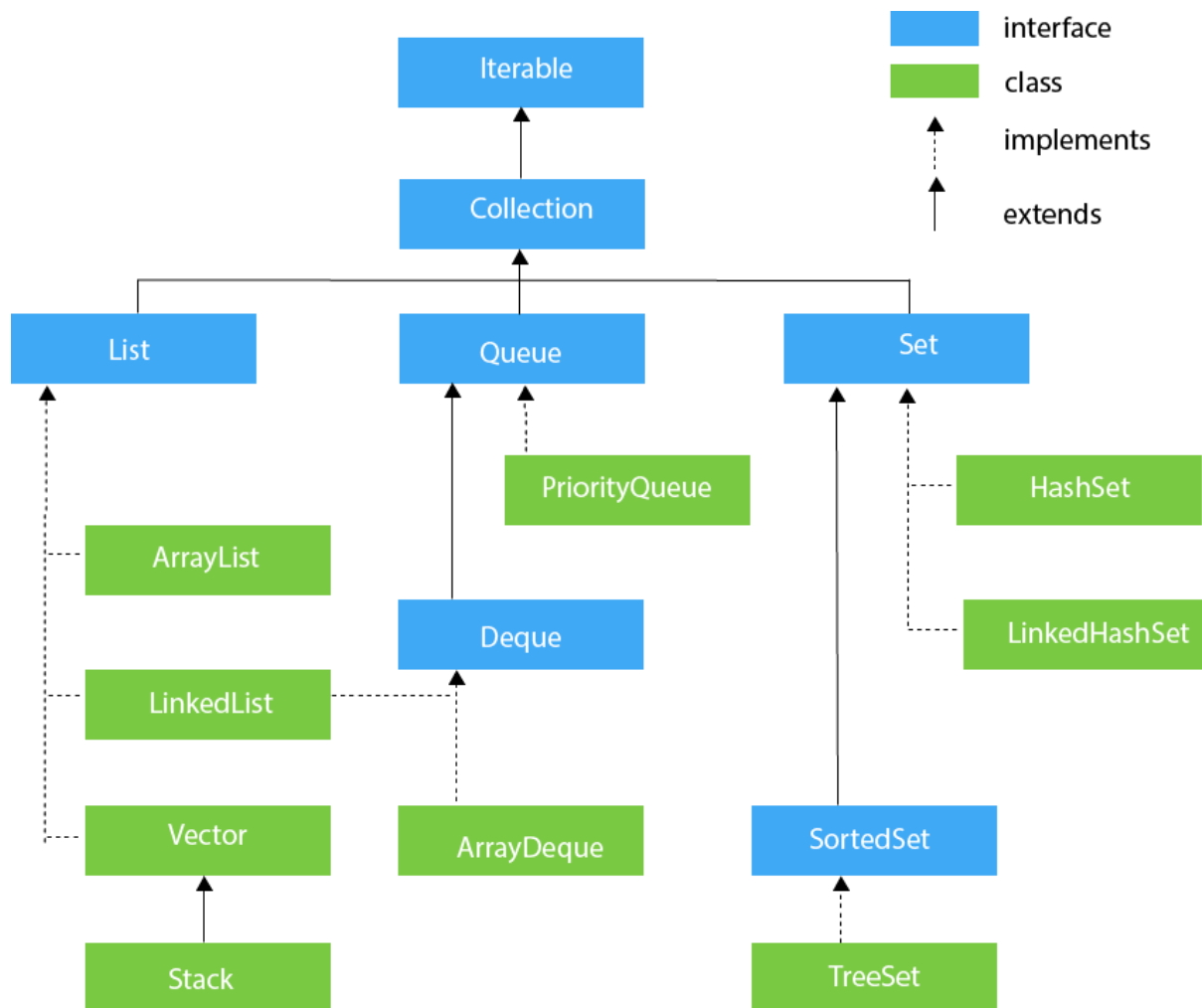
Java Collection means a single unit of objects. Java Collection framework provides many interfaces (Set, List, Queue, Deque) and classes ([ArrayList](#), Vector, [LinkedList](#), [PriorityQueue](#), HashSet, LinkedHashSet, TreeSet).

118) What is Collection in Java

A Collection represents a single unit of objects, i.e., a group.

119) Hierarchy of Collection Framework

Let us see the hierarchy of Collection framework. The **java.util** package contains all the [classes](#) and [interfaces](#) for the Collection framework.



120) . Internal working of HashMap?

Answer – Hashing -It is the process of converting an object into an integer value. The integer value helps in indexing and faster searches.

If you want to represent group of objects as key-value pair then you should go for map.

Example-

```
package com.velocity;
```

```
import java.util.HashMap;
```

```
public class HashMapDemo {
```

```
    public static void main(String[] args) {
        HashMap hm = new HashMap();
        hm.put("Jeevan", 10);
        hm.put("Kulkarni", 20);
        hm.put("Pune", 30);
        hm.put("Jeevan", 50);
        System.out.println(hm);
    }
```

```
}
```

Output-

```
{Pune=30, Kulkarni=20, Jeevan=10}
```

How hashCode() and equals() methods work into HashMap?

hashCode() method

hashCode method is used to get the hashcode of every object.

equals() method

equals method is used to check that 2 objects are equal or not. This method is provided by Object class. HashMap uses equals() to compare the key whether they are equal or not. If equals() method returns true, they are equal; otherwise, not equal.

How to calculate index internally?

Put(K k, V v) //Here put is the method having key and value pair

hash(k) → hash(jeevan) → 23560520

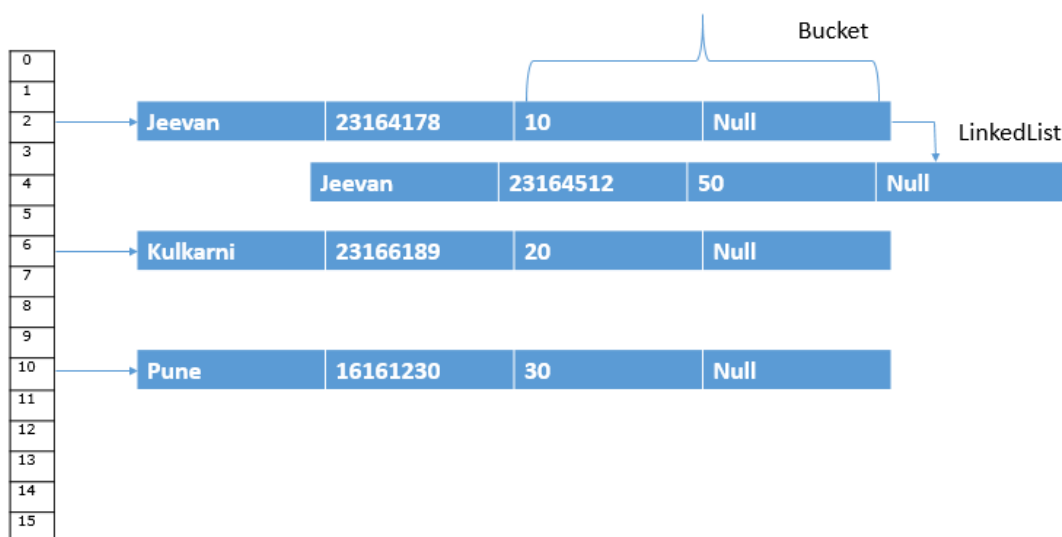
index = hash & (n-1) here index = 2

so jeevan will be stored at 2nd position.

Next time again I get the 2nd index or position then what will happen?

It will store it at 2nd and uses the linked list as shown in fig.

After that I get the 6th index so kulkarni will be stored at 6th position then next time get the index 10th then pune will be placed at 10th position.



How get method works here?

Now let's try some get method to get a value. get(K key) method is used to get a value by its key. If you don't know the key then it is not possible to fetch a value.

121) .Contract between hashCode & Equal method?

Answer - General Contract of equals() method

There are some general principles defined by [Java SE](#) that must be followed while implementing the equals() method in Java. The equals() method must-

- **reflexive**: An object x must be equal to itself, which means, for object x, **equals(x)** should return true.
- **symmetric**: for two given objects x and y, **x.equals(y)** must return true if and only if **equals(x)** returns true.
- **transitive**: for any objects x, y, and z, if **x.equals(y)** returns true and **y.equals(z)** returns true, then **x.equals(z)** should return true.

- **consistent**: for any objects x and y, the value of x.equals(y) should change, only if the property in equals() changes.
- For any object x, the *equals(null)* must return false.

Java hashCode()

- A **hashcode** is an integer value associated with every object in Java, facilitating the hashing in hash tables.
- To get this hashcode value for an object, we can use the hashCode() method in Java. It is the means *hashCode() method that returns the integer hashcode value of the given object*.
- Since this method is defined in the Object class, hence it is inherited by user-defined classes also.
- The hashCode() method returns the same hash value when called on two objects, which are equal according to the equals() method. And if the objects are unequal, it usually returns different hash values.

Syntax:

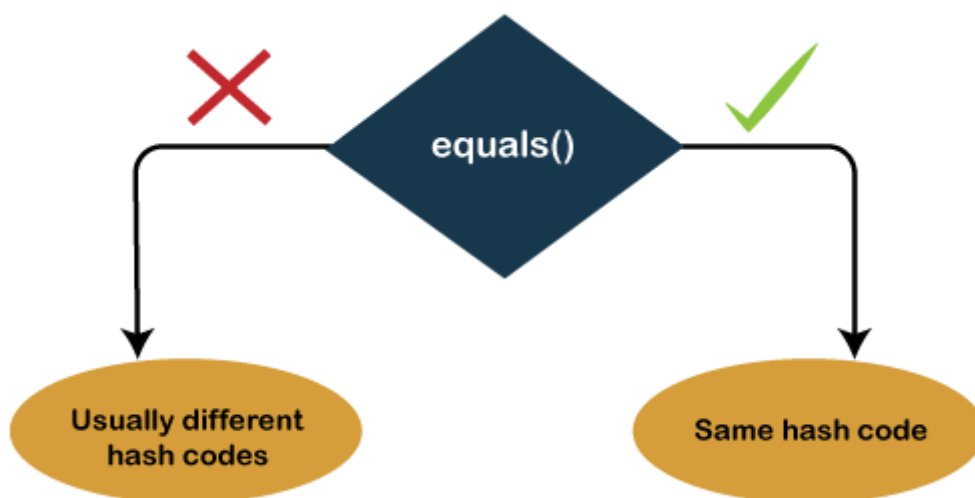
1. **public int** hashCode()

Returns:

It returns the hash code value for the given objects.

Contract for hashCode() method in Java

- If two objects are the same as per the equals(Object) method, then if we call the hashCode() method on each of the two objects, it must provide the same integer result.



Example:

1. **class** Test_hash_equal{
2. **public static void** main(String[] args){

```

3.     String a = "Andrew";
4.     String b = "Andrew";
5.
6.     if(a.equals(b)){ //checking the equality of objects using equals() methods
7.         System.out.println("a & b are equal variables, and their respective hashvalues are:" + " " + a.hashCode() + " & " + b.hashCode());
8.
9.     }
10.
11.    String c = "Maria";
12.    String d= "Julie";
13.
14.    if(!c.equals(d)){ //checking the equality of objects using equals() method
15.        System.out.println("\nc & d are Un-
        equal variables, and their respective hashvalues are:" + " " + c.hashCode() + " & " + d.hashCode());
16.
17.    }
18. }
19. }

```

Output:

```
a & b are equal variables, and their respective hash values are: 1965574029 & 1965574029
```

```
c & d are Un-equal variables, and their respective hash values are: 74113750 & 71933245
```

In the above example, we have taken two 4 variables, out of which two are equal, and two are unequal. First, we have compared the objects whether they are equal or unequal, and based on that, printed their hash values.

122). What is need of service in spring boot?

Answer -Service Components are the class file which contains @Service annotation. These class files are used to write business logic in a different layer, separated from @RestController class file. The logic for creating a service component class file is shown here –

```
public interface ProductService {
}
```

The class that implements the Interface with @Service annotation is as shown –

```
@Service
public class ProductServiceImpl implements ProductService {
}
```

we are using **Product Service API(s)** to store, retrieve, update and delete the products. We wrote the business logic in @RestController class file itself.

123). Concurrent HashMap vs HashMap vs Hashtable

Answer –

Parameter	HashMap	HashTable	All operations are synchronized.
Locking Level	Object Level	Object Level	Segment Level
Synchronized operations	All operations are synchronized.	All operations are synchronized.	Only update operations are synchronized.
How many threads can enter into a map at a time?	Only one thread	Only one thread	By default, 16 threads can perform update operations and any number of threads can perform read operations at a time.
Null Keys And Null Values	Allows one null key and any number of null values.	Doesn't allow null keys and null values.	Doesn't allow null keys and null values.
Nature Of Iterators	Fail-Fast	Fail-Safe	Fail-Safe
Introduced In?	JDK 1.2	JDK 1.1	JDK 1.5
When To Use?	Use only when high level of data consistency is required in multi threaded environment.	Don't Use. Not recommended as it is a legacy class.	Use in all multi threaded environment except where high level of data consistency is required.

124) Q) How to send mail in spring boot

Ans: we need to add the Spring Boot Starter Mail dependency in your build configuration file.

Maven users can add the following dependency into the pom.xml file.

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-mail</artifactId>
</dependency>
```

125) Q) what is microservices and it's Advantage

It is a kind of architectural style which structures an application as a collection of Services

Advantage :

- Improved Scalability
- Better Fault Isolation for More Resilient Applications.
- Programming Language and Technology Agnostic.
- Better Data Security and Compliance.
- Faster Time to Market and “Future-Proofing”.
- Greater Business Agility and Support for DevOps. ...

126)Q) how services communicate with each other in microservices

A microservices-based application is a distributed system running on multiple processes or services, usually even across multiple servers or hosts. Each service instance is typically a process. Therefore, services must interact using an inter-process communication protocol such as HTTP, AMQP, or a binary protocol like TCP, depending on the nature of each service.

127)Q) Difference between @Controller and @RestController

1. The @Controller is a common annotation which is used to mark a class as Spring MVC Controller while the @RestController is a special controller used in RESTFul web services and the equivalent of @Controller + @ResponseBody
2. The @RestController is relatively new, added only on Spring 4.0 but @Controller is an old annotation, exists since Spring started supporting annotation, and officially it was added on Spring 2.5 version.
3. The @Controller annotation indicates that the class is a “Controller” e.g. a web controller while the @RestController annotation indicates that the class is a controller where @RequestMapping methods assume @ResponseBody semantics by default i.e. servicing REST API.
4. The @Controller is a specialization of @Component annotation while @RestController is a specialization of @Controller annotation. It is actually a convenience controller annotated with @Controller and @ResponseBody

128) How HashSet internally Work

Each and every element in set is unique. So that there is no duplicate element in set.

Now what happen internally when you pass duplicate element in set then add () method of set object, it will return false and do not add to hashset as element is already present.

But main problem is arising that how it return false, here is the answer. When you open hashset implementation of add method () in Java API's you will find the following code-

```
Public class HashSet <E> extends AbstractSet<E> implements Set<E>, clonable  
java.o.serializable
```

```
{
```

```
Private transient HashMap<E, Object> Map;
```

```
/* dummy value associate with object in map */
```

```
Private static final Object Present= new Object();
```

```
Public HashSet(){
```

```
Map= new HashMap<>();
```

```
//some code, other method in hashset
```

```
}
```

```
Public Boolean add (E e) {
```

```
return map. Put(e, PRESENT)==null;
```

```
}
```

We are achieving uniqueness in set, internally java through hashmap. Whenever you create the object of hashset it will create the object of hashmap as seen in above.

As we know, in hashmap each key is unique. We do in set is that we pass argument in add(Element E) that is E as key in hashmap, now we need to associate some value to key, so what java developer did to pass dummy value that is (new Object()); which is referred by object reference PRESENT.

So actually when you are adding line in hashset like hashset.add(3) what java internally is that it will put that element as E here as 3 key in hashmap and some dummy value that object is passed as value to key.

If you see code of hashmap put(K k, value v) method, you will find something like this,

```
Public v put (K Key, V value){
```

```
//some code
```

```
}
```

The main point is that .put(key,value) will return

1. Null, if key is unique and added to map.
2. Old value of key, if key is duplicate.

So in Hashset add() method, we check return value of map.put(key,value) method will null value i.e.

```
Public Boolean add(E e){
```

```
// code here
```

```
}
```

So If Map.put(key,value) return null, then map.put(e,PRESENT)==null, then map.put(e,PRESENT)==null will return true & element added to hashset.

So If Map.put(key,value) return old value of key, then map.put(e,PRESENT)==null, then map.put(e,PRESENT)==null will return false & element is not added to hashset.

129)Difference betn Final

Finally Finalize

Final -

- 1) final is the keyword and access modifier which is used to apply restrictions on a class, method or variable.
- 2) Final keyword is used with the classes, methods and variables.
- 3) Once declared, final variable becomes constant and cannot be modified.
final method cannot be overridden by sub class.
final class cannot be inherited
- 4) Final method is executed only when we call it.

Finally

- 1) finally is the block in Java Exception Handling to execute the important code whether the exception occurs or not.
- 2)Finally block is always related to the try and catch block in exception handling.
- 3) finally block runs the important code even if exception occurs or not.
finally block cleans up all the resources used in try block
- 4) Finally block is executed as soon as the try-catch block is executed. It's execution is not dependant on the exception.

Finalize

- 1) finalize is the method in Java which is used to perform clean up processing just before object is garbage collected.
- 2) finalize() method is used with the objects.

- 3) finalize method performs the cleaning activities with respect to the object before its destruction.
- 4) finalize method is executed just before the object is destroyed.

130)Q) Try without catch is it possible?

Yes, It is possible to have a try block without a catch block by using a final block. As we know, a final block will always execute even there is an exception occurred in a try block, except System.exit();

131)Q) If try/catch blocks have a return statement, even then the finally block executes?

Flow control first jumps to the finally block and then goes back to the return statement.

132)Q) Write Join Query in MySQL ??

133)Q) Define primary key and foreign key

Primary Key - Primary key is a value, or a combination of few values from the table, uniquely defining each record in this table. If we know this value/combination, we can easily find the related record and access all remaining values from that record.

Foreign Key -A Foreign Key is a database key that is used to link two tables together. The FOREIGN KEY constraint identifies the relationships between the database tables by referencing a column, or set of columns, in the Child table that contains the foreign key, to the PRIMARY KEY column or set of columns, in the Parent table.

134)Q) How to handle exception in your project

--- Need to discuss after Project Done

135)Dependency Injection? and How it Works?

Dependency Injection (DI) is a design pattern used to implement IoC. It allows the creation of dependent objects outside of a class and provides those objects to a class through different ways. Using DI, we move the creation and binding of the dependent objects outside of the class that depends on them.

Constructor Based –

136) Constructor-based DI is when the container invokes a constructor with a number of arguments, each of which represents a dependency or other class.

137) Calling a static factory method with particular arguments to construct the bean is approximately equivalent, treating arguments to a constructor and to a static factory method.

Setter Based –

138) Setter-based DI is the when the container calls setter methods on your beans after it has invoked a no-argument constructor or no-argument static factory method to instantiate that bean.

136) Q) @springBootApplication Annotation

Ans - @SpringBootApplication=

@Configuration+@ComponentScan+@EnableAutoConfiguration

The @SpringBootApplication annotation is a combination of following three Spring annotations and provides the functionality of all three with just one line of code.

@Configuration

This annotation marks a class as a Configuration class for Java-based configuration. This is particularly important if you favor Java-based configuration over XML configuration.

@ComponentScan

This annotation enables component-scanning so that the web controller classes and other components you create will be automatically discovered and registered as beans in Spring's Application Context. All the @Controller classes you write are discovered by this annotation.

@EnableAutoConfiguration

This annotation enables the magical auto-configuration feature of Spring Boot, which can automatically configure a lot of stuff for you.

For example, if you are writing a Spring MVC application and you have Thymeleaf JAR files on the application classpath, then Spring Boot auto-configuration can automatically configure the Thymeleaf template resolver, view resolver, and other settings automatically.

137) Q) serialization and how to prevent it?

serialization is the conversion of a Java object into a static stream (sequence) of bytes which can then be saved to a database or transferred over a network

For Prevention we can use transient keyword

138) Q) Basic Question in Multithreading (Jeevan sir notes- more than enough)

139) Q) Spring and Spring Boot difference

Spring - 1) Spring Framework is a widely used Java EE framework for building applications.

- 2) It aims to simplify Java EE development that makes developers more productive
- 3) The primary feature of the Spring Framework is dependency injection
- 4) It helps to make things simpler by allowing us to develop loosely coupled applications.
- 5) To test the Spring project, we need to set up the server explicitly.

SpringBoot - 1) Spring Boot Framework is widely used to develop REST APIs.

- 2) It aims to shorten the code length and provide the easiest way to develop Web Applications.

- 3) The primary feature of Spring Boot is Autoconfiguration. It automatically configures the classes based on the requirement.
- 4) It helps to create a stand-alone application with less configuration.
- 5) Spring Boot offers embedded server such as Jetty and Tomcat, etc.

140) Step for design Rest API in Spring Boot

- 1) Create the Spring Boot Project.
- 2) Define Database configurations.
- 3) Create an Entity Class.
- 4) Create JPA Data Repository layer.
- 5) Create Rest Controllers and map API requests.
- 6) Create Unit Testing for API requests and run the unit testing.
- 7) Build and run the Project.

141) Q) What is immutable class

Immutable class in java means that once an object is created, we cannot change its content. In Java, all the wrapper classes (like Integer, Boolean, Byte, Short) and String class is immutable

142) Q) How to create Immutable class in Java?

- The class must be declared as final so that child classes can't be created.
- Data members in the class must be declared private so that direct access is not allowed.
- Data members in the class must be declared as final so that we can't change the value of it after object creation.
- A parameterized constructor should initialize all the fields performing a deep copy so that data members can't be modified with an object reference.

143) Q) What is an index in a database?

A database index is a data structure that improves the speed of data retrieval operations on a database table at the cost of additional writes and storage space to maintain the index data structure. An index is a copy of selected columns of data, from a table, that is designed to enable very efficient search.

144) Q) What is NoSQL database example?

MongoDB, CouchDB, CouchBase, Cassandra, HBase, Redis, Riak, Neo4J are the popular NoSQL databases examples.

145) Q) What is Bean and Scope of Bean?

A bean is an object that is instantiated, assembled, and otherwise managed by a Spring IoC container. These beans are created with the configuration metadata that you supply to the container.

Scopes

- 1) Singleton - Scopes a single bean definition to a single object instance per Spring IoC container.
- 2) Prototype - a single bean definition to any number of object instances.
- 3) Request - a single bean definition to the lifecycle of a single HTTP request; that is each and every HTTP request will have its own instance of a bean created off the back of a single bean definition. Only valid in the context of a web-aware Spring ApplicationContext.
- 4) Session - Scopes a single bean definition to the lifecycle of a HTTP Session. Only valid in the context of a web-aware Spring ApplicationContext.
- 5) Global Session - single bean definition to the lifecycle of a global HTTP Session. Typically only valid when used in a portlet context. Only valid in the context of a web-aware Spring ApplicationContext.

146) What is final Class?

Answer –

A class that declares with final keyword is known as final class. A final class cannot be inherited by any Sub-class. We can make class as final only when the class is complete in nature i.e. it should not be an abstract class. All the wrapper classes present in java are finalclasses like String, Integer etc.

147) Difference between String and String Buffer?

Answer –

No.	String	StringBuffer
1>	String class is immutable.	StringBuffer is mutable.
2>	String is slow and consumes more memory when we concat too many strings because every time it creates new instance.	StringBuffer is fast and consumes less memory when you concat strings.
3>	String class overrides equals() method of Object class. So we can compare the contents of two strings by equals() method.	StringBuffer class doesn't override the equals() method of Object class.

148) Can we catch error in try and catch block?

Answer –

Errors cannot be catch in try and catch block because errors occurred due to lack of resources or system issues. In try and catch block we can handle different types of exceptions only.

149) HashSet internal working?

Answer –

Each and every element in set is unique. Set doesn't allow duplicate element. Below is the example of that how to add element in set.

```
import java.util.HashSet;
```

```

import java.util.Set;

public class HashSetDemo {

    public static void main(String[] args) {Set hashSet =

        new HashSet();

        hashSet.add(5);

        hashSet.add("Pune");

        hashSet.add("India");

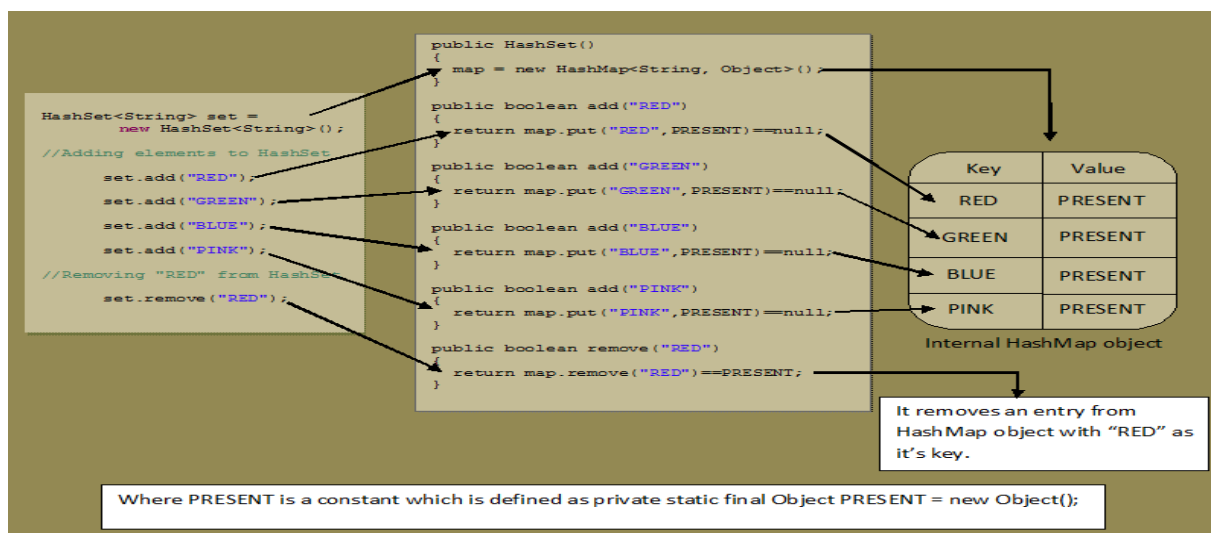
        hashSet.add(5);

        System.out.println(hashSet);

    }
}

```

Output - [5, Pune, India]



When we pass the duplicate element in Set then add method of set will return false and doesn't allow HashSet to add as it is already present. Internally add () method of HashSet implements the below code –

```

Public class HashSet <E> extends AbstractSet<E> implements Set<E>, Clonable, Serializable
{
    Private transient Hashmap<E, Object> Map;

    /* dummy value associate with object in map */

```

```

    Private static final object Present= new
    object ();Public HashSet () {
Map= new Hashmap<> ();
    //some code, other method in hashset
}

    Public Boolean add (E e) {
return map. Put (e, PRESENT) ==null;
    }

```

We are achieving uniqueness in set, internally java through HashMap. Whenever you create the object of HashSet it will create the object of HashMap.

In HashMap each key is unique. In set we pass argument in add (Element E) that is E as key in HashMap, also we need to pass some value to key, so internally it pass dummy value that is (new Object()); which is referred by object reference **PRESENT**.

When we adding values in HashSet like hashset.add(5) internally it will put that element as E here as 5 key in HashMap and dummy value so that object is passed as value to key.

Internally HashMap have put (K k, value v) method.

```

    Public v put (K Key, V value) {
//some code
    }

```

Internally .put(Key, Value) will return

1. Null, if key is unique and added to map.
2. Old value of key, if key is duplicate.

So in HashSet add () method, we check return value of map.put(Key, Value) method will return null value i.e.

