Top Spring Framework Question

Q:⇒ What is Spring Framework?

Ans:- ✱ It is an open source java Application development framework.

✱ It supports building all types of java Application like Web Application, DB Driven Application, Enterprise Application.

Q:- What are the features of Spring framework?

Ans:-
1) Leight Weight
2) Non invasive
3) Loosely Coupled
4) IOC
5) Spring Container
6) AOP

1.1 :- Spring jars are smaller in size <1mb

1.2 :- No need to implement any predefined code.

1.3 :- AOP helps in creating loosely Coupled Code base.

1.4 :- It Allows Creation of object that will be manged by Spring Container and its depency too will be resolved automatically.

1.5 :- It takes care of object creation, initialization and managing object dependencies.

1.6 :- promotes seperation of concern such as logging transaction, security from core business logic.

Q:→ What do you mean by IOC Container?

Ans:→ ★ Represents the inversion of responsibility of object creation, Initialization and destruction (life cycle Management) from application to spring Container.

★ In normal programming, We create object with new keyword.

★ In spring we depend on Container to provide us with the required dependency object.

@Component

★ spring going Container to come ok This is @Component that means it is a bean. I should be creating a bean and handling it, and I should be creating the singleton object of this Engine(class).

Qy:→ Dependency Injection, && IOC?

Qy:→ Implicit object in JSP?

Qy:→ Singleton class / Design Pattern?

Qy:→ MVC layers structure.

Java Interview Question

Qus:→ Can we Override static Methods

Ans:- ✱ No we can Override static Method in java because overriding is based upon dynamic binding run time and static Methods are bonded using static binding at Compile time.

Qus:→ Can we access private Method in java?

Ans:- ✱ yes we can access but in same class but not outside the class.

Qus:→ Final keyword use?

Ans:- 1) Final variable (we can not change value)
2) Final Method (we can't inherit Method)
3) Final class (we can't extend any subclass but it can extend other class.

Qus:- checked and unchecked exception?

Ans:- checked → 1) SQL Exception
2) IO Exception
3) class not found Exception.

unchecked:→ 1) Null pointer Exception
2) Array Index out of Bound Exception
3) Illegal Argument Exception
4) Illegal state Exception

Qus!- Difference Abstract class and Interface in java?

Ans!- (When to use abstract class and Interface)?

Ans:-

Qus!- Method Overloading and Overriding

Qus!- Encapsulation vs Abstraction.

Qus!- Have you Applied object cloning in the project?

Ans!- object cloning in Java is a process of creating and exact copy of an object. It basically means ability to create an object with a similar state as a original object to achieve this java provides a Method of clone.

Qus!- Hashset and HashMap?

Ans!-

Qus!- When to use Transient variable

Ans!- When to want make variable not serializable class

Other

* You can use it for a variable whose value you don't want to save. so by that time will use Transient value.

Qus:- When To use ArrayList and LinkedList?

Ans:- ArrayList

* ArrayList is the best choice if our frequent operation is retrival that means Getting some data or when search operation need to be Performe.

: LinkedList

* Linked List is best choice Frequent operation is Insert and Delete operation in the Middle. since doesn't need several shift operation Internally.

Qus: String Buffer Vs String Builder?

Ans:-

Qus:- How to Create User defined Exceptions?

Ans:-

Qus: Java 8 Features?

Ans:-

Qus:- Loose Coupling & Tight Coupling?

Qus:→ Buffered Reader & Scanner?

Qus:→ Singleton class in Java?

Qus:→ Class Not found Exception Vs No class defound Error.

Qus → Constructor and Method?

Top Interview Spring Boot Question

Q:- Is it possible to change the port of the embedded Tomcat server in Spring Boot?

Ans:- Bydefault → 8080

* yes we can change port. By using the server.port = 9090

Q:- Can we Override or replace the Embedded tomcat server in Spring Boot?

Ans:- * yes we can replace the Embedded Tomcat server with any server by using the starter dependency in the pom.xml File.

Qus:- What is auto-configuration in Spring Boot? How does it help? Why Spring Boot is called opinionated?

Ans:-

* Spring Boot works automatically.
* We will tell only application properties in configuration and it will work automatic configuration.

@SpringBootApplication

* It enable feature (auto configuration)

Qus:- What is the difference b/w @SpringBootApplication and @EnableAutoConfiguration?

Ans:- @EnableAutoConfiguration is used to enable auto-configuration but @SpringBootApplication does a lot more than that.

It also combines @Configuration and @ComponentScan annotations to enable Java-based configuration and Component Scanning in your project.

@SpringBootApplication = @Configuration + @EnableAutoConfiguration + @ComponentScan

Qus:- How to disable a specific auto-configuration class?

Ans:- We can use exclude attribute of @EnableAutoConfiguration annotation to exclude specific class.

Ex:- // use of exclude

@EnableAutoConfiguration(exclude = {ClassName})

Qus:- What is the difference between @RestController and @Controller in Spring Boot?

Ans:-

@Controller:- Map of the model object to view to template and make it human readable.

@RestController:- Simply Return the object and object data is directly

in HTTP response as JSON or XML.

@Controller :- We can create a Controller using @Controller and Mapping the Method of the Url and It will execute the Method but Method returns View Name / Template and we can send Dynamic data.

@RestController :- It is same as Controller but return value directly HTTP Response in JSON automatic Write.

* It create to more API.

@RestController = @Controller + @ResponseBody

Qus:- Describe the flow of HTTP Requests through the Spring Boot Application?