```

    Public Boolean add (E e) {
// code here
    }

```

If map.put(Key, Value) return null then **map.put(e,PRESENT)==null will return true & element added to HashSet.**

If map.put(Key, Value) return old value of key, then **map.put(e,PRESENT)==null will return false & element is not added to HashSet.**

150) Difference between ArrayList

and LinkedList? Answer –

No.1	ArrayList	LinkedList
1>	ArrayList internally uses Dynamic Array structure.	LinkedList internally uses Doubly Linked list structure.
2>	ArrayList is best suitable for store and retrieval of data.	LinkedList is best suitable for manipulation of data.
3>	ArrayList provides random access.	LinkedList doesn't provide random access.
4>	ArrayList takes less memory as it stores only objects.	LinkedList takes more memory as it stores object as well as address of that object.

151) Advantages of

Multithreading? Answer –

Following are the advantages of Multithreading –

1. It doesn't block the user as threads are independent and can perform multiple operation at same time.
2. As multiple operation can be performed at same time which results in it saves time.
3. If exception occurred in one thread it doesn't affect the others as it is independent in nature.
4. A thread is lightweight.
5. Threads share same address space.
6. Cost of communication between thread is very low.

152) What is

ConcurrentModificationException?

Answer –

When one thread iterating the collection and if other thread tried modify the collection object (Adding or Removing Object) then in such cases immediately iterator throws a exception called as **ConcurrentModificationException**.

And the iterator which throws this type of exception as soon as they encounter such situation is called **fail-fast iterator**.

153) What is @Repository annotation in

spring boot? Answer –

This annotation is used on classes which are working directly with DB (Data Base) layer. This annotation shows that the given class is repository. The main use of using this annotation is to catch the platform specific Exception and re-throw as one of the Spring's undefined unchecked exceptions.

154) What are the advantages of Spring

framework? Answer –

1. Predefined template-

It provides templates for hibernate, JPA, JDBC, etc. there is no need to write too much code.

Example- in JDBC template, we don't need to write code for exception handling, creating connection, creating statement, closing connection. Only we need to write the code for executing the query.

2. Loose coupling-

Spring applications are loosely coupled because of dependency injection.

3. Easy to test-

Spring does not require server to run the program. It only needs JDK and Jar file.

4. Light weight-

Spring is light weight component due to its POJO implementation. It does not force any programmer to inherit the class or implement any interface.

5. Development is very fast.

155) How did you handle the exception in

controller of Rest API? Answer –

This topic will be covered in final project.

156) What is SQL injection?

Answer –

SQL injection is a code of injection technique that might destroy your database. It is one of the most common web hacking techniques. It is the placement of malicious code in SQL statements, via web page input. It is the most common type of injection attack.

157) What is request and response in Join?

Answer –

This topic will be covered in final project.

158) Difference between query and path parameters?

Answer –

No.	Query Parameter	Path Parameter
1>	It is not part of URL and passing key value format, those parameter must be define by API Developer.	Path parameter are variable parts of URI path.
2>	It is most common parameter they appear at the end of request URL after a ? with different value pairs separated by & operator.	They are typically used to point to a specific resource within collection such as user identified ID.
3>	Query parameter required and optional.	URL can have several path parameters each denoted with { }.
4>	If there is a scenario where you need to get the details of all employees but only 10 at a time, you may use query param GET /employee?start=1&size=10	If there is a scenario to retrieve a record based on id, for example you need to get the details of the employee whose id is 15, then you can have resource with @PathParam.GET /employee/{id}
5>	Use @PathParam for retrieval based on id.	Use @QueryParam for filter

159) What is 500, 401 HTTP status error code?

Answer –

500 – Internal Server Error

The server has encountered a situation that it doesn't know how to handle it.

401 – Unauthorized

The Client must be authenticated itself to get the requested response.

160) Difference between DDL and DML?

Answer –

No.	DDL	DML
1>	It stands for Data Definition Language.	It stands for Data Manipulation Language.
2>	It is used for create database schema and also used to define some constraints as well.	It is used to retrieve, add or update the data.
3>	It basically defines the column (Attributes) of the table.	It adds or update the row of the table. This rows are called as tuple.
4>	It doesn't have any further classification.	It is further classified into Procedural and Non-Procedural DML.
5>	Basic commands present in DDL are CREATE, RENAME, DROP, ALTER etc	Basic commands present in DML are UPDATE, MERGE, INSERT etc
6>	DDL does not uses where clause in its statement.	DML uses where clause in its statement.

161) What is GIT?

Answer –

Git is a DevOps tool used for source code management. It is a free and open-source version control system used to handle small to very large projects efficiently. Git is used to tracking changes in the source code, enabling multiple developers to work together on non-linear development.

162) Difference between @SpringBootApplication and @EnableAutoConfiguration?

Answer –

No.	SpringBootApplication	EnableAutoConfiguration
1>	It is used in main class or bootstrap class.	It is used to enable auto-configuration and component scanning in your project.

2>	It is a combination of @Configuration, @ComponentScan and @EnableAutoConfiguration annotations.	It is a combination of @ComponentScan and @Configuration annotations.
3>	It is not mandatory to used this annotation.	It is mandatory to used this annotation.

When to use

@EnableAutoConfiguration?

Answer –

Whenever there is requirement of selectively exclude certain classes from auto- configuration then we go for this annotation. For that we have to use exclude attribute with above annotation and specify the fully qualified class name.

163) Write a SQL query for two tables Employee and Department for below scenario –

Find the name of Employees working in Finance Department and lives in Mumbai City.

Answer –

If the city is in Department table, then below query will work –

```
select emp. empname from employee emp where emp.dept_id IN (select  
dep.dept_id from department dept where dept.city = "Mumbai");
```

If the city is in Employee table, then below query will work –

```
select emp. empname from employee emp where emp.dept_id IN (select  
dep.dept_id from department dept where deptname = "Finance") and emp.city  
="Mumbai";
```

164) What are the advantages of Spring Boot over Spring MVC framework?

Answer –

Following are the advantages of Spring Boot over Spring MVC framework –

1. It is mainly used to develop REST API.
2. It required less coding and provides easiest way for developing Web Application.
3. There is no need to build configuration manually.
4. There is no requirement of deployment descriptor.
5. It reduces development time and increases the productivity.

165) What are the annotations used in Spring

Boot framework?

@EnableAutoConfiguration –

It auto-configure the bean that is present in class path and configure it to run method.

@SpringBootApplication –

It is the configuration of 3 annotations i.e. @EnableAutoConfiguration, @ComponentScan and @Configuration. It is not mandatory to used.

167) What is starter in Spring Boot

framework?

Spring Boot provides a number of starters that allow us to add jars in the classpath. Spring Boot built-in starters make development easier and rapid. Spring Boot Starters are the dependency descriptors.

168) What is @RestController and

@RequestMapping?

@RestController –

It is the special annotation used in RESTful web services. It is a combination of 2 annotations i.e. @Controller and @ResponseBody. @RestController annotation is itself annotated with the @ResponseBody annotation. It eliminates the need for annotating each method with @ResponseBody.

@RequestMapping –

It is used to map the web request. It can be used with class as well as method. It has many optional elements like consumes, header, method, name, params, path, produces, and value.

169) How to create simple project in

Spring Boot?

Following are the steps needs to follow to create Simple project in spring boot –

1. Click on New Spring Starter Project
 2. Enter Artifact and Group Name
-

3. Select the dependencies otherwise click on finish button.

170) What are the properties used in Spring Boot and where it to be mentioned?

The following tables provide a list of common Spring Boot properties -

Property	Default Value	Description
Debug	FALSE	It enables debug logs.
spring.application.name		It is used to set the application name.
spring.application.admin.enabled	FALSE	It is used to enable admin features of the application.
spring.config.name	application	It is used to set config file name.
spring.config.location		It is used to config the file name.
server.port	8080	Configures the HTTP server port
server.servlet.context-path		It configures the context path of the application.
logging.file.path		It configures the location of the log file.
spring.banner.charset	UTF-8	Banner file encoding.
spring.banner.location	classpath:banner.txt	It is used to set banner file location.
logging.file		It is used to set log file name. For example, data.log.
spring.application.index		It is used to set application index.
spring.application.name		It is used to set the application name.
spring.application.admin.enabled	FALSE	It is used to enable admin features for the application.
spring.config.location		It is used to config the file locations.

spring.config.name	application	It is used to set config the file name.
spring.mail.default-encoding	UTF-8	It is used to set default MimeMessage encoding.
spring.mail.host		It is used to set SMTP server host. Forexample, smtp.example.com.
spring.mail.password		It is used to set login password of the SMTP server.
spring.mail.port		It is used to set SMTP server port.
spring.mail.test-connection	FALSE	It is used to test that the mail server is available on startup.
spring.mail.username		It is used to set login user of the SMTPserver.
spring.main.sources		It is used to set sources for the application.
server.address		It is used to setnetwork address to which the server should bind to.
server.connection-timeout		It is used to set time in milliseconds that connectors will wait for another HTTP request before closing the connection.
server.context-path		It is used to set context path of the application.
server.port	8080	It is used to set HTTP port.
server.server-header		It is used for the Server response header (no header is sent if empty)

Spring Boot provides various properties that can be configured in the **application.properties** file.

Question 26 – **Which design pattern used in project and which are the dependency injection used?** Answer –

This topic will be covered in final project.

171)What is spring boot actuator?

Spring Boot Actuator is a sub-project of the Spring Boot Framework. It includes a number of additional features that help us to monitor and manage the Spring Boot application. It contains the actuator endpoints (the place where the resources live). We can use **HTTP** and **JMX** endpoints to manage and monitor the Spring Boot application. If we want to get production-ready features in an application, we should use the Spring Boot actuator.

There are three main features of Spring Boot Actuator:

- **Endpoints**
- **Metrics**
- **Audit**

Endpoint: The actuator endpoints allow us to monitor and interact with the application. Spring Boot provides a number of built-in endpoints. We can also create our own endpoint. We can enable and disable each endpoint individually. Most of the application choose **HTTP**, where the Id of the endpoint, along with the prefix of **/actuator**, is mapped to a URL.

For example, the **/health** endpoint provides the basic health information of an application. The actuator, by default, mapped it to **/actuator/health**.

Metrics: Spring Boot Actuator provides dimensional metrics by integrating with the **micrometre**. The micrometre is integrated into Spring Boot. It is the instrumentation library powering the delivery of application metrics from Spring. It provides vendor-neutral interfaces for **timers, gauges, counters, distribution summaries, and long task timers** with a dimensional data model.

Audit: Spring Boot provides a flexible audit framework that publishes events to an **AuditEventRepository**. It automatically publishes the authentication events if spring-security is in execution.

172)What is difference between JDBC and hibernate

Sr. No.	Key	JDBC	Hibernate
1	Basic	It is database connectivity technology	It is a framework,
2	Lazy Loading	It does not support lazy loading	Hibernate support lazy loading
3	Transaction management	We need to maintain explicitly database connection and transaction.	Hibernate itself manage all transaction
4.	Caching	We need to write code for implementing caching	<p>Hibernate provides two types of caching :</p> <p>First level Cache</p> <p>Second level cache</p> <p>No Extra code is required to use first level cache.</p>
5.	Performance	Low performance	High Performance

173) How you use database connectivity using hibernate

use Hibernate-provided JDBC connections, the configuration file requires the following five properties:

- connection.driver_class -The JDBC connection class for the specific database
- connection.url -The full JDBC URL to the database
- connection.Username -The username used to connect to the database
- connection.Password -The password used to authenticate the username
- dialect -The name of the SQL dialect for the database

174) What is Inheritance association

Inheritance: One class can use features from another class to extend its functionality. Inheritance based on IS-A Relationship. Inheritance is uni-directional. Inheritance is indicated by a solid line with an arrowhead pointing at the super class. Example: 1. House is a Building. But Building is not a House. 2. A Car is a Automobile. 3. A Cat is a Animal. Here Animal class is the super class / Base class for the Cat & Dog Classes, which are derived classes.

Association: Association represents a relationship between two or more objects where all objects have their own life cycle and there is no owner. Association is based on HAS-A Relationship. This is represented by a solid line. We take an example of relationship between Teacher and Student. Many students can have one teacher and one student can have many teachers. But there is no ownership between the objects and both have their own lifecycle. Both can be created and deleted independently.

175) Difference between 1st n 2nd cache in hibernate

Sr. No.	Key	First level cache	Second level cache
1	Basic	First level cache is a session level cache and it is always associated with session level object	Second level cache is session factory level cache and it is available across all sessions
2	Enabled	It is enabled by default.	It is not enabled by default.
3	Availability	It is available for a session	It is available across all session.
4	Configuration	No Extra configuration required	We have to decide which concurrency strategy to use and also need to configure cache expiration and physical cache attributes.

176) How many DB available in dialects?

Ans- List of SQL Dialects

There are many Dialects classes defined for RDBMS in the **org.hibernate.dialect** package. They are as follows:

RDBMS	Dialect
Oracle (any version)	org.hibernate.dialect.OracleDialect
Oracle9i	org.hibernate.dialect.Oracle9iDialect
Oracle10g	org.hibernate.dialect.Oracle10gDialect
MySQL	org.hibernate.dialect.MySQLDialect
MySQL with InnoDB	org.hibernate.dialect.MySQLInnoDBDialect
MySQL with MyISAM	org.hibernate.dialect.MySQLMyISAMDialect
DB2	org.hibernate.dialect.DB2Dialect
DB2 AS/400	org.hibernate.dialect.DB2400Dialect
DB2 OS390	org.hibernate.dialect.DB2390Dialect
Microsoft SQL Server	org.hibernate.dialect.SQLServerDialect
Sybase	org.hibernate.dialect.SybaseDialect
Sybase Anywhere	org.hibernate.dialect.SybaseAnywhereDialect
PostgreSQL	org.hibernate.dialect.PostgreSQLDialect
SAP DB	org.hibernate.dialect.SAPDBDialect
Informix	org.hibernate.dialect.InformixDialect
HypersonicSQL	org.hibernate.dialect.HSQLDialect
Ingres	org.hibernate.dialect.IngresDialect
Progress	org.hibernate.dialect.ProgressDialect
Mckoi SQL	org.hibernate.dialect.MckoiDialect
Interbase	org.hibernate.dialect.InterbaseDialect
Pointbase	org.hibernate.dialect.PointbaseDialect
FrontBase	org.hibernate.dialect.FrontbaseDialect
Firebird	org.hibernate.dialect.FirebirdDialect

177) Difference between save and persist, get and load, save and saveOrUpdate?

Sr. No.	Key	save()	persist()
1	Basic	It stores object in database	It also stores object in database
2	Return Type	It return generated id and return type is serializable	It does not return anything. Its void return type.
3	Transaction Boundaries	It can save object within boundaries and outside boundaries	It can only save object within the transaction boundaries
4.	Detached Object	It will create a new row in the table for detached object	It will throw persistence exception for detached object
5.	Supported by	It is only supported by Hibernate	It is also supported by JPA

Sr. No.	Key	Get()	Load()
1	Basic	It is used to fetch data from the database for the given identifier	It is also used to fetch data from the database for the given identifier
2	Null Object	It object not found for the given identifier then it will return null object	It will throw object not found exception

Sr. No.	Key	Get()	Load()
3	Lazy or Eager loading	It returns fully initialized object so this method eager load the object	It always returns proxy object this method is lazy load object
4	Performance	It is slower than load() because it return fully initialized object which impact the performance of the application	It is slightly faster.
5.	Use Case	If you are not sure that object exist then use get() method	If you are sure that object exists then use load() method

178) What is session factory and session

SessionFactory: The SessionFactory is a factory of session and client of ConnectionProvider. It holds second level cache (optional) of data. The org.hibernate.SessionFactory interface provides factory method to get the object of Session.

Session :The session object provides an interface between the application and data stored in the database. It is a short-lived object and wraps the JDBC connection. It is factory of Transaction, Query and Criteria. It holds a first-level cache (mandatory) of data. The org.hibernate.Session interface provides methods to insert, update and delete the object. It also provides factory methods for Transaction, Query and Criteria.

179) What is dirty checking in hibernate

Dirty checking is an essential concept of Hibernate. The Dirty checking concept is used to keep track of the objects. It automatically detects whether an object is modified (or not) or wants to be updated.

It also allows a developer to avoid time-consuming database write actions. It modifies only those fields which require modifications, and the remaining fields are kept unchanged.

Example of Dirty Checking

In this example, we are taking an entity class **Student**. The Student class contains student id(id), name (Sname), course (Scourse), and rollno (Srno) of the student. It also provides default and a parameterized constructor.

To access dirty checking in the application, we are using the annotation **@DynamicUpdate** to the entity class **Student**.

@DynamicUpdate- It is used for updating the objects. This annotation makes the necessary modifications and changes to the required fields.

180) write custom exception

We can create our own Exception that is known as custom exception or user-defined exception. By the help of custom exception, you can have your own exception and message.

Example-1- Scenario

Sometimes it is required to develop meaningful exceptions based on application requirements. For example suppose you have one savings account in SBI Bank and you have 50000 in your account. Suppose you attempted to withdraw 60000 from your account. In java you can handle. You need to display some error message related to insufficient fund.

Steps to create the user defined exception

1. Create the new class.
2. The user defined exception class must extend from java.lang.Exception or java.lang.RuntimeException class.
3. While creating custom exception, prefer to create an unchecked, Runtime exception than a checked exception.
4. Every user defined exception class in which parametrized Constructor must call parametrized Constructor of either java.lang.Exception or java.lang.RuntimeException class by using super(string parameter always).

181) write code for method overloading

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

```
1. public
class Adder{
static int
add(int a,int
b){return
a+b;
```



```

    }
    static int add(int a,int b,int c){
    return a+b+c;
    }
}

```

2. public class

```

TestOverloading1{ public
static void main(String[]
args){
System.out.println(Adder.add(
11,11));
System.out.println(Adder.add(
11,11,11));
}
}

```

182) String str= java is Object oriented programming language

Yes, In Java, string is basically an object that represents sequence of char values. An array of characters works same as Java string.

183) Write code to remove extra space

```

public class RemoveAllSpace {
    public static void main(String[]
        args) { String str = "India
        Is My Country";

        //1st way
        String noSpaceStr = str.replaceAll("\\s", ""); // using built in
        methodSystem.out.println(noSpaceStr);
        //2nd way
        char[] strArray = str.toCharArray();
        StringBuffer stringBuffer = new
        StringBuffer();for (int i = 0; i <
        strArray.length; i++) {
            if ((strArray[i] != ' ') &&
                (strArray[i] != '\t')) {
                stringBuffer.append(strArray[i
                ]);
            }
        }
        String noSpaceStr2 =
        stringBuffer.toString();
        System.out.println(noSpaceStr2);
    }
}

```

```

        }
    }
}

```

Output

```

IndiaIsM
yCountr
y
IndiaIsM
yCountr
y

```

184)What is immutable in java?

Immutable objects are instances whose state doesn't change after it has been initialized. Forexample, **String** is an immutable class and once instantiated its value never changes.

185)How we can create immutable classes?

To create an immutable class in Java, you have to do the following steps.

- 186) Declare the class as final so it can't be extended.
- 187) Make all fields private so that direct access is not allowed.
- 188) Don't provide setter methods for variables.
- 189) Make all **mutable fields final** so that its value can be assigned only once.
- 190) Initialize all the fields via a [constructor](#) performing deep copy.
- 191) Perform [cloning](#) of objects in the getter methods to return a copy rather than returning the actual object reference.

186)Oops concepts with an example.

There are 4 oops pillars

1. Abstraction : It is the process of hiding the certain details and showing the important information to the end user called as "Abstraction".

Example- Real life scenario is car.

2. encapsulation : Binding (or wrapping) code and data together into a single unit are known as encapsulation. For example, a capsule, it is wrapped with different medicines. 2. Class is the entity which contains variables and methods.
3. Inheritance: The process of creating the new class by using the existing class functionality called as Inheritance.

Inheritance means simply

reusability. It is called as -

(IS Relationship)

4. Polymorphism : One entity that behaves differently in different cases called as polymorphism. Example- Light button, we are using that button to on or off the lights.

187)What are types of Exception handling?

Java defines several types of exceptions that relate to its various class libraries. Java also allows users to define their own exceptions.

188) ArithmeticException

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

189) 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.

190) ClassNotFoundException

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

191) FileNotFoundException

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

192) IOException

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

193) InterruptedException

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

194) NoSuchFieldException

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

195) NoSuchMethodException

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

196) NullPointerException

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

197) NumberFormatException

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

198) RuntimeException

This represents any exception which occurs during runtime.

199) StringIndexOutOfBoundsException

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

188) What are serialization and deserialization?

Serialization is a mechanism of converting the state of an object into a byte stream. Deserialization is the reverse process where the byte stream is used to recreate the actual Java object in memory. This mechanism is used to persist the object.

The byte stream created is platform independent. So, the object serialized on one platform can be deserialized on a different platform.

To make a Java object serializable we implement the **java.io.Serializable** interface. The **ObjectOutputStream** class contains **writeObject()** method for serializing an Object. Ex. `public final void writeObject(Object obj)`
throws **IOException**

The **ObjectInputStream** class contains **readObject()** method for deserializing an object. Ex. `public final Object readObject()`

throws **IOException**,

ClassNotFoundException

on

189)How to make a copy of objects?

The class whose object's copy is to be made must have a public clone method in it or in one of its parent class.

- Every class that implements clone() should call super.clone() to obtain the cloned object reference.
- The class must also implement java.lang.Cloneable interface whose object clone we want to create otherwise it will throw CloneNotSupportedException when clone method is called on that class's object.

Syntax: protected Object clone() throws CloneNotSupportedException

190)What's are JPA and Hibernate? Tell me the difference.

JPA : A JPA (Java Persistence API) is a specification of Java which is used to access, manage, and persist data between Java object and relational database. It is considered as a standard approach for Object Relational Mapping. JPA can be seen as a bridge between object-oriented domain models and relational database systems. Being a specification, JPA doesn't perform any operation by itself. Thus, it requires implementation. So, ORM tools like Hibernate, TopLink, and iBatis implements JPA specifications for data persistence.

JPA	Hibernate
Java Persistence API (JPA) defines the management of relational data in the Java applications.	Hibernate is an Object-Relational Mapping (ORM) tool which is used to save the state of Java object into the database.
It is just a specification. Various ORM tools implement it for data persistence.	It is one of the most frequently used JPA implementation.
It is defined in javax.persistence package.	It is defined in org.hibernate package.
The EntityManagerFactory interface is used to interact with the entity manager factory for the persistence unit. Thus, it provides an entity manager.	It uses SessionFactory interface to create Session instances.
It uses EntityManager interface to create, read, and delete operations for instances of mapped entity classes. This interface interacts with the persistence context.	It uses Session interface to create, read, and delete operations for instances of mapped entity classes. It behaves as a runtime interface between a Java application and Hibernate.
It uses Java Persistence Query Language (JPQL) as an object-oriented query language to perform database operations.	It uses Hibernate Query Language (HQL) as an object-oriented query language to perform database operations.

Hibernate : A Hibernate is a Java framework which is used to store the Java objects in the relational database system. It is an open-source, lightweight, ORM (Object Relational Mapping) tool. Hibernate is an implementation of JPA. So, it follows the common standards provided by the JPA.

191) What is Singleton class how can we create a singleton class?

In object-oriented programming, a singleton class is a class that can have only one object (an instance of the class) at a time.

After first time, if we try to instantiate the Singleton class, the new variable also points to the first instance created.

Remember the key points while defining class as singleton class that is while designing a singleton class:

192) Make constructor as private.

193) Write a static method that has return type object of this singleton class. Here, the concept of [Lazy initialization](#) is used to write this static method.

Let us brief how singleton class varies from normal class in java. Here the difference is in terms of instantiation as for normal class we use constructor, whereas for singleton class we use [getInstance\(\) method](#)

192)Find the Second highest number from the array.{100,45,52,67,122,125}

```
public class SecondLargestInArrayExample{
public static int getSecondLargest(int[] a,
int total){ int temp;
for (int i = 0; i < total; i++)
    {
        for (int j = i + 1; j < total; j++)
        {
            if (a[i] > a[j])
            {
                temp= a[i];

                a[i]=a[j];

                a[j]=temp;

            }

        }

    }

return a[total-2];

}

public static void
main(String args[]){ int
a[]={ 100,45,52,67,122,12
5};
System.out.println("Second Largest:"+getSecondLargest(a,6));
}}
```

193)Find the second-highest number when all are negatives

```
public class SecondLargestInArrayExample{
public static int getSecondLargest(int[] a, int total){ int temp;
for (int i = 0; i < total; i++)
{
for (int j = i + 1; j < total; j++)
{
if (a[i] > a[j])
{
temp = a[i];
```

```
a[i] = a[j]; a[j] = temp;
}
}
}
return a[total-2];
}
public static void main(String args[]){ int a[]={-1,-2,-5,-6,-3,-2};
    System.out.println("Second Largest:"+getSecondLargest(a,6));
```


194)Hibernate annotations used?

Ans: Hibernate annotations are the newest way to define mappings without the use of XML file. You can use annotations in addition to or as a replacement of XML mapping metadata. Hibernate annotations is the powerful way to provide the metadata for the object and relational table mapping.

194)Hibernate Session and SessionFactory ?

Ans: Session:-

- a. It is one instance per client/one transaction.
- b. It is not thread safe.
- c. It is light weight.
- d. Session will be opened using sessionFactory.openSession() and some database operations will be done finally session will be closed using session.close().

SessionFactory:-

- a. It is one interface per database.
- b. it is thread safe.
- c. It is heavy weight component.
- d. It will create and manage the sessions.
- e. It is an immutable object and it will be created as singleton while the server initializes itself.

3. How can you use second level cache?

Ans: It uses a common cache for all the session factory. It is useful if you have multiple session objects from a session factory.

There are four ways to use second level cache:

- a. Read-only
- b. Nonstrict-read-write
- c. Read-write
- d. Transactional
- e.

195)Can a constructor be private ?

Ans: Yes. Class can have private constructor. Even abstract class can have private constructor. By making constructor private, we prevent class from being instantiated as well as subclassing of that class.

196)What is Singleton how we create singleton classes ?

Ans: It means defines the class which has single instance that provide the global point of access to it called as singleton pattern.

Steps to create singleton classes:-

- a. Create class singleton class and static member of class.
- b. Make constructor as private.
- c. Create the method for checking the references and use synchronized block instead of method.

197)What are the ways to break it? And how we can stop it?

Ans: There are two ways to break it:

- a. By using serialization and deserialization
- b. By using Clone() method.

198) Why throws declared with the method signature ?

Ans: Checked exceptions are always declared as thrown in the method signature. The Signature lets the method's caller know that an exception may occur as a consequence of the call. If the exception does get thrown, the caller method be prepared to do something about it.

199) What is method overloading and various scenarios of it ?

Ans: It is the same method name with different parameters called as method overloading. It is also called as early binding compile time polymorphism or static binding.

Rules:- a. method name must be same.

b. parameters must be different.

c. return type is anything.

d. access specifier is anything.

e. exception thrown is anything.

Scenarios:

```
Public class Test{  
    Void add(int a, int b){  
        Sysout(a+b);}
```

```
    Void add(double a, double b){  
        Sysout(a+b);}
```

```
    Void add(float a){  
        Sysout(a);}  
}
```

200) What is @Qualifier ?

Ans: It is used to resolve the autowiring conflict, when there are multiple beans of same type. It can be used on any class annotated with @Component or on methods annotated with @Bean. This annotation can also be applied on constructor arguments or method parameters.

201) One question on related to the interface in hibernate ?

Ans: Interfaces in hibernate:

- a. Session interface
- b. SessionFactory interface
- c. Configuration interface
- d. Transaction interface
- e. Query and criteria interface

202) Various scopes of beans ?

Ans: a. singleton
b. prototype
c. request
d. session
e. global-session

203) Comparable and comparator difference ?

Ans:

Comparable	Comparator
------------	------------

Comparable provides compareTo() method to sort elements in Java.	Comparator provides compare() method to sort elements in Java.
Comparable interface is present in java.lang package.	Comparator interface is present in java.util package.
The logic of sorting must be in the same class whose object you are going to sort.	The logic of sorting should be in separate class to write different sorting based on different attributes of objects.
The class whose objects you want to sort must implement comparable interface.	Class, whose objects you want to sort, do not need to implement a comparator interface.
It provides single sorting sequences.	It provides multiple sorting sequences.
This method can sort the data according to the natural sorting order.	This method sorts the data according to the customized sorting order.
It affects the original class. i.e., actual class is altered.	It doesn't affect the original class, i.e., actual class is not altered.
Implemented frequently in the API by: Calendar, Wrapper classes, Date, and String.	It is implemented to sort instances of third-party classes.
All wrapper classes and String class implement comparable interface.	The only implemented classes of Comparator are Collator and RuleBasedCollator.

204)What is Has-A relationship ?

Ans: In Java, a Has-A relationship is also known as **composition**. It is also used for code reusability in Java. In Java, a Has-A relationship simply means that an instance of one class has a reference to an instance of another class or an other instance of the same class. For example, a car has an engine, a dog has a tail and so on.

205)What is ClassNotFoundException and NoClassDefFound ?

Ans:

ClassNotFoundException is an exception that occurs when you try to load a class at run time using **Class.forName()** or **loadClass()** methods and mentioned classes are not found in the classpath.

NoClassDefFoundError is an error that occurs when a particular class is present at compile time, but was missing at run time.

206)Why String used in HashMap ?

Ans: **String** is as a key of the **HashMap** When you create a **HashMap** object and try to store a key-value pair in it, while storing, a hash code of the given key is calculated and its value is placed at the position represented by the resultant hash code of the key.

207)HashMap inner implementation ?

Ans: It internally uses equals and hashCode method. When we override equals method then its compulsory to implement or override hashCode method also. If equals method returns true for two objects then hashCode method should return same hashCode for that two objects. It uses bucket for storing the hashmap. Defines index by using formula.

208)What is hibernate.cfg file contains ?

Ans: One of the most required configuration file in Hibernate is hibernate.cfg.xml file. By default, it is placed under src/main/resource folder. hibernate.cfg.xml file contains **database-related configurations and session-related configurations**.

209)What is association ?

Ans: Association refers to the relationship between multiple objects. It refers to how objects are related to each other and how they are using each other's functionality. Composition and aggregation are two types of association.

210).Difference between abstract class and interface?

Abstract class	Interface
1) Abstract class can have abstract and non-abstract methods.	Interface can have only abstract methods. Since Java 8, it can have default and static methods also.
2) Abstract class doesn't support multiple inheritance .	Interface supports multiple inheritance .
3) Abstract class can have final, non-final, static and non-static variables .	Interface has only static and final variables .
4) Abstract class can provide the implementation of interface .	Interface can't provide the implementation of abstract class .
5) The abstract keyword is used to declare abstract class.	The interface keyword is used to declare interface.
6) An abstract class can extend another Java class and implement multiple Java interfaces.	An interface can extend another Java interface only.
7) An abstract class can be extended using keyword "extends".	An interface can be implemented using keyword "implements".
8) A Java abstract class can have class members like private, protected, etc.	Members of a Java interface are public by default.
9)Example: public abstract class Shape{ public abstract void draw(); }	Example: public interface Drawable{ void draw(); }

211)

212)

213)

214)

215)

216)

217)

211)Collection and collections

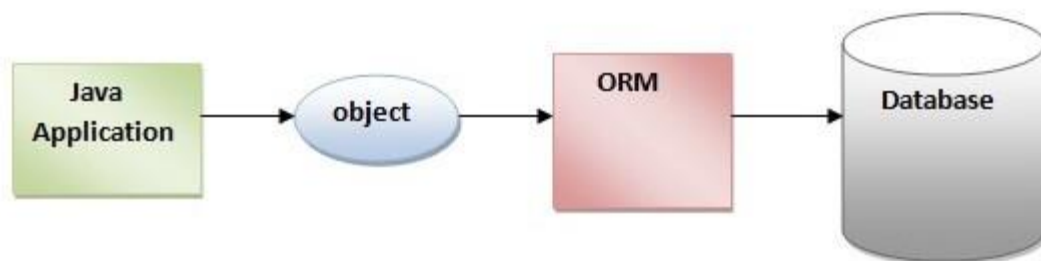
Collection	Collections
java.util.Collection is an interface	java.util.Collections is a class
Is used to represent a group of objects as a single entity	It is used to define various utility method for collection objects
It is the root interface of the Collection framework	It is a utility class
It is used to derive the data structures of the Collection framework	It contains various static methods which help in data structure manipulation

212).What is hibernate

Ans- Hibernate is a Java framework that simplifies the development of Java application to interact with the database. It is an open source, lightweight, ORM (Object Relational Mapping) tool. Hibernate implements the specifications of JPA (Java Persistence API) for data persistence.

ORM Tool

An ORM tool simplifies the data creation, data manipulation and data access. It is a programming technique that maps the object to the data stored in the database.



213)How does Hibernate work?

Ans- Hibernate is an open source Object-Relational Persistence and Query service for any Java Application. Hibernate maps Java classes to database tables and from Java data types to SQL data types and relieves the developer from most common data persistence related programming tasks. Hibernate sits between traditional Java objects and database server to handle all the works in persisting those objects based on the appropriate O/R mechanisms and patterns.

214) Why use Hibernate?

Hibernate reduces lines of code by maintaining object-table mapping itself and returns result to application in form of Java objects. It relieves programmer from manual handling of persistent data, hence reducing the development time and maintenance cost.

215) . what are idempotent methods of rest Api ?

Ans- REST APIs use HTTP methods such as POST, PUT, and GET to interact with resources such as an image, customer name, or document. When using an idempotent method, the method can be called multiple times without changing the result.

HTTP methods include:

- **POST** – Creates a new resource. POST **is not** idempotent and it **is not** safe.
- **GET** – Retrieves a resource. GET **is** idempotent and it **is** safe.
- **HEAD** – Retrieves a resource (without response body). HEAD **is** idempotent and it **is** safe.
- **PUT** – Updates/replaces a resource. PUT **is** idempotent but it **is not** safe.
- **PATCH** – Partially updates a resource. PATCH **is not** idempotent and it **is not** safe.
- **DELETE** – Deletes a resource. DELETE **is** idempotent but it **is not** safe.
- **TRACE** – Performs a loop-back test. TRACE **is** idempotent but it **is not** safe.

216) .what is application. properties file and application.yml file?

Ans- 1.) **.properties file** : It store data in sequential format. **.yml file** : It store data in hierarchical format.

2.) **.properties file** : It supports only key-value pair basically string values. **.yml file** : It supports key-value pair as well as map, list & scalar type values.

3.) **.properties file** : This file specifically used for JAVA. **.yml file** : This file type are used by many of the languages like JAVA, Python, ROR, etc.

4.) If you want to handle multiple profiles, **.properties file** : In this case you need to manage individual file for each profile. **.yml file** : In this file type you just need to manage single file and place configuration data of specific profile inside it.

5.) For Spring project, **.properties file** : @PropertySource annotation support this file type. **.yml file** : @PropertySource annotation can't support this file type.

217) .What is Dependency injection and inversion of control in spring?

- **Ans-**
Dependency Injection(DI):
Dependency injection generally means **passing a dependent object as a parameter to a method, rather than having the method create the dependent object.**

What it means in practice is that the method does not have a direct dependency on a particular implementation; any implementation that meets the requirements can be passed as a parameter.

With this implementation of objects defines their dependencies. And spring makes it available.

This leads to loosely coupled application development.

Inversion of Control(IoC) Container:

This is common characteristic of frameworks, IoC **manages java objects**

- from instantiation to destruction through its BeanFactory.

- Java components that are instantiated by the IoC container are called beans, and the **IoC container manages a bean's scope, lifecycle events, and any AOP features** for which it has been configured and coded.

218) .What is Runtime polymorphism? Live example

Ans- Runtime polymorphism in java is also known as **Dynamic Binding or Dynamic Method Dispatch**. In this process, the call to an overridden method is resolved dynamically at runtime rather than at compile-time. Runtime polymorphism is achieved through **Method Overriding**.

- Method Overriding is done when a child or a subclass has a method with the same name, parameters and return type as the parent or the superclass, then that function overrides the function in the superclass. In simpler terms, if the subclass provides its definition to a method already present in the superclass, then that function in the base class is said to be overridden.

Real-Life Examples of Java Polymorphism

A security guard outside an organization behaves differently with different people entering the organization. He acts in a different way when the Boss comes and, in another way when the employees come.

When the customers enter, the guard will respond differently. So here, the behavior of the guard is in various forms, which depends on the member who is coming.

219) .Diff bet hashSet and hashMap?

Basis	HashMap	HashSet
Definition	Java HashMap is a hash table based implementation of Map interface.	HashSet is a Set. It creates a collection that uses a hash table for storage.
Implementation	HashMap	HashSet implements Set,

	implements Map, Cloneable, and Serializable interfaces.	Cloneable, Serializable, Iterable and Collection interfaces.
Stores	In HashMap we store a key-value pair . It maintains the mapping of key and value.	In HashSet, we store objects .
Duplicate values	It does not allow duplicate keys , but duplicate values are allowed .	It does not allow duplicate values .
Null values	It can contain a single null key and multiple null values .	It can contain a single null value .
Method of insertion	HashMap uses the put() method to add the elements in the HashMap.	HashSet uses the add() method to add elements in the HashSet.
Performance	HashMap is faster/ than HashSet because values are associated with a unique key .	HashSet is slower than HashMap because the member object is used for calculating hashcode value, which can be same for two objects.
The Number of objects	Only one object is created during the add operation.	There are two objects created during put operation, one for key and one for value .
Storing Mechanism	HashMap internally uses hashing to store objects.	HashSet internally uses a HashMap object to store objects.
Uses	Always prefer when we do not maintain the uniqueness .	It is used when we need to maintain the uniqueness of data.

220) .In which scenario you ll go for list and set?

Ans- According to our requirement we have to choose the best one.

List:- List is one interface it has some implementation like, ArrayList, LinkedList, Vector, Stack

List can stores the elements (Objects) in **insertion order**, and **duplicate elements** also allowed, So whenever requirement is there , where duplicates are allowed anmd insertion order required , then we should go for List.

Set:- Similarlty Set is also one interface, it has implementations like, HashSet, LinkedHashSet, SortedSet, TreeSet

Set allows only **unique elements**, and **insertion order is not preserved**, So when ever this types of requirement is there , go for Set.

Map:- Map is also one interface , it has also some implementation classes, Map Stores the object as key and value pair,

Eg, Your emp id and name,

221) .can you explain lambda expression in java 8?

Ans- Lambda expression is a new and important feature of Java which was included in Java SE 8. 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.

Lambda expression provides implementation of *functional interface*. An interface which has only one abstract method is called functional interface.

Why use Lambda Expression

1. To provide the implementation of Functional interface.
2. Less coding.

Java Lambda Expression Syntax

1. (argument-list) -> {body}

222) ..what is stream in java 8 what is use of this?

Ans- Java Stream doesn't store data, it operates on the source data structure (collection and array) and produce pipelined data that we can use and perform specific operations. Such as we can create a stream from the list and filter it based on a condition.

Java Stream operations use functional interfaces, that makes it a very good fit for functional programming using lambda expression. As you can see in the above example that using lambda expressions make our code readable and short.

Java 8 Stream internal iteration principle helps in achieving lazy-seeking in some of the stream operations. For example filtering, mapping, or duplicate removal can be implemented lazily, allowing higher performance and scope for optimization.

1. **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.

```
List number = Arrays.asList(2,3,4,5);  
List square = number.stream().map(x->x*x).collect(Collectors.toList());
```

2. **filter:** The filter method is used to select elements as per the Predicate passed as argument.

```
List names = Arrays.asList("Reflection","Collection","Stream");  
List result = names.stream().filter(s->s.startsWith("S")).collect(Collectors.toList());
```

3. **sorted:** The sorted method is used to sort the stream.

```
List names = Arrays.asList("Reflection","Collection","Stream");
List result = names.stream().sorted().collect(Collectors.toList());
```

4. **collect:** The collect method is used to return the result of the intermediate operations performed on the stream.

```
List number = Arrays.asList(2,3,4,5,3);
Set square = number.stream().map(x->x*x).collect(Collectors.toSet());
```

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

```
List number = Arrays.asList(2,3,4,5);
number.stream().map(x->x*x).forEach(y->System.out.println(y));
```

223) .have you seen concurrent modification exception? why it occur?

Ans- Concurrent Modification Exception occurs when a thread in a program is trying to modify an object, which does not have permissions to be edited while in the current process. So simply, when we attempt to edit an object which is being currently used by another process, the ConcurrentModificationException will appear.

224) ..Which dummy variable is used in internal working of HashSet as value for every key

```
// Dummy value to associate with an Object in the backing Map
private static final Object PRESENT = new Object();
```

And that's how add method is implemented in HashSet class -

```
public boolean add(E e) {
    return map.put(e, PRESENT)!=null;
}
```

For example a statement for adding an element to HashSet- `set.add("Mumbai");` internally translates into `map.put("Mumbai", PRESENT);` and then added to the backing HashMap instance.

225) .can you used JSP directly in spring boot like spring mvc?

Ans- No

tomcat-embed-jasper

we need to include the tomcat-embed-jasper dependency to allow our application to compile and render JSP pages:

```
<dependency>
  <groupId>org.apache.tomcat.embed</groupId>
  <artifactId>tomcat-embed-jasper</artifactId>
```

```
<version>9.0.44</version>
</dependency>
```

Scenario Based questions on Exception handling-----

226) . What will be the output of the below program?

```
public class JavaHungry {
    public static void main(String args[])
    {
        try
        {
            System.out.print("A");
            int num = 99/0;
            System.out.print("B");
        }
        catch(ArithmeticException ex)
        {
            System.out.print("C");
        }
        catch(Exception ex)
        {
            System.out.print("D");
        }
        System.out.print("E");
    }
}
```

04

Output :

ACE

227) . What will be the output of the below program?

```
public class JavaHungry {
    public static void main(String args[])
    {
        try
        {
            System.out.print("A");
            int num = 99/0;
            System.out.print("B");
        }

        System.out.print("C");

        catch(ArithmeticException ex)
        {
            System.out.print("D");
        }
    }
}
```

```
}  
}  
}
```

Out Output :

Compilation Error

/JavaHungry.java:4: error: 'try' without 'catch', 'finally' or resource declarations

228) . What will be the output of the below program?

```
class SubException extends Exception { }  
class SubSubException extends SubException { }  
public class JH  
{  
    public void doStuff() throws SubException { }  
}  
class JH2 extends JH  
{  
    public void doStuff() throws SubSubException { }  
}  
class JH3 extends JH  
{  
    public void doStuff() throws Exception { }  
}  
class JH4 extends JH  
{  
    public void doStuff(int x) throws Exception { }  
}  
class JH5 extends JH  
{  
    public void doStuff() { }  
}
```

Output :

Compilation fails due to line `public void doStuff() throws Exception { }`

229) . What will be the output of the below program?

```
public class JavaHungry {  
    public static void main(String args[])  
    {  
        try  
        {  
            System.out.print("A");  
            throw 99;  
        }  
        catch (int ex)  
        {
```

```

        System.out.print("B");
    }
}
}

```

Output :

Compilation Errors

/JavaHungry.java:7: error: incompatible types: int cannot be converted to Throwable

230) What will be the output of the below program?

```

public class JavaHungry {
    public static void main(String args[])
    {
        try
        {
            int arr[] = {1, 2, 3, 4, 5};
            for (int i = 0; i <= 5; i++)
            {
                System.out.print ("Array elements are : " + arr[i] + "\n");
            }
        }
        catch (Exception e)
        {
            System.out.println ("Exception : " + e);
        }
        catch (ArrayIndexOutOfBoundsException ex)
        {
            System.out.println ("ArrayIndexOutOfBoundsException : " + ex);
        }
    }
}

```

Output :

Compilation Error

231). Write the program for check String is Palindrome or not...

//Palindrome means from forward & from reverse the character sequence should be same.

//(“nitin” is palindrome String.....as reverse of “nitin” is “nitin”.)

```
package com.groupd.practice;
```

```
import java.util.Scanner;
```

```
public class PalendromeDemo {
```

```
    static String reverse = "";
```

```

public String getReverseString(String str) {
    for (int i = (str.length() - 1); i >= 0; i--) {
        reverse = reverse + (str.charAt(i));
    }
    return reverse;
}

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Enter the String");
    String str1 = scanner.next();

    PalendromeDemo palendromeDemo = new PalendromeDemo();
    String input=palendromeDemo.getReverseString(str1);

    System.out.println("Reverse of given String is ==> " + reverse);

    System.out.println("Input we stored >>" + str1);
    System.out.println("reverse >>" + input);

    if (str1.equals(input)) {
        System.out.println("Given String is palindrome");
    } else {
        System.out.println("Given String is not palindrome");
    }
}

```

Output :

Enter the String

nitin

Reverse of given String is ==>nitin

Input we stored >>nitin

reverse >>nitin

Given String is palindrome

232). Explain stereotype annotations in project.

Ans - There are four types of stereotype annotations as follows.

@Component, @Repository, @Service and @Controller annotations

Spring will automatically import the beans into the container and inject to dependencies. These annotations are called **Stereotype annotations** as well.

1.1 @Component

It is a class-level annotation. It is used to mark a Java class as a bean. A Java class annotated with @Component is found during the class-path. The Spring Framework pick it up and configure it in the application context as a **Spring Bean**.

@Component

```

public class Student {
    .....
}

```

1.2 @Repository

It is a class-level annotation. The repository is a **DAOs** (Data Access Object) that access the database directly. The repository does all the operations related to the database.

```
package com.java;

@Repository
public class TestRepository {
    public void delete() {
        // code
    }
}
```

1.3 @Service:

It is also used at class level. It tells the Spring that class contains the **business logic**.

Example

```
package com.service;

@Service
public class Service
{
    public void service1()
    {
        //business code
    }
}
```

1.4 @Controller:

The @Controller is a class-level annotation. It is a specialization of @Component. It marks a class as a web request handler. It is often used to serve web pages. By default, it returns a string that indicates which route to redirect. It is mostly used with @RequestMapping annotation.

@Controller

```
@RequestMapping("books")
public class BooksController
{
    @RequestMapping(value =("/{name}", method = RequestMethod.GET)
    public Employee getBooksByName()
    {
        return booksTemplate;
    }
}
```

233). Explain @SpringBootApplication.

Ans- **@SpringBootApplication:** It is a combination of three annotations **@EnableAutoConfiguration**, **@ComponentScan**, and **@Configuration**.

2.1 @EnableAutoConfiguration:

It auto-configures the bean that is present in the classpath and configures it to run the methods. The use of this annotation is reduced in Spring Boot because developers provided an alternative of the annotation, i.e. **@SpringBootApplication**.

2.2 @ComponentScan:

It is used when we want to scan a package for beans. It is used with the annotation **@Configuration**. We can also specify the base packages to scan for Spring Components.

Example

```
@ComponentScan(basePackages = "com.test")
@Configuration
public class ScanComponent
{
    // ...
}
```

2.3 @Configuration:

It is a class-level annotation. The class annotated with **@Configuration** used by Spring Containers as a source of bean definitions.

Example

```
@Configuration
public class Demo
{
    @Bean Demo m1()
    {
        return new Demo();
    }
}
```

234) . What is IOC and DI ?

Ans- Spring IOC container is responsible to instantiate, configure and assemble the objects. The IoC container gets informations from the XML file and works accordingly. The main tasks performed by IoC container are:

1. Read the XML file.
2. Create the intantiation of xml bean(java classes).
3. It will manage the life cycle of your bean classes.
4. Passing the dynamic parameter to bean (java classes).

There are two types of IOC container—

1. BeanFactory--
2. ApplicationContext--

Dependency Injection-

The process of passing the required input from xml file to POJO class called as “Dependency injection.”

There are two types of dependency injection such as

1. Setter base injection
2. Constructor base injection

235). What is Spring Boot Starter ?

Ans- Spring Boot provides a number of starters that allow us to add jars in the classpath. Spring Boot built-in **starters** make development easier and rapid. Spring Boot Starters are the dependency descriptors.

In the Spring Boot Framework, all the starters follow a similar naming pattern: spring-boot-starter-*, where * denotes a particular type of application. For example, if we want to use Spring and JPA for database access, we need to include the spring-boot-starter-data-jpa dependency in our pom.xml file of the project.

236). How many ways we can break singleton pattern ?

Ans- There are 3 ways to break singleton pattern..

1. Break by Cloning. If a Singleton class implements java.lang.Cloneable interface then invoking clone() method on its single instance creates a duplicate object. ...

* To overcome this issue, override clone() method and throw an exception from clone method that is CloneNotSupportedException. Now whenever user will try to create clone of singleton object, it will throw exception and hence our class remains singleton.

2. Serialization and Deserialization also breaks Singleton.

* To overcome this issue we have to implement readResolve().

3. Reflection can instantiate a Singleton multiple times.

237). What is role of the @RequestBody in rest controller ?

Ans- @RequestBody:

It is used to bind HTTP request with an object in a method parameter. Internally it uses HTTP

MessageConverters to convert the body of the request. When we annotate a method parameter with **@RequestBody**, the Spring framework binds the incoming HTTP request body to that parameter.

Ans- The an HTTP response to a client made to Status code is integer. The status code is specify one of classes of The last two code do not categorization role.

S.N.	Code and Description
1	1xx---Informational-Response It is used to show that the request was received, and the process is continuing.
2	2xx---Successful It is used to show that the request was successfully received, understood, and accepted.
3	3xx---Redirection It is used to show that further action needs to be taken to complete the request.
4	4xx---Client-Error It is used to show that the request contains bad syntax or cannot be fulfilled.
5	5xx---Server-Error It is used to show that the server is failed to fulfill an apparently valid request.

238) .
Explain status code.

Server issues Status Code in request of the the server. a 3-digit first digit of used to five standard responses. digits of status have any

The status codes are divided into 5 parts, as follows:

239).What is Sprint Duration?

Ans- First introducing concept of Sprint..

A *sprint* is a short, time-boxed period when a scrum team works to complete a set amount of work.

Sprint duration we simply say, the time span in which we have to complete defined task.

A “sprint” is like a “mini project” inside your project. And in Scrum we divide our full project up into these equally long “mini projects” called sprints. Each sprint has its own planning. Every sprint has the same length and is never “extended” (think of a sprint like the 24 hours in a day—you wouldn’t add another hour simply because you didn’t get your work done).

How to declare sprint duration?

- **If the goal is risk reduction, then I choose a shorter sprint length.** 1 week sprints
- **If your goal is to develop a new product for which you already reduced basic risks choose a medium sprint length** 2 week sprints
- If your goal is to develop or improve a product that is already profitable go for the longer sprint length. The longest sensible sprint length is 1 month.
→ 4 week sprints

240). What is difference between Spring & Spring Boot?

Ans- Spring vs. Spring Boot

Spring: Spring Framework is the most popular application development framework of Java. The main feature of the Spring Framework is dependency Injection or Inversion of Control (IoC). With the help of Spring Framework, we can develop a loosely coupled application. It is better to use if application type or characteristics are purely defined.

Spring Boot: Spring Boot is a module of Spring Framework. It allows us to build a **stand-alone application with minimal or zero configurations**. It is better to use if we want to develop a simple Spring-based application or RESTful services.

The primary comparison between Spring and Spring Boot are discussed below:

Spring	Spring Boot
Spring Framework is a widely used Java EE (Enterprise Edition) framework for building applications.	Spring Boot Framework is widely used to develop REST APIs.
It aims to simplify Java EE development that makes developers more productive.	It aims to shorten the code length and provide the easiest way to develop Web Applications.
The primary feature of the	The primary feature of Spring Boot is

Spring Framework is dependency injection.	Autoconfiguration. It automatically configures the classes based on the requirement.
It helps to make things simpler by allowing us to develop loosely coupled applications.	It helps to create a stand-alone application (that runs locally on the device & doesn't require anything else to be function) with less configuration.
The developer writes a lot of code (boilerplate code) to do the minimal task.	It reduces boilerplate code. (Boilerplate code means sections of codes that repeated multiple places with small variation or no variation)
To test the Spring project, we need to set up the sever explicitly.	Spring Boot offers embedded server such as Jetty and Tomcat, etc. (Just make changes in application. Properties file)
Developers manually define dependencies for the Spring project in pom.xml.	Spring Boot comes with the concept of starter in pom.xml file that internally takes care of downloading the dependencies JARs based on Spring Boot Requirement.

241 .Tell Java 8 Features

Ans-Oracle released a new version of Java as Java 8 in March 18, 2014. It was a revolutionary release of the Java for software development platform. It includes various upgrades to the Java programming, JVM, Tools and libraries.

Java 8 Programming Language Enhancements

Java 8 provides following features for Java Programming:

- Lambda expressions,
- Method references,
- Functional interfaces,

- Stream API,
- Stream Filter
- Default methods,
- Static methods in interface
- forEach() method,
- Date/Time API

Lambda Expressions

Lambda expression helps us to write our code in functional style. It provides a clear and concise way to implement SAM interface(Single Abstract Method) by using an expression. It is very useful in collection library in which it helps to iterate, filter and extract data.

Method References

Java 8 Method reference is used to refer method of functional interface . It is compact and easy form of lambda expression. Each time when you are using lambda expression to just referring a method, you can replace your lambda expression with method reference.

Default Methods

Java provides a facility to create default methods inside the interface. Methods which are defined inside the interface and tagged with default keyword are known as default methods. These methods are non-abstract methods and can have method body.

forEach

Java provides a new method forEach() to iterate the elements. It is defined in Iterable and Stream interfaces. It is a default method defined in the Iterable interface. Collection classes which extends Iterable interface can use forEach() method to iterate elements.

Date/Time API

Java has introduced a new Date and Time API since Java 8. The java.time package contains Java 8 Date and Time class.

Functional Interface

An Interface that contains only one abstract method is known as functional interface. It can have any number of default and static methods. It can also declare methods of object class. Functional interfaces are also known as Single Abstract Method Interfaces (SAM Interfaces)

Stream API

Java 8 java.util.stream package consists of classes, interfaces and an enum to allow functional-style operations on the elements. It performs lazy computation. So, it executes only when it requires.

Stream Filter

Java stream provides a method filter() to filter stream elements on the basis of given predicate. Suppose, you want to get only even elements of your list, you can do this easily with the help of filter() method

Ans- The **Java throws keyword** is used to declare an exception. It gives an information to the programmer that there may occur an exception. So, it is better for the programmer to provide the exception handling code so that the normal flow of the program can be maintained.

Exception Handling is mainly used to handle the checked exceptions. If there occurs any unchecked exception such as NullPointerException, it is programmers' fault that he is not checking the code before it being used.

Syntax of Java throw

1. return_type method_name() **throws** exception_class_name{
2. //method code
3. }

243) .Can we declare multiple exception in single catch block?

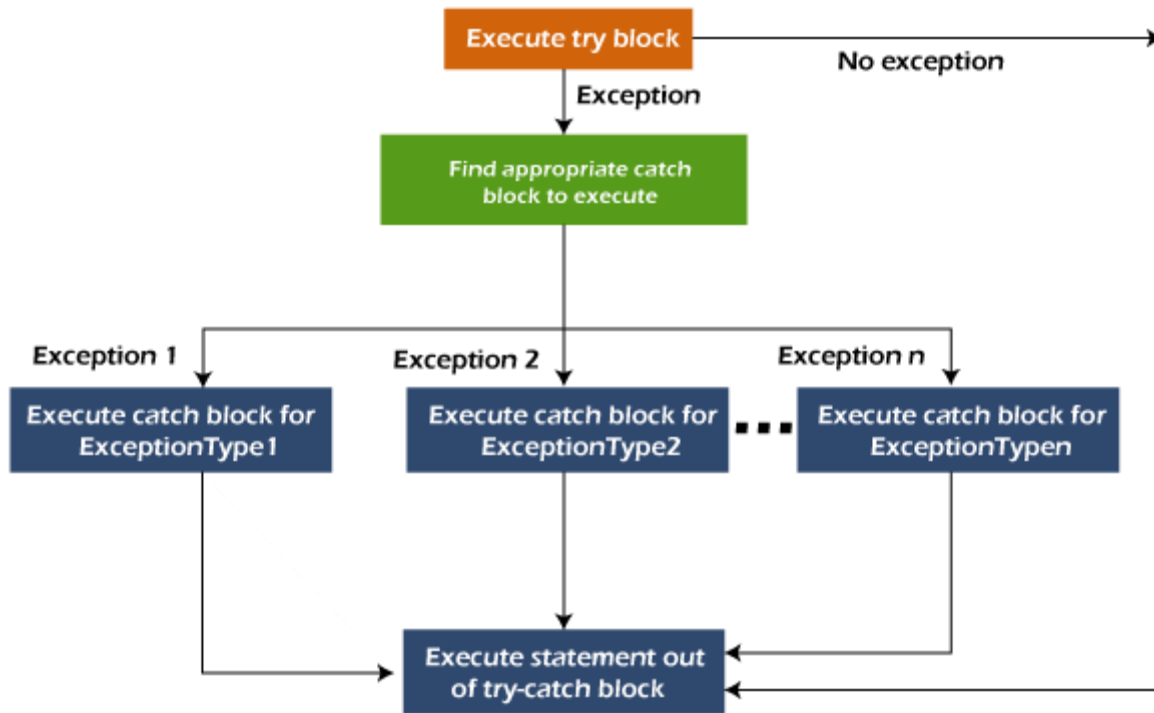
Ans- Yes. we can declare it from java 7.

In java 7, we can catch multiple exceptions in a single catch block as:

```
catch(IOException | SQLException ex){  
    logger.error(ex);  
    throw new MyException(ex.getMessage());  
}
```

If a catch block handles multiple exceptions, you can separate them using a pipe (|) and in this case, exception parameter (ex) is final, so you can't change it. The byte code generated by this feature is smaller and reduce code redundancy.

Flowchart of Multi-catch Block



Example 1

Let's see a simple example of java multi-catch block.

MultipleCatchBlock1.java

```
1. public class MultipleCatchBlock1 {
2.
3.     public static void main(String[] args) {
4.
5.         try{
6.             int a[]=new int[5];
7.             a[5]=30/0;
8.         }
9.         catch(ArithmeticException e)
10.            {
11.                System.out.println("Arithmetic Exception occurs");
12.            }
13.        catch(ArrayIndexOutOfBoundsException e)
14.            {
15.                System.out.println("ArrayIndexOutOfBoundsException Exception occurs");
16.            }
17.        catch(Exception e)
```

```

18.      {
19.      System.out.println("Parent Exception occurs");
20.      }
21.      System.out.println("rest of the code");
22.  }
23. }

```

Test it Now

Output:

```

Arithmetic Exception occurs
rest of the code

```

244) .Custom exception should be checked exception or modified exception?

Ans- There is not a specific rule for this, just keep this in mind: If a client can reasonably be expected to recover from an exception, make it a checked exception. If a client cannot do anything to recover from the exception, make it an unchecked exception

245) .What is application context?

Ans -The Application Context is Spring's advanced container. Similar to BeanFactory, it can load bean definitions, wire beans together, and dispense beans upon request. This container is defined by *org.springframework.context.ApplicationContext* interface .

The *ApplicationContext* includes all functionality of the *BeanFactory*, It is generally recommended over BeanFactory. BeanFactory can still be used for lightweight applications like mobile devices or applet-based applications.

The *ApplicationContext* includes all functionality of the *BeanFactory*, I The most commonly used *ApplicationContext* implementations are –

- *FileSystemXmlApplicationContext* – This container loads the definitions of the beans from an XML file. Here you need to provide the full path of the XML bean configuration file to the constructor.
- *ClassPathXmlApplicationContext* – This container loads the definitions of the beans from an XML file. Here you do not need to provide the full path of the XML file but you need to set CLASSPATH properly because this container will look like bean configuration XML file in CLASSPATH.
- *WebXmlApplicationContext* – This container loads the XML file with definitions of all beans from within a web application.is generally recommended over beanFactory.

246) .What is @Qualifier annotation?

Ans-There may be a situation when you create more than one bean of the same type and want to wire only one of them with a property. In such cases, you can use the @Qualifier annotation along with @Autowired to remove the confusion by specifying which exact bean will be wired.

247) .Types of Autowiring & when to use it?

Ans- Autowiring feature of spring framework enables you to inject the object dependency implicitly. It internally uses setter or constructor injection. In simple sentence we can say, Autowiring enables automatic dependency injection.

Autowiring can't be used to inject primitive and string values. It works with reference only.

Autowiring Modes

There are many autowiring modes:

No.	Mode	Description
1)	no	It is the default autowiring mode. It means no autowiring by default.
2)	byName	The byName mode injects the object dependency according to name of the bean. In such case, property name and bean name must be same. It internally calls setter method.
3)	byType	The byType mode injects the object dependency according to type. So property name and bean name can be different. It internally calls setter method.
4)	constructor	The constructor mode injects the dependency by calling the constructor of the class. It calls the constructor having large number of parameters.
5)	autodetect	It is deprecated since Spring 3

248) .What is Circular dependency ?

Ans- It happens when a bean A depends on another bean B, and the bean B depends on the bean A as well:

Bean A → Bean B → Bean A

Of course, we could have more beans implied:

Bean A → Bean B → Bean C → Bean D → Bean E → Bean A

Spring

2. What Happens in

When Spring context is loading all the beans, it tries to create beans in the order needed for them to work completely. For instance, if we didn't have a circular dependency, like the following case:

Bean A → Bean B → Bean C

Spring will create bean C, then create bean B (and inject bean C into it), then create bean A (and inject bean B into it).

But, when having a circular dependency, Spring cannot decide which of the beans should be created first, since they depend on one another. In these cases, Spring will raise a *BeanCurrentlyInCreationException* while loading context.

It can happen in Spring when using **constructor injection**; if you use other types of injections you should not find this problem since the dependencies will be injected when they are needed and not on the context loading.

249) . If my requirement is -- I want to store policy Details in Policy, Health, Insurance table so policy number as primary key in the form of

PLC_LI_001

PLC_LI_002

PLC_LI_003

not 1 2 3

how u will do or which concept u will use for this?

Ans- Hibernate supports many built in generator classes like assigned, sequence, increment, identity, native etc. But some requirements these generator classes we can't use. There is no built in generator classes support this type of primary key generated value. So we need to go for Custom generator classes.

We need a Hibernate identifier generator that can take any value that we manually assign, as a

PLC_LI_001

PLC_LI_002

PLC_LI_003

In order to create our own generator class, we need to implement with the org.hibernate.id.IdentifierGenerator interface. This interface is having one method. So this method we need override.

```
package com.varasofttech.generator;

import org.hibernate.id.IdentifierGenerator;

public class MyGenerator implements IdentifierGenerator
{
    @Override
    public Serializable generate(SessionImplementor session, Object object)
    {
        // your logic comes here.
    }
}
```

250) .what is threshold is decided in java 8 for internal working of hashmap for internal structure of linked list??

HashMap implementation in Java provides constant-time performance $O(1)$ for `get()` and `put()` methods in the ideal case when the Hash function distributes the objects evenly among the buckets. In Java 8, you still have an array, but it now stores Nodes that contain the exact same information as `Entries` and, therefore, are also linked lists.

Below is the Node implementation in Java 8:

```
static class Node implements Map.Entry {
```

```
    final int hash;
```

```
    final K key;
```

1

2

3

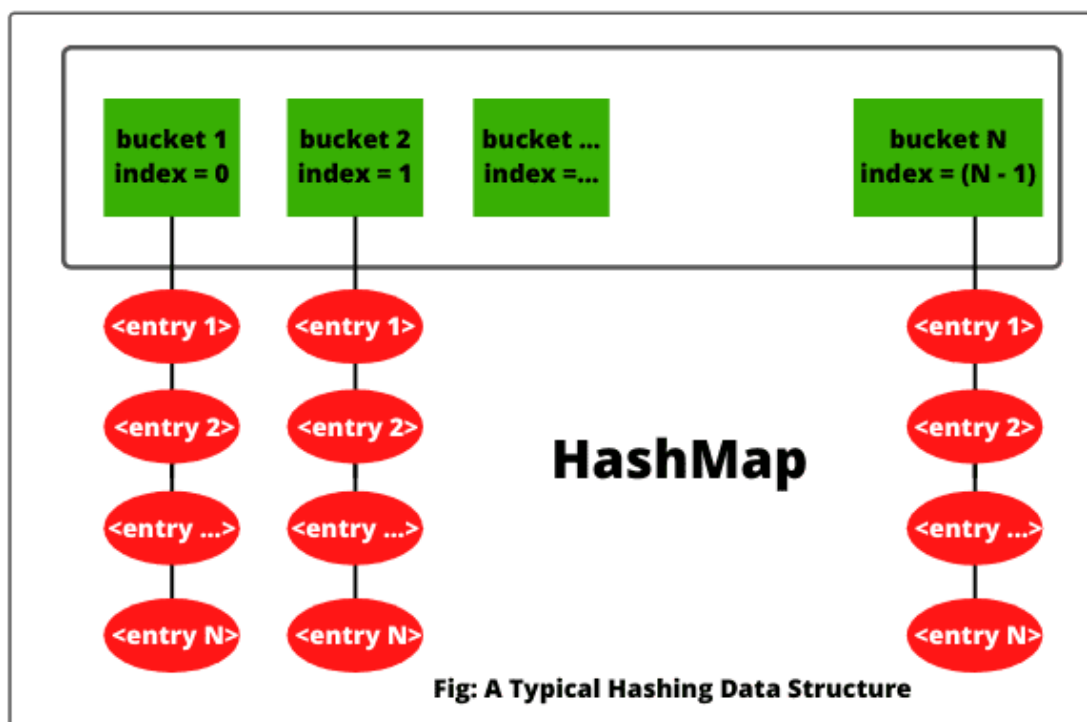
V value;	4
Node next;	5
}	6

Well, **Nodes** can be extended to **TreeNode**s. A **TreeNode** is a balanced tree structure that maintains $O(\log(n))$ complexity for add, delete, or get. **TreeNode** also ensures that their length is always $\log(n)$ despite new adds or removes of nodes.

Thus, the performance of **HashMap** degrades gracefully if **hashCode()** method is not used properly, where it returns values that are poorly distributed or those in which many keys share a **hashCode**.

To address this issue in Java 8 **Map** transforms it into bins of **TreeNode**, each structured similarly to those in **java.util.TreeMap** once the threshold(**TREEIFY_THRESHOLD**) is reached. This means that **HashMap** starts with storing **Entry** objects in bins of the linked list but after the number of items in a **Map** becomes larger than a certain threshold, it is transformed from bins of a linked list to **TreeNode**, this improves the worst-case performance from $O(n)$ to $O(\log n)$. And when they become too small (due to removal or resizing), they are converted back to plain bins.

251) .which internal structure came in picture after threshold crossed for internally hashmap?



252)explain all joins in sql

It is used to retrieve data from multiple tables. It is performed whenever you need to fetch records from two or more tables.

By using one select statement, we can retrieve the the data from multiple table.

types of [SQL](#) joins:

- **(INNER) JOIN**: Returns records that have matching values in both tables
- **LEFT (OUTER) JOIN**: Returns all records from the left table, and the matched records from the right table
- **RIGHT (OUTER) JOIN**: Returns all records from the right table, and the matched records from the left table
- **FULL (OUTER) JOIN**: Returns all records when there is a match in either left or right table

1) Inner Join-

It gives you exactly matching rows called inner join.

Syntax-

```
SELECT columns FROM table1 INNER JOIN table2 ON table1.column = table2.column;
```

Example-

Table 1-

create table customers(

customerid int(10)primary key auto_increment not null, customername varchar(32),

email varchar(32), phone int(255));

	customerid	customername	email	phone
▶	1	jeevan	jk@gmail.com	8888972688
	2	ashok	ashok@gmail.com	8585858585
	3	ram	ram@gmail.com	7878787878
	4	rohan	rohan@gmail.com	8686868686
	5	sohan	sohan@gmail.com	7777888877
*	NULL	NULL	NULL	NULL

create table accounts(customerid int,

accountsid int primary key auto_increment not null,

accountstype varchar(10), balance int(50), foreign key(customerid) references customers(customerid));

	customerid	accountsid	accountstype	balance
▶	1	1	saving	5000
	2	2	current	7500
*	NULL	NULL	NULL	NULL

2. Left outer join-The LEFT OUTER JOIN returns all rows from the left hand table (Table 1) specified in the ON condition and only those rows from the other table where the join condition is fulfilled.

Syntax-SELECT columns FROM table1 LEFT [OUTER] JOIN table2 ON

table1.column = table2.column;

Example-

select customername, email, accountsid, balance from customers left join accounts

on customers.customerid = accounts.customerid;

	customername	email	accountsid	balance
▶	jeevan	jk@gmail.com	1	5000
	ashok	ashok@gmail.com	2	7500
	ram	ram@gmail.com	NULL	NULL
	rohan	rohan@gmail.com	NULL	NULL
	sohan	sohan@gmail.com	NULL	NULL

3. Right outer join-The MySQL Right Outer Join returns all rows from the RIGHT-hand table (Table 2) specified in the ON condition and only those rows from the other table where the join condition is fulfilled.

Syntax-SELECT columns FROM table1 RIGHT [OUTER] JOIN table2

ON table1.column = table2.column;

Example-

select customername, email, accountsid, balance from customers right join accounts

on customers.customerid = accounts.customerid;

	customername	email	accountsid	balance
▶	jeevan	jk@gmail.com	1	5000
	ashok	ashok@gmail.com	2	7500

4. Full outer Join-

In MySQL it is the combination of left join union right join.

select * from customers left join accounts

on customers.customerid = accounts.customerid union

select * from customers right join accounts

on customers.customerid = accounts.customerid;

	customerid	customername	email	phone	customerid	accountsid	accountstype	balance
▶	1	jeevan	jk@gmail.com	8888972688	1	1	saving	5000
	2	ashok	ashok@gmail.com	8585858585	2	2	current	7500
	3	ram	ram@gmail.com	7878787878	NULL	NULL	NULL	NULL
	4	rohan	rohan@gmail.com	8686868686	NULL	NULL	NULL	NULL
	5	sohan	sohan@gmail.com	7777888877	NULL	NULL	NULL	NULL

253)What is abstraction??How to achieve 100% abstraction?

Abstraction- It is the process of hiding the certain details and showing the important information to the end user called as “Abstraction”.

Example- By using ATM GUI screen bank people are highlighting the set of services what they are offering without highlighting internal implementation.

Advantages of abstraction-

- 1) We can achieve security as we are not highlighting our internal implementation.
- 2) Enhancement will become very easy because without effecting end user we can able to perform any type of changes in our internal system.
- 3) It provides more flexibility to the end user to use system very easily.
- 4) It improves maintainability of the application.

How to achieve the Abstraction in java?

There are two ways to achieve the abstraction in java.

1. Abstract class
2. Interface

Using interface, we can achieve 100% abstraction- The user who want to use methods of the interface, he only knows the classes that implement this interface and their methods, information about the implementation is completely hidden from the user, thus achieving 100% abstraction.

254)what is multithreading??

It is the process of executing multiple threads simultaneously.

Multitasking is a process of executing multiple tasks simultaneously. It is achieved by using two ways.

Example for Multitasking-

In online session, what are the different activities done by students as?

- Listen the class
- Taking running notes
- Checking mobile

Process based-

1. Executing several tasks simultaneously where each task is separate independent process such as multitasking is called as process based.
2. Example 1- Typing java program into eclipse, also listening the audio songs, download a file from internet.
3. In this every activity is independent process here.

4. Example-2 Task manager, see the multiple process list.
5. Process is heavy weight components.
6. Each process has address into memory.

Thread based-

1. Executing several tasks simultaneously where each task is separate part of same program called as thread based.
2. Example- suppose I have 1000 lines of code into java program and it will takes 8 hours to execute it where first 500 line is executed after that remaining 500 lines is executed but there is no any dependency between them so I can run that tasks simultaneously to minimize the execution time.
3. Thread is light weight components.
4. Thread shares the same address space.

Java Multithreading is mostly used in games, animation, etc.

Advantages of Java Multithreading-

- 1) It doesn't block the user because threads are independent and you can perform multiple operations at the same time.
- 2) You can perform many operations together, so it saves time.
- 3) Threads are independent, so it doesn't affect other threads if an exception occurs in a single thread.

255) Synchronized thread vs volatile keyword

5) why we use volatile keyword??

Synchronized-

- 1) Synchronized is the modifier applicable for methods and blocks but not for variables and classes.
- 2) If a method or block declared with synchronized keyword then at a time only one thread is allow to execute that method or block on the given object.
- 3) The main advantage of synchronized keyword is we can resolve data inconsistency problems, but the main disadvantage is it increases waiting time of the threads and effects performance of the system. Hence if there is no specific requirement never recommended to use synchronized keyword.

Volatile -

- 1) Volatile is the modifier applicable only for variables but not for classes and methods.
- 2) If the value of variable keeps on changing such type of variables we have to declare with volatile modifier.
- 3) If a variable declared as volatile then for every thread a separate local copy will be created by the jvm, all intermediate modifications performed by the thread will takes place in the local copy instead of master copy.
- 4) Once the value got finalized before terminating the thread that final value will be updated in master copy.

5) The main advantage of volatile modifier is we can resolve data inconsistency problems, but creating and maintaining a separate copy for every thread increases complexity of the programming and affects performance of the system. Hence if there is no specific requirement never recommended to use volatile modifier and it's almost outdated.

6) Volatile means the value keep on changing where as final means the value never changes hence final volatile combination is illegal for variables.

256) Spring Boot Annotations

Spring Boot Annotations is a form of metadata that provides data about a program. In other words, annotations are used to provide **supplemental** information about a program. It is not a part of the application that we develop. It does not have a direct effect on the operation of the code they annotate. It does not change the action of the compiled program.

Spring Boot Annotations

- **@EnableAutoConfiguration:** It auto-configures the bean that is present in the classpath and configures it to run the methods. The use of this annotation is reduced in Spring Boot 1.2.0 release because developers provided an alternative of the annotation, i.e. **@SpringBootApplication**.
- **@SpringBootApplication:** It is a combination of three annotations **@EnableAutoConfiguration**, **@ComponentScan**, and **@Configuration**.

Core Spring Framework Annotations

1) **@Required:** It applies to the **bean** setter method. It indicates that the annotated bean must be populated at configuration time with the required property, else it throws an exception **BeanInitializationException**.

Example

```
public class Machine
{
    private Integer cost;
    @Required
    public void setCost(Integer cost)
    {
        this.cost = cost;
    }
    public Integer getCost()
    {
        return cost;
    }
}
```


2) **@Autowired:** Spring provides annotation-based auto-wiring by providing @Autowired annotation. It is used to autowire spring bean on setter methods, instance variable, and constructor. When we use @Autowired annotation, the spring container auto-wires the bean by matching data-type.

Example

@Component

public class Customer

{

private Person person;

@Autowired

public Customer(Person person)

{

this.person=person;

}

}

3) **@Configuration:** It is a class-level annotation. The class annotated with @Configuration used by Spring Containers as a source of bean definitions.

Example

@Configuration

public class Vehicle

{

@BeanVehicle engine()

{

return new Vehicle();

}

}

4) **@ComponentScan:** It is used when we want to scan a package for beans. It is used with the annotation @Configuration. We can also specify the base packages to scan for Spring Components.

Example

@ComponentScan(basePackages = "com.javatpoint")

@Configuration

public class ScanComponent

{

```
// ...  
}
```

5) **@Bean**: It is a method-level annotation. It is an alternative of XML <bean> tag. It tells the method to produce a bean to be managed by Spring Container.

Example

@Bean

```
public BeanExample beanExample()  
{  
    return new BeanExample ();  
}
```

Spring Framework Stereotype Annotations

1) **@Component**: It is a class-level annotation. It is used to mark a Java class as a bean. A Java class annotated with **@Component** is found during the classpath. The Spring Framework pick it up and configure it in the application context as a **Spring Bean**.

Example

@Component

```
public class Student  
{  
    .....  
}
```

2) **@Controller**: The @Controller is a class-level annotation. It is a specialization of **@Component**. It marks a class as a web request handler. It is often used to serve web pages. By default, it returns a string that indicates which route to redirect. It is mostly used with **@RequestMapping** annotation.

Example

@Controller

@RequestMapping("books")

```
public class BooksController
```

```
{
```

```
@RequestMapping(value =("/{name}", method = RequestMethod.GET)
```

```
public Employee getBooksByName()
```

```
{
```

```
    return booksTemplate;
```

```
}  
}
```

- **@Service:** It is also used at class level. It tells the Spring that class contains the **business logic**.

Example

```
package com.javatpoint;  
  
@Service  
public class TestService  
{  
    public void service1()  
    {  
        //business code  
    }  
}
```

5) **@Repository:** It is a class-level annotation. The repository is a **DAOs** (Data Access Object) that access the database directly. The repository does all the operations related to the database.

```
package com.javatpoint;  
  
@Repository  
public class TestRepository  
{  
    public void delete()  
    {  
        //persistence code  
    }  
}
```

Spring MVC and REST Annotations

- **@RequestMapping:** It is used to map the **web requests**. It has many optional elements like **consumes**, **header**, **method**, **name**, **params**, **path**, **produces**, and **value**. We use it with the class as well as the method.

Example

```
@Controller  
public class BooksController  
{  
    @RequestMapping("/computer-science/books")  
    public String getAllBooks(Model model)
```

```
{  
//application code  
return "bookList";  
}
```

257) Types of beans

1) singleton

This scopes the bean definition to a single instance per Spring IoC container (default).

2) prototype

This scopes a single bean definition to have any number of object instances.

3) request

This scopes a bean definition to an HTTP request. Only valid in the context of a web-aware Spring ApplicationContext.

4)session

This scopes a bean definition to an HTTP session. Only valid in the context of a web-aware Spring ApplicationContext.

5)global-session

This scopes a bean definition to a global HTTP session. Only valid in the context of a web-aware Spring ApplicationContext.

258) @Controller vs @RestController

- 1) The **@Controller** is a annotation to mark class as Controller Class in Spring While **@RestController** is used in REST Web services and similar to **@Controller** and **@ResponseBody**.
- 2) The **@Controller** annotation indicates that the class is controller like web Controller while **@RestController** annotation indicates that the class is controller where **@RequestMapping** Method assume **@ResponseBody** by Default(i.e REST APIs).
- 3) The key difference is that you do not need to use **@ResponseBody** on each and every handler method once you annotate the class with **@RestController**.
- 4) **@Controller** create a Map of Model Object and find a view while **@RestController** simply return object and object data directly written into http response as JSON orXML.

This can be also done with **@Controller** annotation and **@ResponseBody** annotation but since this is the default

behaviour of RESTful Web Services. Spring introduced **@RestController** which is combined behaviour of **@Controller** and **@ResponseBody** together.

259) Methods in Rest API

What is REST?

The term REST stands for **RE**presentational **S**tate **T**ransfer. It is an architectural style that defines a set of rules in order to create Web Services.

In a client-server communication, REST suggests to create an object of the data requested by the client and send the values of the object in response to the user.

There are four types of method as

- Get-It is used to read resource
- Post-It is used to create new resource.
- Put-It is generally used to update resource
- Delete-it is used to delete source

260) what is use of Patch in Rest API

A PATCH request is used **to make changes to part of the resource at a location**. That is, it PATCHES the resource — changing its properties. It is used to make minor updates to resources and it's not required to be idempotent

261) SQL Constraints?

SQL constraints are used to specify rules for the data in a table.

Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.

Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table.

The following constraints are commonly used in SQL:

- NOT NULL - Ensures that a column cannot have a NULL value
- UNIQUE - Ensures that all values in a column are different
- PRIMARY KEY - A combination of a **NOT NULL** and **UNIQUE**. Uniquely identifies each row in a table
- FOREIGN KEY - Prevents actions that would destroy links between tables
- CHECK - Ensures that the values in a column satisfies a specific condition
- DEFAULT - Sets a default value for a column if no value is specified
- CREATE INDEX - Used to create and retrieve data from the database very quickly

262) What is stream API? How it is different than collection?

The Stream API is used to process collections of objects.

A [Stream](#) is a fixed data structure, in which the elements are computed on demand. The Stream API is used to process collections of objects. A stream is a sequence of objects that supports various methods that can be pipelined to produce the desired result.

The features of Java stream are:

- It takes input from the Collections, Arrays or I/O channels and is not a data structure.
- Streams only provide the result as per the pipelined methods and don't change the original data structure.

Collections and Streams, both are conceptually two different things which are used for two different purposes. If the collections are used to store the data then the streams are used to perform operations on that data.

Collections And Streams In Java :

1) Conceptual Difference

Collections are used to store and group the data in a particular data structure like *List*, *Set* or *Map*. But, streams are used to perform complex data processing operations like *filtering*, *matching*, *mapping* etc on stored data such as arrays, collections or I/O resources. That means, collections are mainly about data and streams are mainly about operations on data.

```
//Usage of collections
//Collections are mainly used to store the data
//Here, names are stored as List
List<String> names = new ArrayList<>();
names.add("Charlie");
names.add("Douglas");
names.add("Sundaraman");
names.add("Charlie");
names.add("Yuki");
//Usage of streams
//Streams are mainly used to perform operations on data
//like selecting only unique names
names.stream().distinct().forEach(System.out::println);
//Output :
//Charlie
//Douglas
//Sundaraman
//Yuki
```

2) Data Modification

You can add to or remove elements from collections. But, you can't add to or remove elements from streams. Stream consumes a source, performs operations on it and returns a result. They don't

modify even the source also.

```
List<String> names = Arrays.asList("Charlie", "Douglas", "Jacob");
```

```
//Adding elements to names
```

```
names.add("Sundaraman");
```

```
names.add("Yuki");

//Removing elements from names

names.remove(2);

//getting stream of unique names

Stream<String> uniqueNames = names.stream().distinct();

//You can't add or remove elements from stream

//There are no such methods in Stream
```

3) External Iteration Vs Internal Iteration

The main specialty of Java 8 Streams is that you need not to worry about iteration while using streams. Streams perform iteration internally behind the scene for you. You just have to mention the operations to be performed on a source.

On the other hand, you have to do the iteration externally over collections using loops.

```
List<String> names = new ArrayList<>();
```

```
names.add("Charlie");
```

```
names.add("Douglas");
```

```
names.add("Sundaraman");
```

```
names.add("Charlie");
```

```
names.add("Yuki");
```

```
//External iteration of collections
```

```
for (String name : names)
{
    System.out.println(name);
}
```

```
//Output :
```

```
//Charlie
```

```
//Douglas
```

```
//Sundaraman
```

```
//Charlie
```

```
//Yuki
```

```
//Internal iteration of streams. No for loops
```

```
names.stream().map(String::toUpperCase).forEach(System.out::println);
```

//Output :

```
//CHARLIE
//DOUGLAS
//SUNDARAMAN
//CHARLIE
//YUKI
```

Collections	Streams
Collections are mainly used to store and group the data.	Streams are mainly used to perform operations on data.
You can add or remove elements from collections.	You can't add or remove elements from streams.
Collections have to be iterated externally.	Streams are internally iterated.
Collections can be traversed multiple times.	Streams are traversable only once.
Collections are eagerly constructed.	Streams are lazily constructed.
Ex : List, Set, Map...	Ex : filtering, mapping, matching...

263) When you create the object?

An object is **created based on its class**. You can consider a class as a blueprint, template, or a description how to create an object. When an object is created, memory is allocated to hold the object properties. An object reference pointing to that memory location is also created.

264) What is the Hashtable?

- The Underlying Data Structure for Hashtable is Hashtable Only.
- Duplicate Keys are Not Allowed. But Values can be Duplicated.
- Insertion Order is Not Preserved and it is Based on Hashcode of the Keys.
- Heterogeneous Objects are Allowed for Both Keys and Values.
- null Insertion is Not Possible for Both Key and Values. Otherwise we will get Runtime
- Exception Saying NullPointerException.
- Every Method Present in Hashtable is Synchronized and Hence Hashtable Object is Thread Safe.

Constructors:

1) Hashtable h=new Hashtable();

- Creates an empty Hashtable object with default initialcapacity 11 and default fill ratio 0.75.

2) Hashtable h=new Hashtable(int initialcapacity);

3) Hashtable h=new Hashtable(int initialcapacity,float fillratio);

4) Hashtable h=new Hashtable (Map m);

265) What is the difference between Hashmap and hashtable? which is better?

HashMap	Hashtable
No Method Present in HashMap is Synchronized	Every Method Present in Hashtable is Synchronized.
At a Time Multiple Threads are allowed to Operate on HashMap Object simultaneously and Hence it is Not Thread Safe.	At a Time Only One Thread is allowed to Operate on the Hashtable Object and Hence it is Thread Safe.
Relatively Performance is High.	Relatively Performance is Low.
null is allowed for Both Keys and Values	null is Not allowed for Both Keys and Values. Otherwise we will get NPE.
Introduced in 1.2 Version and it is Non – Legacy.	Introduced in 1.0 Version and it is Legacy.

HashMap better for non-threaded applications, as unsynchronized Objects typically perform better than synchronized ones. Hashtable does not allow null keys or values.

266) What is client-server architecture?

This model is designed for the end-users called clients to access the resources from a central computer known as a server using network services.

Here, the clients make requests through a graphical user interface (GUI), and the server will give the desired output as soon as the instructions are matched.

267) What is autoconfiguration in spring ?

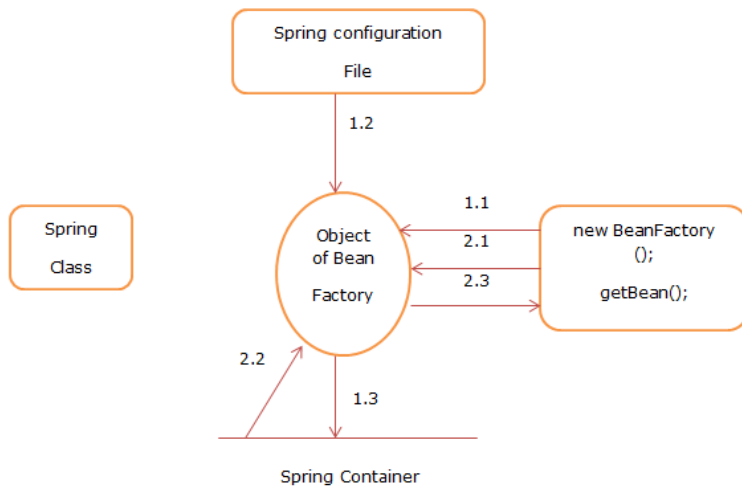
Spring Boot auto-configuration automatically configures the Spring application based on the jar dependencies that we have added.

For example, if the H2 database Jar is present in the classpath and we have not configured any beans related to the database manually, the Spring Boot's auto-configuration feature automatically configures it in the project.

We can enable the auto-configuration feature by using the annotation **@SpringBootApplication**.

268) Explain Spring bean life cycle?

Internal working of Spring-



- 1.1 BeanFactory object is created.
- 1.2 Spring Configuration file is injected into bean factory object.
- 1.3 Spring bean (POJO) object created and stored into Spring/IOC container.
- 2.1 getBean() method is invoked.
- 2.2 get the spring bean object (Student class object)
- 2.3 Bean object provided to user program.

269) What are the different scopes of spring bean?

Sr.No.	Scope & Description
1	singleton This scopes the bean definition to a single instance per Spring IoC container (default).
2	prototype This scopes a single bean definition to have any number of object instances.
3	request This scopes a bean definition to an HTTP request. Only valid in the context of a web-aware Spring ApplicationContext.
4	session This scopes a bean definition to an HTTP session. Only valid in the context of a web-aware Spring ApplicationContext.

5

global-session

This scopes a bean definition to a global HTTP session. Only valid in the context of a web-aware Spring ApplicationContext.

270) What is spring boot starter?

Spring Boot Starters are the **dependency descriptors**.

Spring Boot provides a number of **starters** that allow us to add jars in the classpath. Spring Boot built-in **starters** make development easier and rapid.

271) How do you connect your spring boot application to database?

1. Step 1 - Add a Dependency for Your Database Connector to pom. xml. ...
2. Step 2 - Remove H2 Dependency From pom.xml. Or at least make its scope as test. ...
3. Step 3 - Setup Your MySQL Database. ...
4. Step 4 - Configure Your Connection to Your Database. ...
5. Step 5 - Restart

272) What is YAML file? Difference between YAML and properties file?

YAML (.yaml) File: YAML is a configuration language. Languages like Python, Ruby, Java heavily use it for configuring the various properties while developing the applications.

Elastic Search instance and MongoDB database, both of these applications use YAML(.yaml) as their default configuration format.

.properties File: This file extension is used for the configuration application. These are used as the Property Resource Bundles files in technologies like Java, etc.

YAML(.yaml)	.properties
Spec can be found here	It doesn't really actually have a spec. The closest thing it has to a spec is actually the javadoc.
Human Readable (both do quite well in human readability)	Human Readable
Supports key/val, basically map, List and scalar types (int, string etc.)	Supports key/val, but doesn't support values beyond the string
Its usage is quite prevalent in many languages like Python, Ruby, and Java	It is primarily used in java
Hierarchical Structure	Non-Hierarchical Structure
Spring Framework doesn't support @PropertySources with .yaml files	supports @PropertySources with .properties file
If you are using spring profiles, you can have multiple profiles in one single .yaml file	Each profile need one separate .properties file
While retrieving the values from .yaml file we get the value as whatever the respective type (int, string etc.) is in the configuration	While in case of the .properties files we get strings regardless of what the actual value type is in the configuration

273) What is dialect? How it works?

The dialect specifies the type of database used in hibernate so that hibernate generate appropriate type of SQL statements. For connecting any hibernate application with the database, it is required to provide the configuration of SQL dialect.

Syntax of SQL Dialect

```
<property name="dialect"> org.hibernate.dialect.MySQLDialect</property>
```

274) What is foreign key? Difference between primary key and foreign key.

Foreign Key-

The foreign key is used to link one or more than one table together. It is also known as the **referencing** key.

A foreign key matches the primary key field of another table. It means a foreign key field in one table refers to the primary key field of the other table.

Comparison Basis	Primary Key	Foreign Key
Basic	It is used to identify each record into the database table uniquely.	It is used to links two tables together. It means the foreign key in one table refers to the primary key of another table.
NULL	The primary key column value can never be NULL.	The foreign key column can accept a NULL value.
Count	A table can have only one primary key.	A table can have more than one foreign key.
Duplication	The primary key is a unique attribute; therefore, it cannot stores duplicate values in relation.	We can store duplicate values in the foreign key column.
Indexing	The primary key is a clustered index by default, which means it is indexed automatically.	A foreign key is not a clustered index by default. We can make clustered indexes manually.
Deletion	The primary key value can't be removed from the table. If you want to delete it, then make sure the referencing foreign key does not contain its value.	The foreign key value can be removed from the table without bothering that it refers to the primary key of another table.
Insertion	We can insert the values into the primary key column without any limitation, either it present in a foreign key or not.	The value that is not present in the column of a primary key cannot be inserted into the referencing foreign key.
Temporary table	The primary key constraint can be defined on the temporary tables.	A foreign key constraint cannot be defined on the temporary tables.
Relationship	It cannot create a parent-child relationship in a table.	It can make a parent-child relationship in a table.

275) What is unique key?

A unique key in MySQL is a single field or combination of fields that ensure all values going to store into the column will be unique. It means a column cannot stores **duplicate values**. For example, the email addresses and roll numbers of students in the "student_info" table or contact number of employees in the "Employee" table should be unique.

MySQL allows us to use more than one column with UNIQUE constraint in a table. It can accept a **null** value, but MySQL allowed only one null value per column. It ensures the **integrity** of the column or group of columns to store different values into a table.

Needs of Unique Key

- It is useful in preventing the two records from storing identical values into the column.
- It stores only distinct values that maintain the integrity and reliability of the database for accessing the information in an organized way.
- It also works with a foreign key in preserving the uniqueness of a table.
- It can contain null value into the table.

Comparison Basis	Primary Key	Unique Key
Basic	The primary key is used as a unique identifier for each record in the table.	The unique key is also a unique identifier for records when the primary key is not present in the table.
NULL	We cannot store NULL values in the primary key column.	We can store NULL value in the unique key column, but only one NULL is allowed.
Purpose	It enforces entity integrity.	It enforces unique data.
Index	The primary key, by default, creates clustered index.	The unique key, by default, creates a non-clustered index.
Number of Key	Each table supports only one primary key.	A table can have more than one unique key.
Value Modification	We cannot change or delete the primary key values.	We can modify the unique key column values.
Uses	It is used to identify each record in the table.	It prevents storing duplicate entries in a column except for a NULL value.
Syntax	We can create a primary key column in the table using the below syntax: CREATE TABLE Employee (Id INT PRIMARY KEY, name VARCHAR(150), address VARCHAR(250))	We can create a unique key column in the table using the below syntax: CREATE TABLE Person (Id INT UNIQUE, name VARCHAR(150), address VARCHAR(250))

An index is a data structure that allows us to add indexes in the existing table. It enables you to improve the faster retrieval of records on a database table. It creates an **entry** for each value of the indexed columns. We use it to quickly find the record without searching each row in a database table whenever the table is accessed. We can create an index by using one or more **columns** of the table for efficient access to the records.

When a table is created with a primary key or unique key, it automatically creates a special index named **PRIMARY**. We called this index as a clustered index. All indexes other than PRIMARY indexes are known as a non-clustered index or secondary index

277) What are the JSTL tags you have used?

The JSP Standard Tag Library (JSTL) represents a set of tags to simplify the JSP development.

Advantage of JSTL

1. **Fast Development** JSTL provides many tags that simplify the JSP.
2. **Code Reusability** We can use the JSTL tags on various pages.
3. **No need to use scriptlet tag** It avoids the use of scriptlet tag.

JSTL Tags

There JSTL mainly provides five types of tags:

Tag Name	Description
Core tags	The JSTL core tag provide variable support, URL management, flow control, etc. The URL for the core tag is http://java.sun.com/jsp/jstl/core . The prefix of core tag is c .
Function tags	The functions tags provide support for string manipulation and string length. The URL for the functions tags is http://java.sun.com/jsp/jstl/functions and prefix is fn .
Formatting tags	The Formatting tags provide support for message formatting, number and date formatting, etc. The URL for the Formatting tags is http://java.sun.com/jsp/jstl/fmt and prefix is fmt .
XML tags	The XML tags provide flow control, transformation, etc. The URL for the XML tags is http://java.sun.com/jsp/jstl/xml and prefix is x .
SQL tags	The JSTL SQL tags provide SQL support. The URL for the SQL tags is http://java.sun.com/jsp/jstl/sql and prefix is sql .

278) What is error page in JSP?

In JSP, there are two ways to perform exception handling:

1. By **errorPage** and **isErrorPage** attributes of page directive
2. By **<error-page>** element in web.xml fil

This approach is better because you don't need to specify the `errorPage` attribute in each jsp page. Specifying the single entry in the `web.xml` file will handle the exception. In this case, either specify exception-type or error-code with the location element. If you want to handle all the exception, you will have to specify the `java.lang.Exception` in the exception-type element.

279) What is interface? Difference between class and interface?

Interface-

1. It contain public abstract methods and public static final variables by default.
2. We must follow I to C design principle in java. It means every class must be implemented by some interfaces.
3. In company, Team Lead or Manager level people can design the interface then give it to developer for implementing it.
5. Before 1.7, interface does not have any method body.
6. 1.8 Declare the default & static method with body in interface.
7. 1.9 we can define the private methods in interface also.
8. We cannot create the object of interface.
9. In interface, we can just define the method only but implemented that methods into implemented class.
10. Java supports multiple inheritance in the terms of interfaces but not classes.
11. Interface does not have constructor.

Class	Interface
The keyword used to create a class is “class”	The keyword used to create an interface is “interface”
A class can be instantiated i.e, objects of a class can be created.	An Interface cannot be instantiated i.e, objects cannot be created.
Classes does not support multiple inheritance.	Interface supports multiple inheritance.
It can be inherit another class.	It cannot inherit a class.
It can be inherited by another class using the keyword ‘extends’.	It can be inherited by a class by using the keyword ‘implements’ and it can be inherited by an interface using the keyword ‘extends’.
It can contain constructors.	It cannot contain constructors.
It cannot contain abstract methods.	It contains abstract methods only.
Variables and methods in a class can be declared using any access specifier(public, private, default, protected)	All variables and methods in a interface are declared as public.
Variables in a class can be static, final or neither.	All variables are static and final.

280) How we can achieve multiple inheritance using interface?

an interface can extend any no. of interfaces at a time hence java provides support for multiple inheritance through interfaces.

Example:

Interface A { }

Interface B { }

Interface C { } extends A,B

281) What is diamond problem? How can we resolve?

```
class A {  
    public void display()  
    { System.out.println("class A display() method called");  
    }  
    class B extends A {  
        @Override  
        public void display() {  
            System.out.println("class B display() method called");  
        }  
    }  
    class C extends A {  
        @Override  
        public void display() {  
            System.out.println("class C display() method called");  
        }  
    }  
    //not supported in Java  
    public class D extends B,C {  
        public static void main(String args[])  
        {  
            D d = new D();  
            //creates ambiguity which display() method to call  
            d.display();  
        }  
    }
```

In Java multiple inheritance does not allow one class can inherit properties from more than one class. It is known as the **diamond problem**. In the above figure, we find that class D is trying to inherit from class B and class C, that is not allowed in Java.

It is an ambiguity that can rise as a consequence of allowing multiple inheritance. It is a serious problem for other OOPs languages. It is sometimes referred to as the **deadly diamond of death**.

The solution to the diamond problem is **default methods** and **interfaces**. We can achieve multiple inheritance by using these two things.

282) What is profile in Spring Boot? How to select profile?

A *profile* is a set of configuration settings. Spring Boot allows to define profile specific property files in the form of application-{profile}.properties. It automatically loads the properties in an application.properties file for all profiles, and the ones in profile-specific property files only for the specified profile. The keys in the profile-specific property override the ones in the master property file.

283) What is Actuator in spring boot?

Spring Boot Actuator is a sub-project of the Spring Boot Framework. It includes a number of additional features that help us to monitor and manage the Spring Boot application. It contains the actuator endpoints (the place where the resources live). We can use **HTTP** and **JMX** endpoints to manage and monitor the Spring Boot application. If we want to get production-ready features in an application, we should use the **Spring Boot actuator**.

Spring Boot Actuator Features

There are **three** main features of Spring Boot

Actuator:

- **Endpoints**
- **Metrics**
- **Audit**

We can enable actuator by injecting the dependency **spring-boot-starter-actuator** in the pom.xml file

what is repository layer? which methods are present in repository layer?

What is IOC, DI?

Inversion of Control(IoC) is also known as *Dependency injection (DI)*.

284) What are the different types of IOC container? difference between them?

There are two types of IOC container as

BeanFactory-

It is called as core container.

By using this, we can develop the standalone application.

XMLBeanFactory is the implementation class for BeanFactory(I).

To use BeanFactory, we need to create the instance of XmlBeanFactory class.

```
Resource resource= new classPathResource("applicationContext.xml");
```

```
BeanFactory factory= new XmlBeanFactory(resource);
```

The constructor of XmlBeanFactory class receives resource object so we need to pass resource object to create the object of BeanFactory.

ApplicationContext-

It is called as J2EE container.

ClassPathXmlApplicationContext is implementation class for ApplicationContext(I).

To use applicationContext, we need to create the instance of ClassPathXmlApplicationContext as given below-

```
ApplicationContext context= new ClassPathXmlApplicationContext("applicationcontext.xml");
```

The constructor of ClassPathXmlApplicationContext class receives string object so we can pass name of xml file

to create the object of ApplicationContext.

difference between applicationcontext and beanfactory in spring

ApplicationContext Container is advanced than Beanfactory Container...

- 1) BeanFactory Container is basic container, it can only create objects and inject Dependencies. But we can not attach other services like security, transaction, messaging etc. To provide all the services we have to use ApplicationContext Container.
- 2) BeanFactory Container doesn't support the feature of AutoScanning, but ApplicationContext Container supports.
- 3) Beanfactory Container will not create a bean object upto the request time. It means Beanfactory Container loads beans lazily. While ApplicationContext Container creates objects of Singleton bean at the time of loading only. It means there is early loading.
- 4) Beanfactory Container supports only two scopes (singleton & prototype) of the beans. But ApplicationContext Container supports all the beans scope.

285) Explain Spring MVC flow?

Spring MVC-

In MVC, the whole functionality is divided into three parts as-model, view and controller

Model- It can be an object or collection of objects which basically contains the data of the application which consist of POJO.

View- It is used for displaying the information to the user.

Controller- provides information between model and view. It contains the logical part of the application. @Controller annotation is used to mark that class as controller.

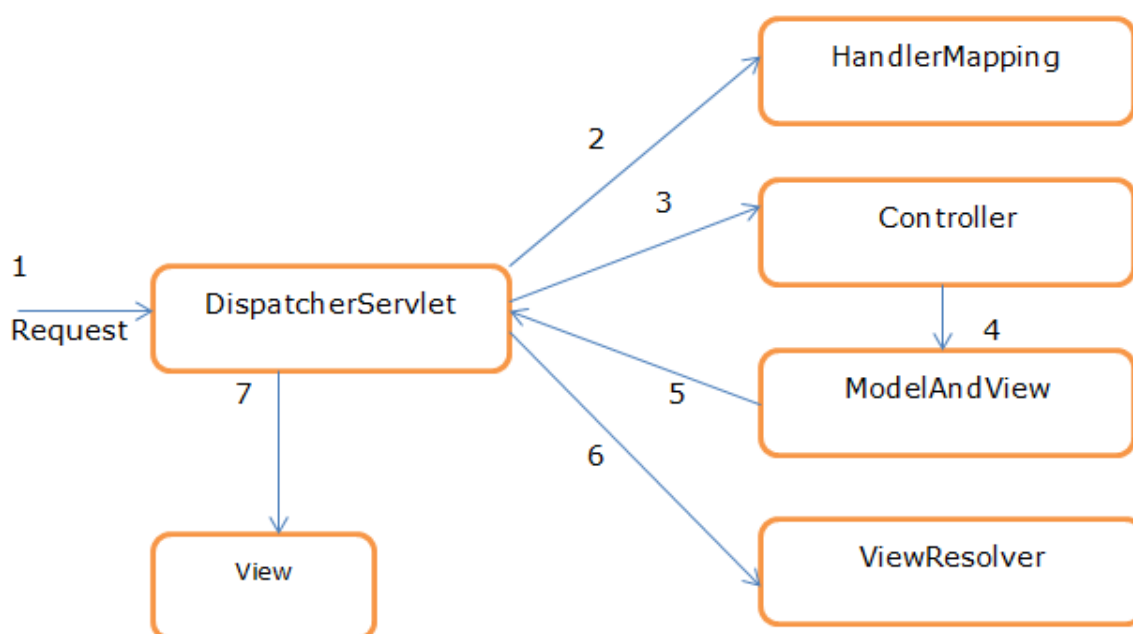


Fig- Spring MVC Flow

Description-

Step 1:

When browser sent the request will be received by DispatcherServlet.

Step 2:

DispatcherServlet will take the help of HandlerMapping and get to know the Controller class name associated with the given request.

Step 3:

So request transfer to the Controller, and then controller will process the request by executing appropriate methods and returns ModelAndView object (contains *Model* data and *View* name) back to the DispatcherServlet.

Step 4:

Now DispatcherServlet send the model object to the ViewResolver to get the actual view page.

Step 5:

Finally DispatcherServlet will pass the *Model* object to the *View* page to display the result.

286) What is view Resolver? (need to check ans)

The **InternalResourceViewResolver** is a class which is used to resolve internal view in Spring MVC. Here, you can define the properties like prefix and suffix where prefix contains the location of view page and suffix contains the extension of view page. For example:-

1. `<bean id="viewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver">`
2. `<property name="prefix" value="/WEB-INF/jsp/"></property>`
3. `<property name="suffix" value=".jsp"></property>`
4. `</bean>`

287) What is Dispatcher servlet? ? (need to check ans)

In Spring MVC all incoming requests go through a single servlet is called **Dispatcher Servlet (front controller)**. The front controller is a design pattern in web application development. A single servlet receives all the request and transfers them to all other components of the application.

288) Difference between spring MVC & Spring boot?

Spring	Spring Boot
Spring Framework is a widely used Java EE framework for building applications.	Spring Boot Framework is widely used to develop REST APIs .
It aims to simplify Java EE development that makes developers more productive.	It aims to shorten the code length and provide the easiest way to develop Web Applications .
The primary feature of the Spring Framework is dependency injection .	The primary feature of Spring Boot is Autoconfiguration . It automatically configures the classes based on the requirement.

It helps to make things simpler by allowing us to develop loosely coupled applications.	It helps to create a stand-alone application with less configuration.
The developer writes a lot of code (boilerplate code) to do the minimal task.	It reduces boilerplate code.
To test the Spring project, we need to set up the sever explicitly.	Spring Boot offers embedded server such as Jetty and Tomcat , etc.
It does not provide support for an in-memory database.	It offers several plugins for working with an embedded and in-memory database such as H2 .
Developers manually define dependencies for the Spring project in pom.xml .	Spring Boot comes with the concept of starter in pom.xml file that internally takes care of downloading the dependencies JARs based on Spring Boot Requirement.

289) How we can do one to many mapping in program?

- 1) Create the Persistent class. This persistent class defines properties of the class including List. ...
- 2) Add project information and configuration in pom. xml file. ...
- 3) Create the configuration file. ...
- 4) Create the class to store the data.

290) What is cfg file? what are you writing in that?

A file with the CFG extension is a configuration file used by several programs to store specific configurations of their functions and/or activities. Some configuration files are plain text files, but others can be stored in a specific program format like JSON or XML, or even in binary form.

291) What is bean? What are the tags? ? (need to check ans)

A bean is **an object that is instantiated, assembled, and otherwise managed by a Spring IoC container**. These beans are created with the configuration metadata that you supply to the container.

Tags::

Name/id

Class

Scope

constructor-arg

properties

autowiring mode

lazy-init (lazy-initialization mode)

init-method (initialization method)

destroy-method (destruction method)

292) What is date and time Api and explain? ? (need to check ans)

Java has introduced a new Date and Time API since Java 8. The java.time package contains Java 8 Date and Time

classes.

Java.time.LocalDate.class

Java.time.LocalTime.class

Java.time.LocalDateTime.class

Java.time.MonthDay.class etc.....

We can format date and time in java by the use of following classes:

Java.time.DateFormat.class

Java.time.SimpleDateFormat.class

293) @SpringBootApplication: It is a combination of three annotations **@EnableAutoConfiguration**, **@ComponentScan**, and **@Configuration**.

@Qualifier annotation

There may be a situation when you create more than one bean of the same type and want to wire only one of them with a property. In such cases, you can use the **@Qualifier** annotation along with **@Autowired** to remove the confusion by specifying which exact bean will be wired.

- **@ResponseBody:** It binds the method return value to the response body. It tells the Spring Boot Framework to serialize a return an object into JSON and XML format.
- **@PathVariable:** It is used to extract the values from the URI. It is most suitable for the RESTful web service, where the URL contains a path variable. We can define multiple **@PathVariable** in a method.

@Controller: The **@Controller** is a class-level annotation. It is a specialization of **@Component**. It marks a class as a web request handler. It is often used to serve web pages. By default, it returns a string that indicates which route to redirect. It is mostly used with **@RequestMapping** annotation.

- **@RestController:** It can be considered as a combination of **@Controller** and **@ResponseBody** annotations. The **@RestController** annotation is itself annotated with the **@ResponseBody** annotation. It eliminates the need for annotating each method with **@ResponseBody**.

294) What is function?

how do we ensure that it will return only one object?

How to map class with table id in the database?

what is repository layer? which methods are present in repository layer?

Qus no 22,23,27 ans need to be check ?

Q- 295) @ControllerAdvice use?

Ans -> **Since spring 3.2** , @ControllerAdvice is a **specialization** of the @Component annotation which **allows to handle exceptions across the whole application in one global handling component**. It can be viewed as an interceptor of exceptions thrown by methods annotated with @RequestMapping and similar. Classes annotated with @ControllerAdvice can be declared explicitly as Spring beans or auto-detected via classpath scanning.

Q- 296) @Required annotation

Ans -> It applies to the bean setter method. It indicates that the annotated bean must be populated at configuration time with the required property; else it throws an exception BeanInitializationException.

Q- 297) How to change port number from tomcat

Ans -> In order to change the default port in Apache Tomcat, you need to:

1. Stop Apache Tomcat service
2. Go to your Apache Tomcat folder (for example C:\Program Files\Apache Software Foundation\Tomcat 7.0) and find file server.xml under \conf\ folder.
3. Modify the Connector port value from 8080" to the one you want to assign to your web server.

Example:

From: <connector port="8080" protocol="HTTP/1.1"

To: <connector port="8085" protocol="HTTP/1.1"

4. Save the file
5. Restart the Apache Tomcat service

Q- 298) How to do configuration

in spring boot

6. Ans ->

1. We can perform XML based configuration. We can have spring boot starter

configured as follows ->

- **spring-boot-starter-parent** : Parent POM for dependency management.
- **spring-boot-starter-web**: Starter for building web, REST applications. It uses tomcat server as default embedded server.
- **spring-boot-devtools** : It provides developer tools. These tools are helpful in application development mode. One of the features of developer tool is automatic restart of the server for any change in code.
- **spring-boot-maven-plugin** : It is used to create executable JAR of the application.

2. **We can perform Annotation based configuration : -**

@SpringBootApplication :- We use this annotation to **mark the main class of a Spring Boot application**:

We can place the annotations on *@Configuration* classes or *@Bean* methods. **Further conditional annotations** such as *@ConditionalOnClass* , *@ConditionalOnMissingClass* , *@ConditionalOnProperty* , *@ConditionalOnResource* and so on.

Q- 299) Difference between static and final

Ans ->

BASIS FOR COMPARISON	STATIC	FINAL
Applicable	Static keyword is applicable to nested static class, variables, methods and block.	Final keyword is applicable to class, methods and variables.
Initialization	It is not compulsory to initialize the static variable at the time of its declaration.	It is compulsory to initialize the final variable at the time of its declaration.
Modification	The static variable can be reinitialized.	The final variable cannot be reinitialized.
Methods	Static methods can only access the static members of the class, and can only be called by other static methods.	Final methods cannot be inherited.
Class	Static class's object cannot be created, and it only contains static members only.	A final class cannot be inherited by any class.
Block	Static block is used to initialize the static variables.	Final keyword supports no such block.

Q- 300) Can we overload static method

Ans -> **Yes**, you can overload a static method in Java

Q- 301) What is string constant pool

Ans -> String pool is nothing but a **storage area in Java heap where string literals stores**. It is also known as **String Intern Pool or String Constant Pool**. It is just like object allocation. By default, it is empty and **privately maintained by the Java String class**.

Q- 301) Can we use static method in interface

Ans -> Yes. It can be used. It is one of the features of Java 8 to have static & default methods defined in an interface.

Q- 302) Parent class for hashmap

Ans -> It is AbstractMap class.

Q- 303) Difference between throws and throw

Ans ->

throw	throws
throw keyword is used to throw an exception explicitly.	throws keyword is used to declare one or more exceptions, separated by commas.
Only single exception is thrown by using throw.	Multiple exceptions can be thrown by using throws.
throw keyword is used within the method.	throws keyword is used with the method signature.
Syntax wise throw keyword is followed by the instance variable.	Syntax wise throws keyword is followed by exception class names.
Checked exception cannot be propagated using throw only. Unchecked exception can be propagated using throw.	For the propagation checked exception must use throws keyword followed by specific exception class name.

Q- 304) Difference between wait and sleep method

Ans -> **Sleep():** This Method is used to pause the execution of current thread for a specified time in Milliseconds. Here, Thread does not lose its ownership of the monitor and resume's its execution

Wait(): This method is defined in object class. It tells the calling thread (a.k.a Current Thread) to wait until another thread invoke's the notify() or notifyAll() method for this object, The thread waits until it reobtains the ownership of the monitor and Resume's Execution.

Wait()	Sleep()
Wait() method belongs to Object class.	Sleep() method belongs to Thread class.
Wait() method releases lock during Synchronization.	Sleep() method does not release the lock on object during Synchronization.
Wait() should be called only from Synchronized context.	There is no need to call sleep() from Synchronized context.
Wait() is not a static method.	Sleep() is a static method.
Wait() Has Three Overloaded Methods: <ul style="list-style-type: none"> • wait() • wait(long timeout) • wait(long timeout, int nanos) 	Sleep() Has Two Overloaded Methods: <ul style="list-style-type: none"> • sleep(long millis) millis: milliseconds • sleep(long millis, int nanos) nanos: Nanoseconds
public final void wait(long timeout)	public static void sleep(long millis) throws InterruptedException

Q- 305) What is JDBC steps to create connection

Ans ->

1. Import JDBC packages.
2. Load and register the JDBC driver.
3. Open a connection to the database.
4. Create a statement object to perform a query.
5. Execute the statement object and return a query resultset.
6. Process the resultset.
7. Close the resultset and statement objects.
8. Close the connection.

Q- 306) Difference between statement and prepared statement in java

Ans ->

Statement	PreparedStatement
It is used when SQL query is to be executed only once.	It is used when SQL query is to be executed multiple times.
You cannot pass parameters at runtime.	You can pass parameters at runtime.
Used for CREATE, ALTER, DROP statements.	Used for the queries which are to be executed multiple times.
Performance is very low.	Performance is better than Statement.
It is base interface.	It extends statement interface.
Used to execute normal SQL queries.	Used to execute dynamic SQL queries.
We cannot use statement for reading binary data.	We can use PreparedStatement for reading binary data.
It is used for DDL statements.	It is used for any SQL Query.
We cannot use statement for writing binary data.	We can use PreparedStatement for writing binary data.
No binary protocol is used for communication.	Binary protocol is used for communication.

Q- 307) Do you know about callable statements?

Ans -> It is CallableStatement Interface in Java. CallableStatement interface is used to call the stored procedures and functions.

We can have business logic on the database by the use of stored procedures and functions that will make the performance better because these are precompiled.

Q- 308) Difference between web server and application server

Ans ->

S.NO	Web Server	Application Server
1.	Web server encompasses web container only.	While application server encompasses Web container as well as EJB container.
2.	Web server is useful or fitted for static content.	Whereas application server is fitted for dynamic content.
3.	Web server consumes or utilizes less resources.	While application server utilize more resources.
4.	Web servers arrange the run environment for web applications.	While application servers arrange the run environment for enterprises applications.
5.	In web servers, multithreading is not supported.	While in application server, multithreading is supported.
6.	Web server's capacity is lower than application server.	While application server's capacity is higher than web server.
7.	In web server, HTML and HTTP protocols are used.	While in this, GUI as well as HTTP and RPC/RMI protocols are used.

Q- 309) Get and post method difference

Ans ->

	GET	POST
BACK button/Reload	Harmless	Data will be re-submitted (the browser should alert the user that the data are about to be re-submitted)
Bookmarked	Can be bookmarked	Cannot be bookmarked
Cached	Can be cached	Not cached
Encoding type	application/x-www-form-urlencoded	application/x-www-form-urlencoded or multipart/form-data. Use multipart encoding

		for binary data
History	Parameters remain in browser history	Parameters are not saved in browser history
Restrictions on data length	Yes, when sending data, the GET method adds the data to the URL; and the length of a URL is limited (maximum URL length is 2048 characters)	No restrictions
Restrictions on data type	Only ASCII characters allowed	No restrictions. Binary data is also allowed
Security	GET is less secure compared to POST because data sent is part of the URL Never use GET when sending passwords or other sensitive information!	POST is a little safer than GET because the parameters are not stored in browser history or in web server logs
Visibility	Data is visible to everyone in the URL	Data is not displayed in the URL

Q- 310) Servlet life cycle

Ans -> The web container maintains the life cycle of a servlet instance.

1. Servlet class is loaded.
2. Servlet instance is created.
3. init method is invoked.
4. service method is invoked.
5. destroy method is invoked.

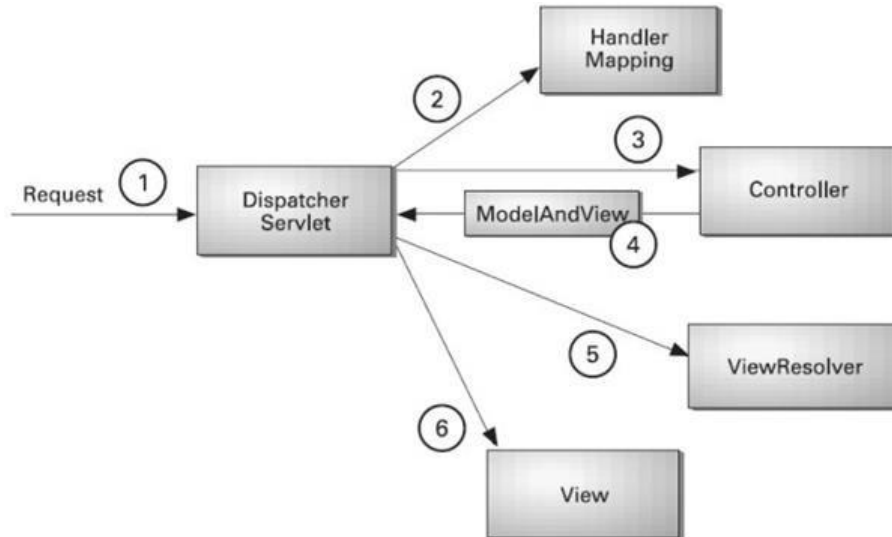
Q- 311) How to create singleton class

Ans -> To create a singleton class, we must follow the steps, given below:

1. Ensure that only one instance of the class exists.
2. Provide global access to that instance by:
 - i. Declaring all constructors of the class to be private.

- ii. Providing a static method that returns a reference to the instance. The lazy initialization concept is used to write the static methods.
- iii. The instance is stored as a private static variable.

Q- 312) Spring mvc flow



- ✓ As displayed in the figure, all the incoming request is intercepted by the DispatcherServlet that works as the front controller.
- ✓ The DispatcherServlet gets an entry of handler mapping from the XML file and forwards the request to the controller.
- ✓ The controller returns an object of ModelAndView.
- ✓ The DispatcherServlet checks the entry of view resolver in the XML file and invokes the specified view component.

Q- 313) Types of view resolver

Ans -> There are **THREE** types viz.-

1. **InternalResourceViewResolver** - is used to resolve the provided URI to actual URI. The InternalResourceViewResolver allows mapping webpages with requests.
2. **XmlViewResolver** - is used to resolve the view names using view beans defined in xml file.
3. **ResourceBundleViewResolver** - is used to resolve the view names using view beans defined in the properties file.

Q- 314) Scope of beans

Ans ->

Sr. No.	Scope & Description
1	singleton This scopes the bean definition to a single instance per Spring IoC container (default).
2	prototype This scopes a single bean definition to have any number of object instances.
3	request This scopes a bean definition to an HTTP request. Only valid in the context of a web-aware Spring ApplicationContext.
4	session This scopes a bean definition to an HTTP session. Only valid in the context of a web-aware Spring ApplicationContext.
5	global-session This scopes a bean definition to a global HTTP session. Only valid in the context of a web-aware Spring ApplicationContext.

Q- 315) How to change scope in spring and spring boot

Ans -> In Spring,

1. Scope can be defined using spring bean @Scope annotation.

e.g. @Component

```
@Scope("prototype") public class  
BeanClass {  
}
```

2. Scope can also be defined using XML configuration.

e. g. <bean id="beanId" class="com.springjava.BeanClass" scope="prototype" />

Q316) 23 Solid principle

Ans -> In Java, SOLID principles are an object-oriented approach that are applied to software structure design.

The word SOLID acronym for:

- Single Responsibility Principle (SRP)

- Open-Closed Principle (OCP)
- Liskov Substitution Principle (LSP)
- Interface Segregation Principle (ISP)
- Dependency Inversion Principle (DIP)

Q- 317) Union Operator in database

Ans -> The SQL UNION operator is used to combine the result sets of 2 or more SELECT statements. It removes duplicate rows between the various SELECT statements.

Each SELECT statement within the UNION must have the same number of fields in the result sets with similar data types.

- UNION removes duplicate rows.
- UNION ALL does not remove duplicate rows.

Q- 318) Test unit development

Ans -> **UNIT TESTING** is a type of software testing where individual units or components of a software are tested. The purpose is to validate that each unit of the software code performs as expected. Unit Testing is done during the development (coding phase) of an application by the developers. Unit Tests isolate a section of code and verify its correctness. A unit may be an individual function, method, procedure, module, or object.

Q- 319) @Autowired annotation

Ans -> In Spring, you can use @Autowired annotation to auto-wire bean on the setter method, constructor, or a field. Moreover, it can autowire the property in a particular bean.

Q- 320) If you don't use @Autowired annotation then how you do that

Ans -> We can use XML configuration for Autowiring. The XML-configuration-based autowiring functionality has five modes – **no, byName, byType, constructor, and autodetect**. The **default mode is no**.

Q- 321) @componentScan annotation

Ans -> The @ComponentScan annotation is used with the @Configuration annotation to tell the Spring packages to scan for annotated components.

Q- 322) Sql basic query

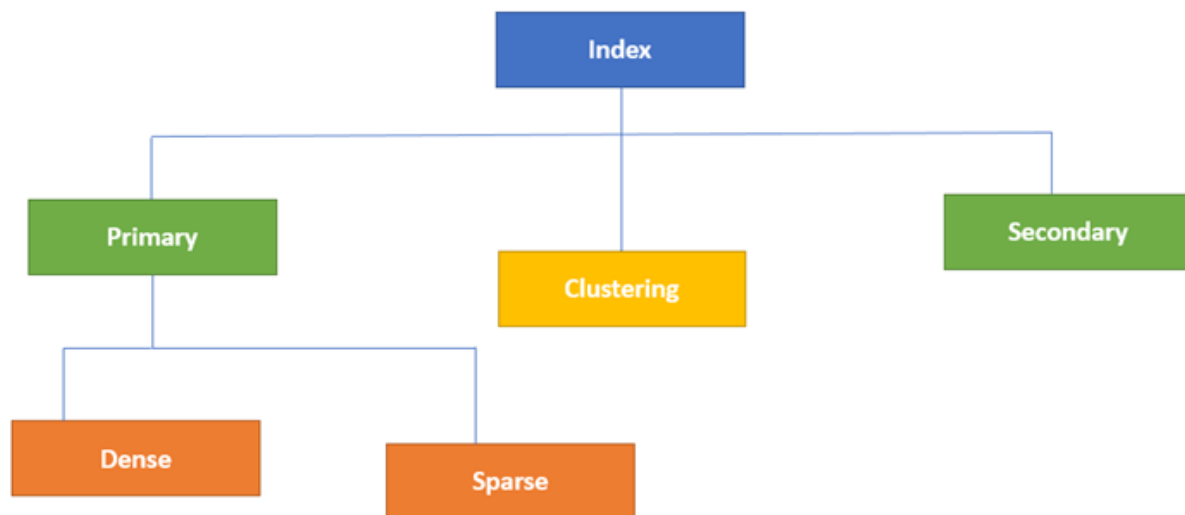
- SELECT - extracts data from a database
- UPDATE - updates data in a database
- DELETE - deletes data from a database
- INSERT INTO - inserts new data into a database
- CREATE DATABASE - creates a new database
- ALTER DATABASE - modifies a database

- CREATE TABLE - creates a new table
- ALTER TABLE - modifies a table
- DROP TABLE - deletes a table
- CREATE INDEX - creates an index (search key)
- DROP INDEX - deletes an index

Q- 323) Soap vs rest : Ans ->

No.	SOAP	REST
1)	SOAP is a protocol .	REST is an architectural style .
2)	SOAP stands for Simple Object Access Protocol .	REST stands for REpresentational State Transfer .
3)	SOAP can't use REST because it is a protocol.	REST can use SOAP web services because it is a concept and can use any protocol like HTTP, SOAP.
4)	SOAP uses services interfaces to expose the business logic .	REST uses URI to expose business logic .
5)	JAX-WS is the java API for SOAP web services.	JAX-RS is the java API for RESTful web services.
6)	SOAP defines standards to be strictly followed.	REST does not define too much standards like SOAP.
7)	SOAP requires more bandwidth and resource than REST.	REST requires less bandwidth and resource than SOAP.
8)	SOAP defines its own security .	RESTful web services inherits security measures from the underlying transport.
9)	SOAP permits XML data format only.	REST permits different data format such as Plain text, HTML, XML, JSON etc.
10)	SOAP is less preferred than REST.	REST more preferred than SOAP.

Q- 324) Different types of indexes into database



Indexing is a data structure technique to efficiently retrieve records from the database files based on some attributes on which the indexing has been done.

- **Primary Index**
- **Secondary Index**
- **Clustering Index**

Primary Indexing is of two subtypes –

- a. Dense Index
- b. Sparse Index

Q- 325) Cartesian product in sql

Ans -> **CARTESIAN JOIN:** The CARTESIAN JOIN is also known as **CROSS JOIN**. In a CARTESIAN JOIN there is a join for each row of one table to every row of another table. This usually happens when the matching column or WHERE condition is not specified.

- ☐ **In the absence of a WHERE condition the CARTESIAN JOIN will behave like a CARTESIAN PRODUCT . i.e., the number of rows in the result-set is the product of the number of rows of the two tables.**
- ☐ In the presence of WHERE condition this JOIN will function like a INNER JOIN.
- ☐ Generally speaking, Cross join is similar to an inner join where the join-condition will always evaluate to True

Q- 326) Savepoint and trigger in sql

Ans -> **SAVEPOINT:** creates points within the groups of transactions in which to ROLLBACK. A SAVEPOINT is a point in a transaction in which you can roll the transaction back to a certain point without rolling back the entire transaction.

Trigger: 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.

Q- 34 Singleton and factory design pattern

Ans -> **Singleton Pattern** says that just “define a class that has only one instance and provides a global point of access to it”. A class must ensure that only single instance should be created and single object can be used by all other classes.

Factory Pattern says that just define an interface or abstract class for creating an object but let the subclasses decide which class to instantiate. Subclasses are responsible to create the instance of the class.

Q- 327) Did you use JSF, (answer is NO)

JavaServer Faces :- It is a server side component based user interface framework. It is used to develop web applications. It provides a well-defined programming model and consists of rich API and tag libraries.

Q- 328) Difference between arraylist and list

Base of Comparison	List	ArrayList
General	It is an interface.	It is class.
Work	It creates a list of objects that can be accessed by the individual index number.	It creates a dynamic array that can be expanded when needed.
Implementation	List <data-type> list1= new ArrayList();	ArrayList myList = new ArrayList();
Extend/Implement	It extends the collection framework.	It extends AbstractList class and implements the List interface.
Base Package	Java.util	Java.util
Namespace	System.Collection.Generic	System.Collection
Performance	It provides faster manipulation of objects.	It provides slow manipulation on objects compared to List.
Instantiation	It cannot be instantiated.	It can be instantiate

Q- 329) Iterator vs collection

- ☐ Iterator can only move to **next()** element or **remove()** an element. However Collection can **add()**, iterate, **remove()** or **clear()** the elements of the collection.
- ☐ Iterator provides a better speed than Collections, as the Iterator interface has limited number of operations.

Q- 330) Array vs arraylist

Ans ->

Basis	Array	ArrayList
Definition	An array is a dynamically-created object. It serves as a container that holds the constant number of values of the same type. It has a contiguous memory location.	The ArrayList is a class of Java Collections framework. It contains popular classes like Vector , HashTable , and HashMap .
Static/ Dynamic	Array is static in size.	ArrayList is dynamic in size.
Resizable	An array is a fixed-length data structure.	ArrayList is a variable-length data structure. It can be resized itself when needed.
Initialization	It is mandatory to provide the size of an array while initializing it directly or indirectly.	We can create an instance of ArrayList without specifying its size. Java creates ArrayList of default size.
Performance	It performs fast in comparison to ArrayList because of fixed size.	ArrayList is internally backed by the array in Java. The resize operation in ArrayList slows down the performance.
Primitive/ Generic type	An array can store both objects and primitives type.	We cannot store primitive type in ArrayList. It automatically converts primitive type to object.
Iterating Values	We use for loop or for each loop to iterate over an array.	We use an iterator to iterate over ArrayList.
Type-Safety	We cannot use generics along with array because it is not a convertible type of array.	ArrayList allows us to store only generic/ type, that's why it is type-safe .
Length	Array provides a length variable which denotes the length of an array.	ArrayList provides the size() method to determine the size of ArrayList.
Adding Elements	We can add elements in an array by using the assignment operator.	Java provides the add() method to add elements in the ArrayList.
Single/ Multi- Dimensional	Array can be multi-dimensional .	ArrayList is always single-dimensional .

Q- 331) Data access object layer

Ans -> Data Access Object Pattern or DAO pattern is used to separate low level data accessing API or operations from high level business services. Following are the participants in Data Access Object Pattern.

- ❑ **Data Access Object Interface** - This interface defines the standard operations to be performed on a model object(s).
- ❑ **Data Access Object concrete class** - This class implements above interface. This class is responsible to get data from a data source which can be database / xml or any other storage mechanism.
- ❑ **Model Object or Value Object** - This object is simple POJO containing get/set methods to store data retrieved using DAO class.

Q- 332) What is hibernate

Ans -> Hibernate is a Java framework that simplifies the development of Java application to interact with the database. It is an open source, lightweight, ORM (Object Relational Mapping) tool.

Hibernate implements the specifications of JPA (Java Persistence API) for data persistence.

Q-333) JDBC vs Hibernate

1. As Hibernate is set of Objects , you don't need to learn SQL language. You can treat TABLE as a Object . Only Java knowledge is need.

In case of JDBC you need to learn SQL.

2. Hibernate supports Query cache and It will provide the statistics about your query and database status.

JDBC Not provides any statistics.

3. Hibernate is data base independent, your code will work for all ORACLE, MySQL , SQLServer etc.

In case of JDBC query, it must be data base specific.

4. You will get benefit of Cache. Hibernate support two level of cache. First level and 2nd level. So you can store your data into Cache for better performance.

In case of JDBC you need to implement your java cache .

5. No need to create any connection pool in case of Hibernate. You can use c3p0.

In case of JDBC you need to write your own connection pool

6. Development fast in case of Hibernate because you don't need to write queries

7. You can load your objects on start up using lazy=false in case of Hibernate.

JDBC Don't have such support.

8. Hibernate Supports automatic versioning of rows but JDBC Not.

Q- 334) Internal working of hashmap

If you want to represent group of objects as key-value pair then you should go for map.

How hashCode() and equals() methods work into HashMap?

hashCode() method

hashCode method is used to get the hashcode of every objects.

equals() method

equals method is used to check that 2 objects are equal or not. This method is provided by Object class. HashMap uses equals() to compare the key whether they are equal or not. If equals() method return true, they are equal otherwise not equal.

How to calculate index internally?

Put(K k, V v) //Here put is the method having key and value pair

hash(k) -> hash(jeevan) -> 23560520

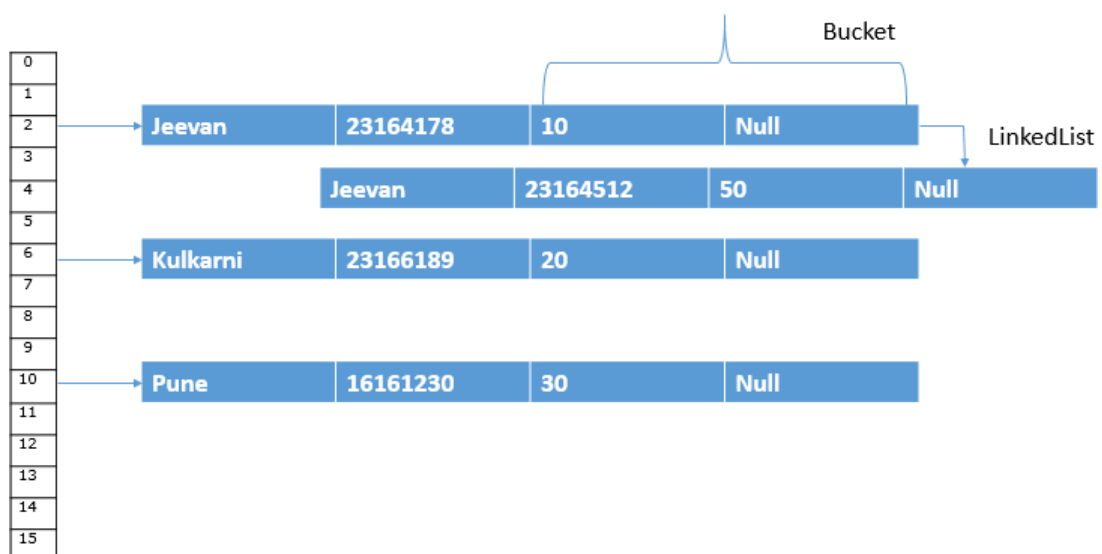
index = hash & (n-1) here index=2

so jeevan will be stored at 2nd position.

Next time again I get the 2nd index or position then what will happen?

It will store it at 2nd and uses the linked list as shown in fig.

After than I get the 6th index so kulkarni will be stored at 6th position then next time get the index 10th then pune will be placed at 10th position.



How get method works here?

Now let's try some get method to get a value. get(K key) method is used to get a value by its key. If you don't know the key then it is not possible to fetch a value.

Q- 335) Microservices basics

It is a kind of architectural style which structures an application as a collection of Services

Advantage-

1. Simple to deploy
2. Simple to understand
3. Reusability across business
4. Faster defect isolation
5. Minimized risk of change
6. They are loosely coupled.
7. Independently deployable.

In Case of Monolithic we consider whole project as a single code-base & hence we deploy the whole app as a single war file.

But when it comes to Microservices application, each of the service is considered as a separate code-base & hence we deploy a separate war file for each code-base (service).

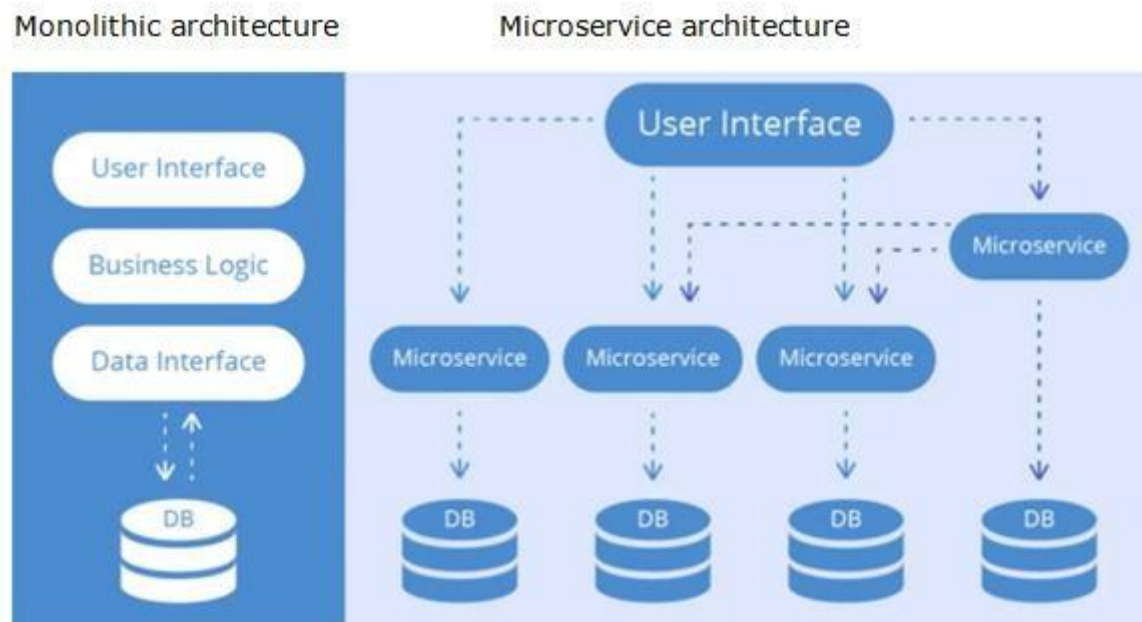


Fig- Microservice architecture

Note

1. We need to create a separate spring boot project for every microservice in the project
2. We need the separate tomcat instances for running each of the application independently.
3. For microservices to communicate with each other , we will use Spring-cloud [EurekaServer].
4. EurekaServer can be considered as a Discovery server.
5. For Microservices to communicate with each other, we have to register them on this EurekaServer.
6. Here each Microservice will be the Client for the EurekaServer.

Q- 336) @SpringBootApplication annotation use ?

- **@SpringBootApplication:** It is a combination of following three annotations -
- **@EnableAutoConfiguration - , @ComponentScan, and @Configuration.**
- **@EnableAutoConfiguration:** It auto-configures the bean that is present in the classpath and configures it to run the methods..
- **@ComponentScan:** It is used when we want to scan a package for beans. It is used with the annotation @Configuration. We can also specify the base packages to scan for Spring Components.
- **@Configuration:** It is a class-level annotation. The class annotated with @Configuration used by Spring Containers as a source of bean definitions.

Q- 337) What dependancies you have used in

your project Q- 46 What is devtool

dependency

Spring Boot 1.3 provides another module called Spring Boot DevTools. DevTools stands for **Developer Tool**. The aim of the module is to try and improve the development time while working with the Spring Boot application. Spring Boot DevTools pick up the changes and restart the application.

Spring Boot DevTools provides the following features:

- **Property Defaults** - When we use the spring-boot-devtools module, we are not required to set properties. During the development caching for Thymeleaf, Freemarker, Groovy Templates are automatically disabled.
- **Automatic Restart** - Auto-restart means reloading of Java classes and configure it at the server-side. After the server-side changes, it deployed dynamically, server restarts happen, and load the modified code.
- **LiveReload** - The Spring Boot DevTools module includes an embedded server called **LiveReload**. It allows the application to automatically trigger a browser refresh whenever we make changes in the resources. It is also known as **auto-refresh**.
- **Remote Debug Tunneling** - Spring Boot can tunnel JDWP (Java Debug Wire Protocol) over HTTP directly to the application. It can even work application deployment to Internet Cloud providers that only expose port 80 and 443.
- **Remote Update and Restart** - it supports remote application **updates** and **restarts**. It monitors local classpath for file changes and pushes them to a remote server, which is then restarted. We can also use this feature in combination with LiveReload.

Q- 338) What is immutable, what is immutable classes in java

Immutable class means that once an object is created, we cannot change its content.

In Java, all the wrapper classes (like Integer, Boolean, Byte, Short) and String class is immutable.

We can create our own immutable class as well. ... The class must be declared as final (So that child classes can't be created)

Q- 339) Collection classes and interfaces, which you have used in your project
Q- 49 Internal working of hashmap - repeated

Q- 340) Why map is not under collection

Because they are of an incompatible type. List Set and Queue are a collection of similar kind of objects but just values where as a Map is a collection of Key and Value pairs.

List, Set and Queue have add as a function which takes a value as a param to add an element whereas Map has put as a function which takes a key and a value as params to add a key value pair.

List, Set and Queue provide iterate functionality over the value whereas Map has keys to iterate over which is ultimately a Set and Values as Collection.

Q- 51 Why we use hibernate over JDBC you define pojo class

- ❑ In JDBC, if we open a database connection we need to write in try, and if any exceptions occurred catch block will take care about it, and finally used to close the connections.
- ❑ We must close the connection, or we may get a chance to get connections error message.
- ❑ Actually if we didn't close the connection in the finally block, then jdbc doesn't responsible to close that connection.
- ❑ In JDBC we need to write Sql commands in various places, after the program has created if the table structure is modified then the JDBC program doesn't work, again we need to modify and compile and re-deploy required, which is tedious.
- ❑ To overcome above drawbacks we should go for Hibernate framework.

POJO Class

Java classes whose objects or instances will be stored in database tables are called persistent classes in Hibernate. Hibernate works best if these classes follow some simple rules, also known as the **Plain Old Java Object (POJO)** programming model.

There are following main rules of persistent classes, however, none of these rules are hard requirements –

- ❑ All Java classes that will be persisted need a default constructor.
- ❑ All classes should contain an ID in order to allow easy identification of your objects within Hibernate and the database. This property maps to the primary key column of a database table.
- ❑ All attributes that will be persisted should be declared private and have **getter** and **setter** methods defined in the JavaBean style.
- ❑ A central feature of Hibernate, proxies, depends upon the persistent class being either non-final, or the implementation of an interface that declares all public methods.
- ❑ The POJO name is used to emphasize that a given object is an ordinary Java Object

Q- 341) In many-to-many mapping, how many tables are created. – 3 tables

Q- 342) What is rest API

Suppose I have one online website called as fresherworlds.com in which I want to show the tour packages of different countries then what options I have?

There is only one option call the API of tour packages into my site.

That's why Rest API's comes into picture.

Or

If your application supports third party application in that case we can use REST API.

What is REST?

The term REST stands for **RE**presentational **S**tate **T**ransfer. It is an architectural style that defines a set of rules in order to create Web Services.

In a client-server communication, REST suggests to create an object of the data requested by the client and send the values of the object in response to the user.

There are four types of method as

- Get-It is used to read resource
- Post-It is used to create new resource.
- Put-It is generally used to update resource
- Delete-it is used to delete source

Pre-requisites-

1. Create the spring boot project.
2. Pom.xml
3. Postman tool

Q- 343) Scope of maven

Maven defines the behaviour for each scope as following –

- **compile** This is the default scope, used if none is specified. Compile dependencies are available in all classpaths of a project. Furthermore, those dependencies are propagated to dependent projects.
- **provided** This is much like compile, but indicates you expect the JDK or a container to provide the dependency at runtime. For example, when building a web application for the Java Enterprise Edition, you would set the dependency on the Servlet API and related Java EE APIs to scope provided because the web container provides those classes. This scope is only available on the compilation and test classpath, and is not transitive.
- **runtime** This scope indicates that the dependency is not required for compilation, but is for execution. It is in the runtime and test classpaths, but not the compile classpath.
- **test** This scope indicates that the dependency is not required for normal use of the application, and is only available for the test compilation and execution phases. This scope is not transitive.
- **system** This scope is similar to provided except that you have to provide the JAR which contains it explicitly. The artifact is always available and is not looked up in a repository.
- **import** (only available in Maven 2.0.9 or later) This scope is only supported on a dependency of type pom in the section. It indicates the dependency to be replaced with the effective list of dependencies in the specified POM's section. Since they

are replaced, dependencies with a scope of import do not actually participate in limiting the transitivity of a dependency.

Q- 344) What is SDLC

SDLC (Software Development Life Cycle) is a process followed for a software project, within a software organisation. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

Below are the list of different SDLC phases as

1. Requirement analysis- In this phase, details requirements of software are to be taken by the client.
2. System design- It decides the programming language such as java or .net, database like MYSQL or oracle, then front end tool, etc.
3. Implementation- In this, actual project development is to be started.
4. System testing- It performs the testing of software, also check whether the software is designed as per client requirement or not.
5. System deployment- It will deploy the application on respective environments e.g. tomcat server.
6. System maintenance- Once you deploy the application on server, later point you might be add or changes the requirement as per client request.

Q- 345) What is static?

Static-

It is used for memory management.

It can be applied to variable, method, inner class and static block.

It means single copy storage.

Static variable-

A variable which is defined with static keyword called as “static variable.”

It is used to refer the common property of all the objects.

Static variables get loaded into memory at the time of class loading.

Note- Static members get loaded into memory as soon as (StaticDemo sd1) read this line.

Non static members gets loaded into memory after reading new StaticDemo()- this line.

Static method-

If you define any method with static keyword then it is called as static method.

It belongs to class rather than object of class.

It loads into memory before object creation.

It can access only static data member only.

Note: - Main method is static method.

Static block-

It is group of statements that are executed when class is loading into memory by Classloader.

It is widely used to create the static resource.

We cannot access non-static variable into static block.

It is always executed first.

Q- 346) Have you use static block in your project

Q- 347) How to make thread as safe

There are 4 ways to achieve thread safety in Java. The ways are as follows –

1. By using Synchronization - Synchronization is the process of allowing only one thread at a time to complete the particular task. It means when multiple threads executing simultaneously, and want to access the same resource at the same time, then the problem of inconsistency will occur. so synchronization is used to resolve inconsistency problem by allowing only one thread at a time. Synchronization uses a *synchronized keyword*. Synchronized is the modifier that creates a block of code known as a critical section.
2. By using Volatile Keyword -
A volatile keyword is a field modifier that ensures that the object can be used by multiple threads at the same time without having any problem. volatile is one good way of ensuring that the Java program is thread-safe. a volatile keyword can be used as an alternative way of achieving Thread Safety in Java.
3. By using Atomic Variable - Using an atomic variable is another way to achieve thread-safety in java. When variables are shared by multiple threads, the atomic variable ensures that threads don't crash into each other.
4. By using Final Keyword -
Final Variables are also thread-safe in java because once assigned some reference of an object It cannot point to reference of another object.

Q- 348) Suppose I have static method and static block which will be executed first.

The static block in a program always executed first before any static method

Q- 349) pipes in angular

Pipes are simple functions to use in template expressions to accept an input value and return a transformed value. Pipes are useful because you can use them throughout your application, while only declaring each pipe once.

Angular 7 provides some built-in pipes:

- Lowercasepipe
- Uppercasepipe
- Datepipe
- Currencypipe

- Jsonpipe
- Percentpipe
- Decimalpipe
- Slicepipe

Q- 350) interpolation in angular

Angular interpolation is used display a component property in the respective view template with double curly braces syntax. We can display all kind of properties data into view e.g. string number, date, arrays, list or map. Interpolation is used for one way data binding. Interpolation moves data in one direction from our components to HTML elements.

Usage of Interpolation in angular

- ☐ Display simple properties
- ☐ Evaluate arithmetic expressions
- ☐ Invoke methods and display return values
- ☐ Display array items

Q- 351) Spring Initializer

The Spring Initializer is ultimately a web application that can generate a Spring Boot project structure for you. It doesn't generate any application code, but it will give you a basic project structure and either a maven or a gradle build specification to build your code with.

It provides an extensible API to generate JVM based projects and to inspect the metadata used to generate projects, for instance to list the available dependencies and versions.

Q- 352) project architecture

Q- 353) annotation used

in project

Q- 354) spring scope

there are five types of bean scopes supported :

1. singleton(default*)

Scopes a single bean definition to a single object instance per Spring IoC container.

2. prototype

Scopes a single bean definition to any number of object instances.

3. request

Scopes a single bean definition to the lifecycle of a single HTTP request; that is each and every HTTP request will have its own instance of a bean created off the back of a single bean definition. Only valid in the context of a web-aware Spring ApplicationContext.

4. session

Scopes a single bean definition to the lifecycle of a HTTP Session. Only valid in the context of a web-aware Spring ApplicationContext.

5. global session

Scopes a single bean definition to the lifecycle of a global HTTP Session. Typically, only valid when used in a portlet context. Only valid in the context of a web-aware Spring ApplicationContext.

Q- 355) Component and controller annotation we can replace or not

If we use it on a controller class in Spring MVC and it will be more readable and appropriate. If we replace @Controller with @Component, Spring can automatically detect and register the controller class but it may not work with respect to request mapping.

Q- 356) Put and post annotation difference

Put Annotation (@put mapping) is generally used to update the resources

Post annotation (@post mapping) is generally used to create new resources

Q- 357) Key in mysql other than primary and foreign

There are mainly Eight different types of Keys in DBMS and each key has it's different functionality:

1. Super Key
2. Primary Key
3. Candidate Key
4. Alternate Key
5. Foreign Key
6. Compound Key
7. Composite Key
8. Surrogate Key

Let's look at each of the keys in DBMS with example:

- **Super Key** - A super key is a group of single or multiple keys which identifies rows in a table.
- **Primary Key** - is a column or group of columns in a table that uniquely identify every row in that table.
- **Candidate Key** - is a set of attributes that uniquely identify tuples in a table. Candidate Key is a super key with no repeated attributes.
- **Alternate Key** - is a column or group of columns in a table that uniquely identify every row in that table.
- **Foreign Key** - is a column that creates a relationship between two tables. The purpose of Foreign keys is to maintain data integrity and allow navigation between two different instances of an entity.
- **Compound Key** - has two or more attributes that allow you to uniquely recognize a specific record. It is possible that each column may not be unique by itself within the database.
- **Composite Key** - is a combination of two or more columns that uniquely identify rows in a table. The combination of columns guarantees uniqueness, though individual uniqueness is not guaranteed.

- **Surrogate Key** - An artificial key which aims to uniquely identify each record is called a surrogate key. These kind of key are unique because they are created when you don't have any natural primary key.

Q- 358) singleton class

It means define the class which has single instance that provide the global point of access to it called as singleton class. If you have business requirement in which only one object is created then you should go for singleton design pattern.

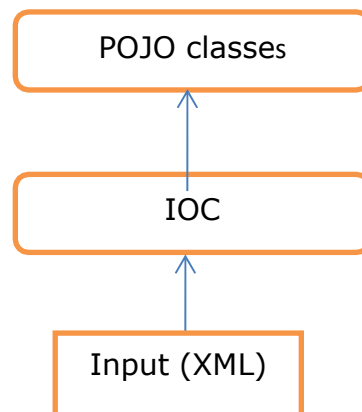
For example –If we created the three instance of same class but we want to create only one instance of class then we should go for singleton design pattern.

Q- 359) ioc container

IOC container is responsible to instantiate, configure and assemble this object.

Why container?

If you want to pass / apply any input to POJO classes. You must have container supports.



What does IOC container?

1. Read the XML file.
2. Create the instantiation of xml bean(java classes)
3. It will manage the life cycle of your bean classes.
4. Passing the dynamic parameter to bean (java classes).

There are two types of IOC container as

1. BeanFactory- It is called as core container.By using this, we can develop the standalone application.
2. ApplicationContext- It is called as J2EE container.

Q- 350) @controller , @service @repository, @component difference

Spring provides four different types of auto component scan annotations, they are @Component, @Service, @Repository and @Controller. Technically, there is no difference between them, but every auto component scan annotation should be used for a special purpose and within the defined layer.

@Component: It is a basic auto component scan annotation, it indicates annotated class is an auto scan component.

@Controller: Annotated class indicates that it is a controller component, and mainly used at the presentation layer.

@Service: It indicates annotated class is a Service component in the business layer.

@Repository: You need to use this annotation within the persistence layer, this acts like database repository.

Q- 351) lamda expression means and how it used

Lambda expression is a new and important feature of Java which was included in Java SE 8. 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.

The Lambda expression is used to provide the implementation of an interface which has functional interface. It saves a lot of code. In case of lambda expression, we don't need to define the method again for providing the implementation. Here, we just write the implementation code.

Java lambda expression is treated as a function, so compiler does not create .class file.

Q- 352) spring MVC flow

In MVC, the whole functionality is divided into three parts as-model, view and controller

Model- It can be an object or collection of objects which basically contains the data of the application which consist of POJO.

View- It is used for displaying the information to the user.

Controller- provides information between model and view. It contains the logical part of the application. @Controller annotation is used to mark that class as controller.

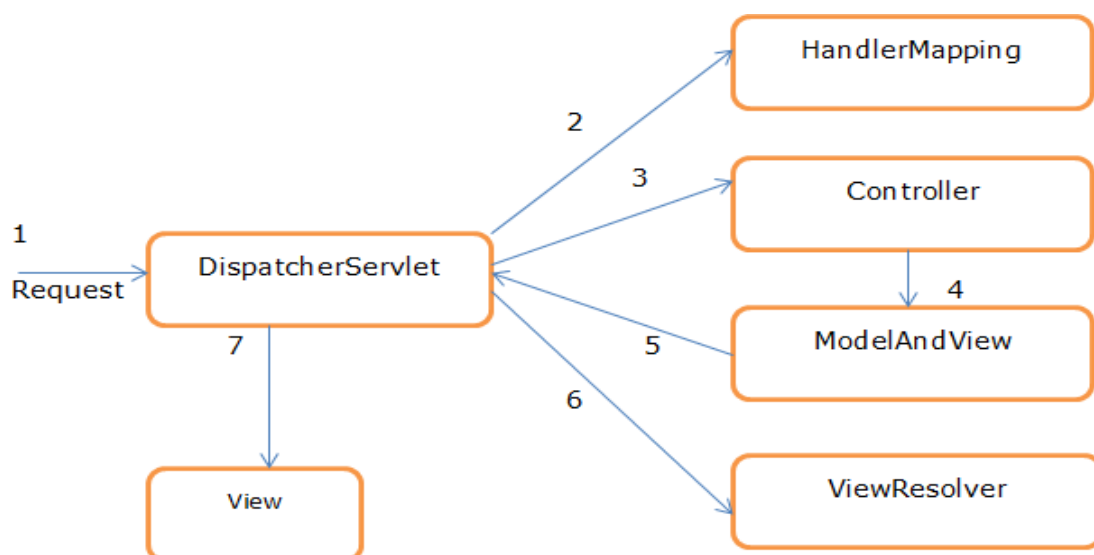


Fig- Spring MVC Flow

Description-

Step 1: When browser sent the request will be received by DispatcherServlet.

Step 2: DispatcherServlet will take the help of HandlerMapping and get to know the Controller class name associated with the given request.

Step 3: So request transfer to the Controller, and then controller will process the request by executing appropriate methods and returns ModelAndView object (contains *Model* data and *View* name) back to the DispatcherServlet.

Step 4: Now DispatcherServlet send the model object to the ViewResolver to get the actual view page.

Step 5: Finally DispatcherServlet will pass the *Model* object to the *View* page to display the result.

Q- 77 I have two jar 5 lit and 3 lit then I want 4 lit how you can give me

Q- 78 Car A 100km/hr and car B 200 km/hr distance between two car
100 km at what point they meet

Q- 353) fail-fast Iterator

Iterators in Java are part of the Java Collection framework. They are used to retrieve elements one by one. The Java Collection supports two types of iterators; Fail Fast and Fail Safe. These iterators are very useful in exception handling.

The Fail fast iterator aborts the operation as soon it exposes failures and stops the entire operation. Comparatively, Fail Safe iterator doesn't abort the operation in case of a failure. Instead, it tries to avoid failures as much as possible.

Q- 354) Can we access the non-final local variable, inside the local inner class?

Till JDK 7 Local Inner class can access only final local variable of the enclosing block however from JDK 8, it is possible to access the non final local variable of enclosing block in local inner class.

355) What is an applet, methods in applets?

Ans:- Applet is a special type of program that is embedded in the webpage to generate the dynamic content. It runs inside the browser and works at client side.

For creating any applet java.applet class must be inherited. It provides 4 life cycle methods of applet.

1. public void init(): is used to initialize the Applet. It is invoked only once.
2. public void start(): is invoked after the init() method or browser is maximized. It is used to start the Applet.

3. public void stop(): is used to stop the Applet. It is invoked when Applet is stop or browser is minimized.
4. public void destroy(): is used to destroy the Applet. It is invoked only once.

356) transient keyword?

Ans:- transient variable in Java is a variable whose value is not serialized during Serialization and which is initialized by its default value during de-serialization, for example for object transient variable it would be null.

357) javap command, strictfp keyword?

Ans:- The javap command disassembles a class file. The javap command displays information about the fields, constructors and methods present in a class file.

Syntax to use javap tool

Let's see how to use javap tool or command.

1. javap fully_class_name

Example to use javap tool

1. javap java.lang.Object

Java strictfp keyword ensures that you will get the same result on every platform if you perform operations in the floating-point variable. The precision may differ from platform to platform that is why java programming language have provided the strictfp keyword, so that you get same result on every platform. So, now you have better control over the floating-point arithmetic.

358) Can we declare an interface as final?

Ans:- If you declare a class final cannot extend it. ... If you make an interface final, you cannot implement its methods which defines the very purpose of the interfaces. Therefore, you cannot make an interface final in Java.

359) Can we override the static method?

Ans:- No, we can not override static method in java. Static methods are those which can be called without creating object of class, they are class level methods.

360) Can we override overloaded method?

Ans:- Yes, we can override a method which is overloaded in super class.

370) SQL query to display the current date?

Ans:- To get the current date and time in SQL Server, use the GETDATE() function. This function returns a datetime data type; in other words, it contains both the date and the time, e.g. 2019-08-20 10:22:34 .

SYSDATE returns the current date and time set for the operating system

```
SELECT TO_CHAR
      (SYSDATE, 'MM-DD-YYYY HH24:MI:SS') "NOW"
FROM DUAL;
```

371) -what is unique,composite key?

Ans:-

unique: A unique key is a set of one or more than one fields/columns of a table that uniquely identify a record in a database table. You can say that it is little like primary key but it can accept only one null value and it cannot have duplicate values.

Composite key:- A composite key is a candidate key that consists of two or more attributes (table columns) that together uniquely identify an entity occurrence (table row). A compound key is a composite key for which each attribute that makes up the key is a foreign key in its own right.

372) linux command mkdir & rmdir,ping,cat,ls,pwd?

Ans:-

mkdir:- **mkdir** command in Linux allows the user to create directories (also referred to as folders in some operating systems). This command can create multiple directories at once as well as set the permissions for the directories.

Rmdir:- The rmdir command removes the directory, specified by the Directory parameter, from the system. The directory must be empty before you can remove it, and you must have write permission in its parent directory. Use the ls -al command to check whether the directory is empty.

Ping:- PING (Packet Internet Groper) command is used to check the network connectivity between host and server/host. This command takes as input the IP address or the URL and sends a data packet to the specified address with the message “PING” and get a response from the server/host this time is recorded

which is called latency.

Cat:- Cat(concatenate) command is very frequently used in Linux. It reads data from the file and gives their content as output. It helps us to create, view, concatenate files.

Ls:- The **ls** is the list command in Linux. It will show the full list or content of your directory. Just type *ls* and press the enter key

Pwd:- pwd stands for Print Working Directory. It prints the path of the working directory, starting from the root. pwd is shell built-in command(pwd) or an actual binary(/bin/pwd). \$PWD is an environment variable which stores the path of the current directory.

373) Is Session a thread-safe object?

Ans:- No, Session is not a thread-safe object, many threads can access it simultaneously. In other words, you can share it between threads.

374) -lazy loading in hibernate?

Ans:- Its a design pattern which is used to delay initialization of an object as long as it's possible.

375) @PathVariable, @ResponseBody, @Valid, @RequestMapping, annotation?

Ans:- @pathvariable:-The @PathVariable annotation is used to extract the value from the URI. It is most suitable for the RESTful web service where the URL contains some value. Spring MVC allows us to use multiple @PathVariable annotations in the same method

@responsebody:-The @ResponseBody annotation tells a controller that the object returned is automatically serialized into JSON and passed back into the HttpResponse object.

@valid:-The @Valid annotation will tell spring to go and validate the data passed into the controller by checking to see that the integer numberBetweenOneAndTen is between 1 and 10 inclusive because of those min and max annotations.

@requestmapping:-@RequestMapping is one of the most common annotation used in Spring Web applications. This annotation maps HTTP requests to handler methods of MVC and REST controllers.

376) routingmodule, filter, factory method in Angular?

Ans:- routing module:- In Angular, the best practice is to load and configure the router in a separate, top-level module that is dedicated to routing and imported by the root AppModule . By convention, the

module class name is AppRoutingModuleModule and it belongs in the app-routing. ... --flat puts the file in src/app instead of its own folder.

Filter:- Filters allow us to format the data to display on UI without changing original format. Filters can be used with an expression or directives using pipe | sign. Angular includes various filters to format data of different data types.

Factory methods:- Factory Method makes the development process of AngularJS application more robust. A factory is a simple function which allows us to add some logic to a created object and return the created object.

377) Custom exception query?

Ans:-

378) Why we use runtime exception to extend?

Ans:- Having to add runtime exceptions in every method declaration would reduce a program's clarity. Thus, the compiler does not require that you catch or specify runtime exceptions (although you can). If you extend RuntimeException, you don't need to declare it in the throws clause (i.e. it's an unchecked exception).

379) Versions of java spring?

Ans:-

380) Which collection used mostly?

Ans:- The Java Collections Framework provides several general-purpose implementations of the core interfaces: For the Set interface, HashSet is the most commonly used implementation. For the List interface, ArrayList is the most commonly used implementation.

381) Hibernate cache types?

- **Ans:-** Session Cache. The session cache caches objects within the current session. It is enabled by default in Hibernate. ...
- Second Level Cache. The second level cache is responsible for caching objects across sessions. ...
- Query Cache. Query Cache is used to cache the results of a query

382) Hibernate ORM (native query) joining tables?

Ans:- You can assign aliases to associated entities or to elements of a collection of values using a join.

from Cat as cat

```
inner join cat.mate as mate
left outer join cat.kittens as kitten
```

383) Spring Boot - building tool?

Ans:- Spring STS - The Spring Tool Suite (STS) is an Eclipse-based development environment that is customized for developing Spring applications. Maven: A tool that you can use for building and managing any Java-based project.

384) Use of service in REST API - business logic?

Ans:- An API can be a front-end controller that receives the command from a user action or a REST Controller. ... The API is the facade. It should not contain the actual business logic of the task at hand. Instead, the real behavior should remain within the underlying Service class.

385) -Rest api crud operations (only Logic needed)?

Ans:- (Project related question)

386) -why java primitive?

Ans:- The main reason primitive data type are there because, creating object, allocating heap is too costly and there is a performance penalty for it. As you may know primitive data types like int, float etc are most used, so making them as Objects would have been huge performance hit.

387) features of java ??

Ans:-

1. Simple:-Java is easy to learn and its syntax is quite simple, clean and easy to understand.
2. Object oriented:-In java, everything is an object which has some data and behaviour. Java can be easily extended as it is based on Object Model. Following are some basic concept of OOP's.
3. Portable:-Java Byte code can be carried to any platform. No implementation dependent features. Everything related to storage is predefined, example: size of primitive data types
4. Platform independent:-On compilation Java program is compiled into bytecode. This bytecode is platform independent and can be run on any machine, plus this bytecode format also provide security. Any machine with Java Runtime Environment can run Java Programs.
5. Robust:-Java makes an effort to eliminate error prone codes by emphasizing mainly on compile time error checking and runtime checking. But the main areas which Java improved were Memory Management and mishandled Exceptions by introducing automatic garbage collector and Exception Handling.

388) what is multi threading and what is difference between sleep vs awake?

Ans: - Multithreading refers to a process of executing two or more threads simultaneously for maximum utilization of the CPU.

sleep() is used to introduce pause on execution

389) hibernate mapping in your project(project related question)

390) hibernate lazy loding ?

Ans: Lazy loading is a fetching technique used for all the entities in Hibernate. It decides whether to load a child class object while loading the parent class object. When we use association mapping in Hibernate, it is required to define the fetching technique. The main purpose of lazy loading is to fetch the needed objects from the database.

For example, we have a parent class, and that parent has a collection of child classes. Now, Hibernate can use lazy loading, which means it will load only the required classes, not all classes. It prevents a huge load since the entity is loaded only once when necessary. Lazy loading improves performance by avoiding unnecessary computation and reduce memory requirements.

391) spring MVC vs spring Boot difference

SPRING MVC

Spring MVC is a Model View, and Controller based web framework widely used to develop web applications.

1

If we are using Spring MVC, we need to build the configuration manually.

2.

In the Spring MVC, a deployment descriptor is required.

3.

Spring MVC specifies each dependency separately.

4.

Spring MVC framework consists of four components : Model, View, Controller, and Front Controller.

5.

SPRING BOOT

Spring Boot is built on top of the conventional spring framework, widely used to develop REST APIs.

If we are using Spring Boot, there is no need to build the configuration manually.

In the Spring Boot, there is no need for a deployment descriptor.

It wraps the dependencies together in a single unit.

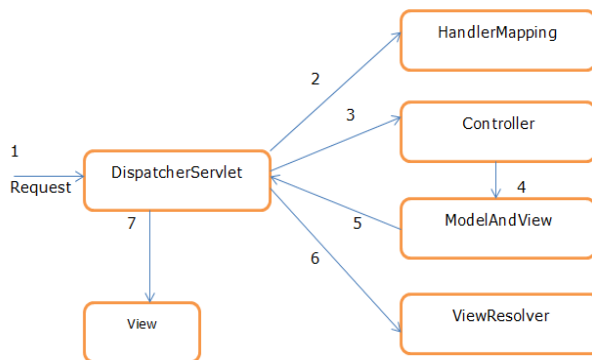
There are four main layers in Spring Boot: Presentation Layer, Data Access Layer, Service Layer, and Integration Layer.

6. It takes more time in development.

It reduces development time and increases productivity.

392) spring boot annotation use in your project(project related question)

393) how internally spring MVC work ?(spring MVC flow)



Step 1: When browser sent the request will be received by DispatcherServlet.

Step 2 : DispatcherServlet will take the help of HandlerMapping and get to know the Controller class name associated with the given request.

Step 3 : So request transfer to the Controller, and then controller will process the request by executing appropriate methods and returns ModelAndView object (contains *Model* data and *View* name) back to the DispatcherServlet.

Step 4: Now DispatcherServlet send the model object to the ViewResolver to get the actual view page.

Step 5: Finally DispatcherServlet will pass the *Model* object to the *View* page to display the result.

394) basics of git .how to push data and how to check status in git

Basics of git : Git is a **DevOps** tool used for source code management. It is a free and open-source version control system used to handle small to very large projects efficiently. Git is used to tracking changes in the source code, enabling multiple developers to work together on non-linear development

The git status command is used to display the state of the repository and staging area. It allows us to see the tracked, untracked files and changes. This command will not show any commit records or information.

395) what is agile difference between agile vs waterfall model ?

Waterfall	Agile
The waterfall methodology is sequential and linear.	Agile methodology is incremental and iterative.
Requirements have to be frozen at the beginning of SDLC.	Requirements are expected to change and changes are incorporated at any point.
The working model of software is delivered at the later phases of SDLC.	The working model is delivered during the initial phases and successive

	iteration of the model is delivered to the client for feedback.
It is difficult to scale-up projects based on waterfall methodology.	Scaling up of products is easy because of the iterative approach.
Customers or end-user doesn't have a say after the requirements are frozen during the initial phases. They only get to know the product once it is built completely.	Frequent customer interaction and feedbacks are involved in agile methodology.
Waterfall requires formalized documentations.	In agile documentation is often neglected and a working prototype serves as the basis for customer evaluation and feedback.
Testing is performed once the software is built.	Continuous testing is performed during each iteration.

396)

what you have used for connecting to DB in REST?

Ans: (Project related)

397). Methods of JPA

Ans: JPA stands for Java Persistence API. JPA allows you to avoid writing DML in the database specific dialect of SQL. JPA allows you to load and save Java objects and graphs without any DML language at all. When you do need to perform queries JPQL allows you to express the queries in terms of the Java entities rather than the (native)QL tables and columns

1.deleteAllByIdInBatch(Iterable<ID> ids)

Deletes the entities identified by the given ids using a single query.

2.deleteAllInBatch()

Deletes all entities in a batch call.

3.findAll()

4.flush()

Flushes all pending changes to the database.

5.getById(ID id)

Returns a reference to the entity with the given identifier.

6.getOne(ID id)

Deprecated.

use `JpaRepository#findById(ID)` instead.

398) where you have used multithreading in project?(project related question)

399). what is circle injection?

400) what is method reference ?

Ans: Method references are a special type of lambda expressions. They're often used to create simple lambda expressions by referencing existing methods.

There are four kinds of method references:

- **Reference to Static methods**

Syntax: ContainingClass::staticMethodName :

- **Reference to Instance methods**

Syntax: containingObject::instanceMethodName

401) Reference to Constructor

Syntax: ClassName::**new**

Example: reference of static method

```
interface Sayable{
    void say();
}

public class MethodReference {
    public static void saySomething(){
        System.out.println("Hello, this is static method.");
    }

    public static void main(String[] args) {
        // Referring static method
        Sayable sayable = MethodReference::saySomething;
        // Calling interface method
        sayable.say();
    } }
```

402). what is spring initializr ?

Ans: Spring Initializr is a **web-based tool** provided by the Pivotal Web Service. With the help of **Spring Initializr**, we can easily generate the structure of the **Spring Boot Project**. It offers extensible API for creating JVM-based projects.

It also provides various options for the project that are expressed in a metadata model. The metadata model allows us to configure the list of dependencies supported by JVM and platform versions, etc. It serves its metadata in a well-known that provides necessary assistance to third-party clients.

403). advantage of spring boot other than web application development ?

Ans: Spring Boot uses the Spring Framework as its foundation. It simplifies Spring dependencies and runs applications directly from the command line. It also doesn't require an external application container. Spring Boot helps control and customize application components externally.

- Fast and easy development of Spring-based applications;
- No need for the deployment of war files;
- The ability to create standalone applications;
- Helping to directly embed Tomcat, Jetty, or Undertow into an application;
- No need for XML configuration;
- Reduced amounts of source code;

404). types of collection ?

Ans: collection represent group of object as single entity.

Collection are divided into three interfaces:

1. Set: it is a child interface of collection. It represent group of object as a single entity . in set duplicates are not allowed and insertion order is not preserved.

2. List: : it is a child interface of collection. It represent group of individual object as a single entity .In list duplicates are allowed and insertion order is preserved.

3 Queue: it is a child interface of collection. If we want to represent a group of individual objects prior to processing then we should go for queue.

Eg: before sending a mail all mail id's we have to store somewhere and in which order we saved in the same order mail's should be delivered (First In First Out)for this requirement queue is best choice.

405).what is collection ?

Ans: If we want to represent group of object as a single entity then we should go for collection. Collection is a growable in nature. It holds homogenous and heterogeneous type of data .

406).What is REST?

The term REST stands for **REpresentational State Transfer**. It is an architectural style that defines a set of rules in order to create Web Services.

In a client-server communication, REST suggests to create an object of the data requested by the client and send the values of the object in response to the user.

407).Resquest used in REST?

Ans:The first REST API request in a session must be a **sign-in request**. This is a POST request that sends

the user credentials in the body of the request. Because this is a POST request, the request must include the Content-Type header. You can send your the body of the request block as XML or JSON.

408).Use of GET and POST?

- **Ans: Get-**It is used to read and retrieve resource
By using @GetMapping annotation
- **Post-**It is used to save and create new resource.

409).Can we send data using (HTTP)GET req.?

Ans: No, HTTP GET requests cannot have a message body. But you still can send data to the server using the URL parameters. In this case, you are limited to the maximum size of the URL, which is about 2000 characters (depends on the browser).

410).How you manage connection Front end with Back ?

Ans: Frontend and backend communicate with each other - via **Http requests**. The frontend will, for example, send entered data to the backend. The backend might then again validate that data (since frontend code can be tricked) and finally store it in some database.

411).How you do unit testing in Angular and REST?(project related)

412).How do you pass status code from service of REST?(project related)

413).What is angular?

Ans: Angular is a Typescript-based free and open-source web application framework led by the Angular Team at Google and by a community of individuals and corporations. Angular is a complete rewrite from the same team that built AngularJS.

414).What is Singleton design pattern?

Ans: It means define the class which has single instance that provide the global point of access to it called as singleton design pattern.

415).What is Factory Design pattern?

Ans: Design the factory method in which multiple objects are stored called as factory design pattern. Sometime our application or frameworks don't know what kind of object has to create at run time. It only knows when it has to create so that time we need to use factory design pattern.

416).Sql constrain?

Ans: SQL constraints are a set of rules implemented on tables in relational databases to dictate what data can be inserted, updated or deleted in its tables. This is done to ensure the accuracy and the reliability of information stored in the table.

Page 20 complete

1) Sql Join?

Ans- A SQL Join statement is used to combine data or rows from two or more tables based on a common field between them. Different types of Joins are:

1) **INNER JOIN** :- SELECT table1.column1,table1.column2,table2.column1,....

FROM table1
INNER JOIN table2
ON table1.matching_column = table2.matching_column;

2)LEFT JOIN :- SELECT table1.column1,table1.column2,table2.column1,....
FROM table1
LEFT JOIN table2
ON table1.matching_column = table2.matching_column;

3)RIGHT JOIN :- SELECT table1.column1,table1.column2,table2.column1,....
FROM table1
RIGHT JOIN table2
ON table1.matching_column = table2.matching_column;

4)FULL JOIN :- SELECT table1.column1,table1.column2,table2.column1,....
FROM table1
FULL JOIN table2
ON table1.matching_column = table2.matching_column;

417) How you manage Session within angular?

Ans:- With most AngularJS apps, there is no concept of a session. There might still be authentication and some kind of token, but storing session information, like form contents between pages, is stored in the browser in ram, cookies, session storage or local storage. This is how session is maintained from front end

418)How you authorized the user?

Ans: - In authentication, the user or computer has to prove its identity to the server or client. Usually, authentication by a server entails the use of a user name and password.

(Add project's details about authentication).

419What are Collection Type?

Ans:- There are three generic types of collection: ordered lists, dictionaries/maps, and sets.

419)Whats Servlet?

Ans:- Servlet is the technology that is used to create the dynamic web pages.

420) Custom exception query in notpad

Ans :- its wrong question

421) How hashCode & equals works in hashmap

Ans :- In HashMap, hashCode() is used to calculate the bucket and therefore calculate the index. equals method is used to check that 2 objects are equal or not. HashMap uses equals() to compare the key whether they are equal or not. If equals() method return true, they are equal otherwise not equal.

422) what is REST (EXACT MEANING)

Ans:- (a bodily state characterized by minimal functional and metabolic activities.)
REST stands for REpresentational State Transfer. It means when a RESTful API is called, the server will transfer to the client a representation of the state of the requested resource.

423) overload and method overriding

Ans:- Method Overloading :-

Method Overloading is a **Compile time polymorphism**. In method overloading, more than one method shares the same method name with different signature in the class. In method overloading, return type can or can not be same, but we must have to change the parameter because in java, we can not achieve the method overloading by changing only the return type of the method.

Method Overriding :-

Method Overriding is a **Run time polymorphism**. In method overriding, derived class provides the specific implementation of the method that is already provided by the base class or parent class. In method overriding, return type must be same or co-variant (return type may vary in same direction as the derived class).

424) Internal working of Map?

Ans :-

the process of converting an object into an integer value. The integer value helps in indexing and faster searches.

What is HashMap

HashMap is a part of the Java collection framework. It uses a technique called Hashing. It implements the map interface. It stores the data in the pair of Key and Value. HashMap contains an array of the nodes, and the node is represented as a class. It uses an array and LinkedList data structure internally for storing Key and Value. There are four fields in HashMap.

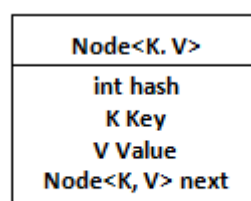


Figure: Representation of a Node

- equals(): It checks the equality of two objects. It compares the Key, whether they are equal or not. It is a method of the Object class. It can be overridden. If you override the equals() method, then it is mandatory to override the hashCode() method.
- hashCode(): This is the method of the object class. It returns the memory reference of the object in integer form. The value received from the method is used as the bucket number. The bucket number is the address of the element inside the map. Hash code of null Key is 0.
- Buckets: Array of the node is called buckets. Each node has a data structure like a LinkedList. More than one node can share the same bucket. It may be different in capacity.

Insert Key, Value pair in HashMap

We use put() method to insert the Key and Value pair in the HashMap. The default size of HashMap is 16 (0 to 15).

When we call the put() method, then it calculates the hash code of the Key "Aman." Suppose the hash code of "Aman" is 2657860. To store the Key in memory, we have to calculate the index.

Calculating Index

Index minimizes the size of the array. The Formula for calculating the index is:

$$1. \text{ Index} = \text{hashCode}(\text{Key}) \& (n-1)$$

Where n is the size of the array. Hence the index value for "Aman" is:

$$1. \text{ Index} = 2657860 \& (16-1) = 4$$

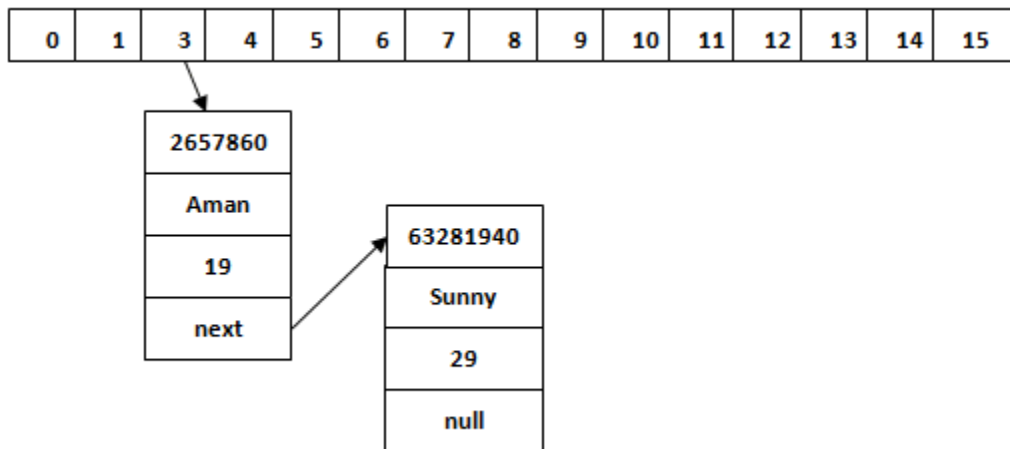
The value 4 is the computed index value where the Key and value will store in HashMap.

Hash Collision

This is the case when the calculated index value is the same for two or more Keys. Let's calculate the hash code for another Key "Sunny." Suppose the hash code for "Sunny" is 63281940. To store the Key in the memory, we have to calculate index by using the index formula.

$$1. \text{ Index} = 63281940 \& (16-1) = 4$$

The value 4 is the computed index value where the Key will be stored in HashMap. In this case, equals() method check that both Keys are equal or not. If Keys are same, replace the value with the current value. Otherwise, connect this node object to the existing node object through the LinkedList. Hence both Keys will be stored at index 4.



Similarly, we will store the Key "Ritesh." Suppose hash code for the Key is 2349873. The index value will be 1. Hence this Key will be stored at index 1.

get() method in HashMap

get() method is used to get the value by its Key. It will not fetch the value if you don't know the Key. When get(K Key) method is called, it calculates the hash code of the Key.

Suppose we have to fetch the Key "Aman." The following method will be called.

1. `map.get(new Key("Aman"));`

It generates the hash code as 2657860. Now calculate the index value of 2657860 by using index formula. The index value will be 4, as we have calculated above. get() method search for the index value 4. It compares the first element Key with the given Key. If both keys are equal, then it returns the value else check for the next element in the node if it exists. In our scenario, it is found as the first element of the node and return the value 19.

Let's fetch another Key "Sunny."

The hash code of the Key "Sunny" is 63281940. The calculated index value of 63281940 is 4, as we have calculated for put() method. Go to index 4 of the array and compare the first element's Key with the given Key. It also compares Keys. In our scenario, the given Key is the second element, and the next of the node is null. It compares the second element Key with the specified Key and returns the value 29. It returns null if the next of the node is null.

425)write code for remove duplicate from string=hello

Ans:-

1. `//Creating RemoveDuplicatesExample1 class`
2. `class RemoveDuplicatesExample1`
3. `{`
4. `//Creating removeDuplicates() method to remove duplicates from array`
5. `static void removeDuplicate(char str[], int length)`


```
6.  {
7.    //Creating index variable to use it as index in the modified string
8.    int index = 0;
9.
10.   // Traversing character array
11.   for (int i = 0; i < length; i++)
12.   {
13.
14.       // Check whether str[i] is present before or not
15.       int j;
16.       for (j = 0; j < i; j++)
17.       {
18.           if (str[i] == str[j])
19.           {
20.               break;
21.           }
22.       }
23.       // If the character is not present before, add it to resulting string
24.       if (j == i)
25.       {
26.           str[index++] = str[i];
27.       }
28.   }
29.   System.out.println(String.valueOf(Arrays.copyOf(str, index)));
30. }
31.
32. // main() method starts
33. public static void main(String[] args)
34. {
35.     String info = "javaTpoint is the best learning website";
36.     //Converting string to character array
37.     char str[] = info.toCharArray();
38.     //Calculating length of the character array
39.     int len = str.length;
40.     //Calling removeDuplicates() method to remove duplicate characters
41.     removeDuplicate(str, len);
42. }
43. }
```

426)can we replace @RestController annotation with @controller

Ans :-

@Controller is used in legacy systems which use JSPs. it can return views. @RestController is to mark the controller is providing REST services with JSON response type. so it wraps @Controller and @ResponseBody annotations together.

427)can we replace @repository to @service

Ans:-

According to [documentaion](#) @Repository,@Service,@Controller are all synonyms. They all are just specializations of @Component annotation. So, generally, they can be used one instead of other. But ... you should not do this.

428)difference between comparable and comparator

Ans:-

Comparable	Comparator
1) Comparable provides a single sorting sequence . In other words, we can sort the collection on the basis of a single element such as id, name, and price.	The Comparator provides multiple sorting sequences . In other words, we can sort the collection on the basis of multiple elements such as id, name, and price etc.
2) Comparable affects the original class , i.e., the actual class is modified.	Comparator doesn't affect the original class , i.e., the actual class is not modified.
3) Comparable provides compareTo() method to sort elements.	Comparator provides compare() method to sort elements.
4) Comparable is present in java.lang package.	A Comparator is present in the java.util package.
5) We can sort the list elements of Comparable type by Collections.sort(List) method.	We can sort the list elements of Comparator type by Collections.sort(List, Comparator) method.

429)difference between @controller and @restcontroller

Ans:-

1. The **@Controller** is a annotation to mark class as Controller Class in Spring
While **@RestController** is used in REST Web services and similar
to **@Controller** and **@ResponseBody**.
2. The **@Controller** annotation indicates that the class is controller like web Controller
while **@RestController** annotation indicates that the class is controller
where **@RequestMapping** Method assume **@ResponseBody** by Default(i.e REST APIs).
3. The key difference is that you do not need to use **@ResponseBody** on each and every handler method
once you annotate the class with **@RestController**.
4. **@Controller** create a Map of Model Object and find a view while **@RestController** simply return
object and object data directly written into http response as JSON orXML.

430)Stack Data-Structure push () & pop () methods implementation

Ans:-

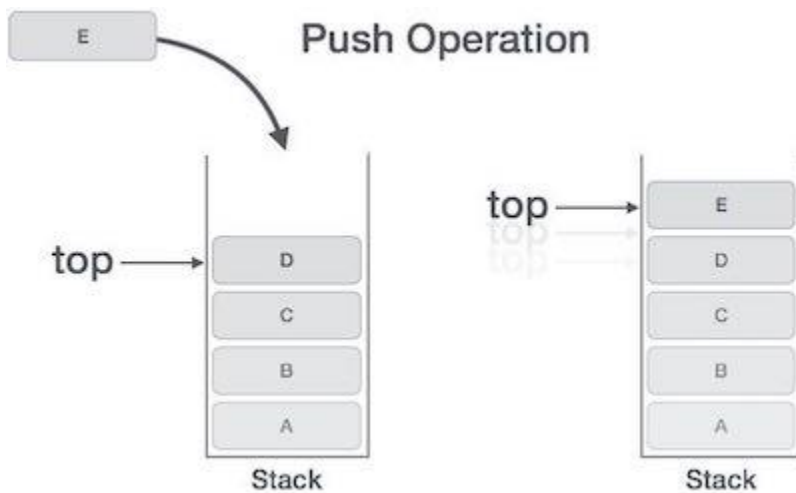
- **push()** – Pushing (storing) an element on the stack.
- **pop()** – Removing (accessing) an element from the stack.

When data is PUSHed onto stack.

Push Operation

The process of putting a new data element onto stack is known as a Push Operation. Push operation involves a series of steps –

- **Step 1** – Checks if the stack is full.
- **Step 2** – If the stack is full, produces an error and exit.
- **Step 3** – If the stack is not full, increments **top** to point next empty space.
- **Step 4** – Adds data element to the stack location, where top is pointing.
- **Step 5** – Returns success.



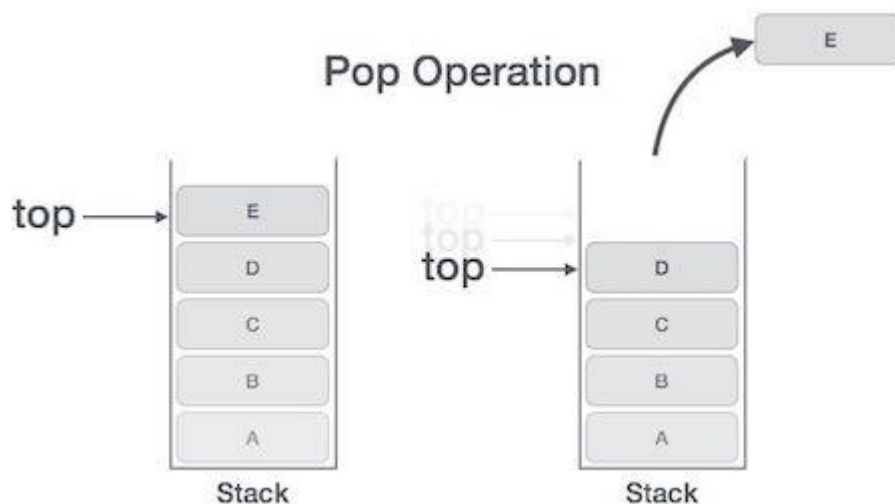
If the linked list is used to implement the stack, then in step 3, we need to allocate space dynamically.

Pop Operation

Accessing the content while removing it from the stack, is known as a Pop Operation. In an array implementation of pop() operation, the data element is not actually removed, instead **top** is decremented to a lower position in the stack to point to the next value. But in linked-list implementation, pop() actually removes data element and deallocates memory space.

A Pop operation may involve the following steps –

- **Step 1** – Checks if the stack is empty.
- **Step 2** – If the stack is empty, produces an error and exit.
- **Step 3** – If the stack is not empty, accesses the data element at which **top** is pointing.
- **Step 4** – Decreases the value of top by 1.
- **Step 5** – Returns success.



431) Different steps of Tomcat server

Ans:-

432) How to use JUnit

Ans:- To use JUnit you must create a separate **. java file** in your project that will test one of your existing classes. In the Package Explorer area on the left side of the Eclipse window, right-click the class you want to test and click New → JUnit Test Case. A dialog box will pop up to help you create your test case.

433) what is index

Ans:- An index is an on-disk structure associated with a table or view that speeds retrieval of rows from the table or view.

434) What is stored procedure

Ans:- A procedure is a combination of SQL statements written to perform a specified tasks. It helps in code re-usability and saves time and lines of code.

435) what is trigger

Ans:- A trigger is a special kind of procedure which executes only when some triggering event such as INSERT, UPDATE, DELETE operations occurs in a table.

436) How to control amount of thread execution at one time

Ans:- By using Synchronization.

437). what is servlet filter

Ans:- A Servlet filter is **an object that can intercept HTTP requests targeted at your web application**. A servlet filter can intercept requests both for servlets, JSP's, HTML files or other static content, as illustrated in the diagram below: A Servlet Filter in a Java Web Application

438) 10. difference between union and union all.

Ans:- The only difference between Union and Union All is that **Union extracts the rows that are being specified in the query** while Union All extracts all the rows including the duplicates (repeated values) from both the queries.

361) diff betn controller and restcontroller ?

@Controller	@RestController
1) The @Controller is a common annotation which is used to mark a class as Spring MVC Controller	1) The @RestController is a special controller used in RESTful web services and the equivalent of @Controller + @ResponseBody.

2) @Controller is an old annotation, exists since Spring started supporting annotation, and officially it was added on Spring 2.5 version.	2) The @RestController is relatively new, added only on Spring 4.0
3) The @Controller annotation indicates that the class is a "Controller" e.g. a web controller	3) @RestController annotation indicates that the class is a controller where @RequestMapping methods assume @ResponseBody semantics by default i.e. servicing REST API.

439) What is spring bean? How it is defined?

Ans: In Spring, the objects that form the backbone of your application and that are managed by the Spring IoC container are called beans. A bean is an object that is instantiated, assembled, and otherwise managed by a Spring IoC container. Otherwise, a bean is simply one of many objects in your application.

A bean is an object that is instantiated, assembled, and otherwise managed by a Spring IoC container. These beans are created with the configuration metadata that you supply to the container.

440) status code?

Ans: Status code is a 3-digit integer. The first digit of status code is used to specify one of five standard classes of responses. The last two digits of status code do not have any categorization role.

441) diff between native query and named query ?

Ans: 1) Native query refers to actual SQL queries (referring to actual database objects). These queries are the SQL statements which can be directly executed in database using a database client.

2) Named query is the way you define your query by using by giving it a name. You could define this in mapping file in hibernate or also using annotations at entity level.

442) how do you integrate spring with hibernate ?

Ans: In hibernate framework, we provide all the database information in hibernate.cfg.xml file.

But if we are going to integrate the hibernate application with spring, we don't need to create the hibernate.cfg.xml file. We can provide all the information in the applicationContext.xml file.

We can simply integrate hibernate application with spring application.

...

Steps

1. create table in the database It is optional.
2. create applicationContext.xml file It contains information of DataSource,SessionFactory etc.
3. create Employee.java file It is the persistent class.
4. create employee.hbm.xml file It is the mapping file.

443) why jps come into picture ?

Ans: JPS command is used to check if a specific daemon is up or not. The command of JPS displays all the processes that are based on Java for a particular user. The command of JPS should run from the root to check all the operating nodes in the host

444) diff betn properties file and map ?

Ans: 1. HashMap :- If you are searching for a particular key in the HashMap and you have implemented correct hashCode() and equals() contract then by the hashing the getting and setting key, value will be fast as it will use hashing and red black tree indexing for searching.

cons :- you have to initialize the HashMap i.e. reading every value from File and put it to the HashMap.

2. PropertyFile :- If you look at the internal implementation of PropertyFile.java it internally uses the HashTable. public class Properties extends Hashtable<Object, Object> So once you loaded the property file into the object. The performance comparison between them is same as HashTable vs. HashMap performance.

445) real time example of dependency injection ?

Ans: Suppose you give 1ltr water to me frequently, You use 10 cups of 100ml for that. So every time you come with 10 cups !....

Now, suppose you got a jug of 1ltr What would you do.?

I'll use that every time, because it has functionality of doing your work easily...simple...

In technical way, The 1ltr jug is your Dependency Injection, it will make your work much easier...

446)how can you manage users in application ?

Ans: Add new user profiles to the app.

1. Search for a user's profile.
2. Assign user tags to user's profiles.
3. Ban and unban users.
4. Export the data of your registered users.

447)why spring bean is immutable ?

Ans: In fact, spring beans are immutable by idea, even though you are not enforcing this. You can provide only a getter to a final field that is initialized through constructor injection.

448) strong oops concept required.

Ans: OOps, concepts in java is to improve code readability and reusability by defining a Java program efficiently. The main principles of object-oriented programming are abstraction, encapsulation, inheritance, and polymorphism. These concepts aim to implement real-world entities in programs.

OOP concepts let us create working methods and variables, then re-use all or part of them without compromising security.

449) Strong exception handling and string is required.Ans:

Java exception handling is important because it helps maintain the normal, desired flow of the program even when unexpected events occur. If Java exceptions are not handled, programs may crash or requests may fail. This can be very frustrating for customers and if it happens repeatedly, you could lose those customers.

450) java 8 features must 1 question definitely sometimes program

also.Ans: java 8 provides following features for Java Programming:

Lambda expressions,

Method references,

Functional interfaces,
Stream API,
Default methods, Base64
Encode Decode,
Static methods in interface,
Optional class,
Collectors class,
ForEach() method,
Parallel array sorting,
Nashorn JavaScript Engine,
Parallel Array Sorting,
Type and Repating Annotations,IO
Enhancements,
Concurrency Enhancements,JDBC
Enhancements etc.

Lambda Expressions

Lambda expression helps us to write our code in functional style. It provides a clear and concise way to implement SAM interface(Single Abstract Method) by using an expression. It is very useful in collection library in which it helps to iterate, filter and extract data.

Method References

Java 8 Method reference is used to refer method of functional interface . It is compact and easy form of lambda expression. Each time when you are using lambda expression to just referring a method, you can replace your lambda expression with method reference.

Functional Interface

An Interface that contains only one abstract method is known as functional interface. It can have any number of default and static methods. It can also declare methods of object class.

Functional interfaces are also known as Single Abstract Method Interfaces (SAM Interfaces).

Optional

Java introduced a new class `Optional` in Java 8. It is a public final class which is used to deal with `NullPointerException` in Java application. We must import `java.util` package to use this class. It provides methods to check the presence of value for particular variable.

forEach

Java provides a new method `forEach()` to iterate the elements. It is defined in `Iterable` and `Stream` interfaces.

It is a default method defined in the `Iterable` interface. Collection classes which extend `Iterable` interface can use `forEach()` method to iterate elements.

This method takes a single parameter which is a functional interface. So, you can pass lambda expression as an argument.

Date/Time API

Java has introduced a new Date and Time API since Java 8. The `java.time` package contains Java 8 Date and Time classes.

Default Methods

Java provides a facility to create default methods inside the interface. Methods which are defined inside the interface and tagged with `default` keyword are known as default methods. These methods are non-abstract methods and can have method body.

Nashorn JavaScript Engine

Nashorn is a JavaScript engine. It is used to execute JavaScript code dynamically at JVM (Java Virtual Machine). Java provides a command-line tool `jjc` which is used to execute JavaScript code.

You can execute JavaScript code by two ways:

1. Using `jjc` command-line tool, and
2. By embedding into Java source code.

StringJoiner

Java added a new final class StringJoiner in java.util package. It is used to construct a sequence of characters separated by a delimiter. Now, you can create string by passing delimiters like comma(,), hyphen(-) etc.

Collectors

Collectors is a final class that extends Object class. It provides reduction operations, such as accumulating elements into collections, summarizing elements according to various criteria etc.

Stream API

Java 8 java.util.stream package consists of classes, interfaces and an enum to allow functional-style operations on the elements. It performs lazy computation. So, it executes only when it requires.

Stream Filter

Java stream provides a method filter() to filter stream elements on the basis of given predicate. Suppose, you want to get only even elements of your list, you can do this easily with the help of filter() method.

This method takes predicate as an argument and returns a stream of resulted elements.

Java Base64 Encoding and Decoding

Java provides a class Base64 to deal with encryption and decryption. You need to import java.util.Base64 class in your source file to use its methods.

This class provides three different encoders and decoders to encrypt information.

Java Parallel Array Sorting

Java provides a new additional feature in Arrays class which is used to sort array elements parallelly. The parallelSort() method has been added to java.util.Arrays class that uses the JSR 166 Fork/Join parallelism common pool to provide sorting of arrays. It is an overloaded method.

451) .Junit basics is imp and also diff tools used for that .

Ans: JUnit is a unit testing framework for the Java programming language. JUnit has been important in the development of test-driven development and is one of a family of unit testing frameworks which is collectively known as xUnit that originated with SUnit.

452) How to handle exception in java project?

- Use try/catch/finally blocks to recover from errors or release resources. ...
- Handle common conditions without throwing exceptions. ...
- Design classes so that exceptions can be avoided. ...
- Throw exceptions instead of returning an error code.

OR

- If a method is considered risky (it might not execute properly due to exceptional circumstances) it must declare that it throws an exception.
- The risky method that is called will throw the exception.
- The method that calls the risky method will catch the exception.
- The files are in the computer...

453) Define default exception provided by spring boot?

Handling exceptions and errors in APIs and sending the proper response to the client is good for enterprise applications.

Controller Advice :

The @ControllerAdvice is an annotation, to handle the exceptions globally.

Exception Handler :

The @ExceptionHandler is an annotation used to handle the specific exceptions and sending the custom responses to the client.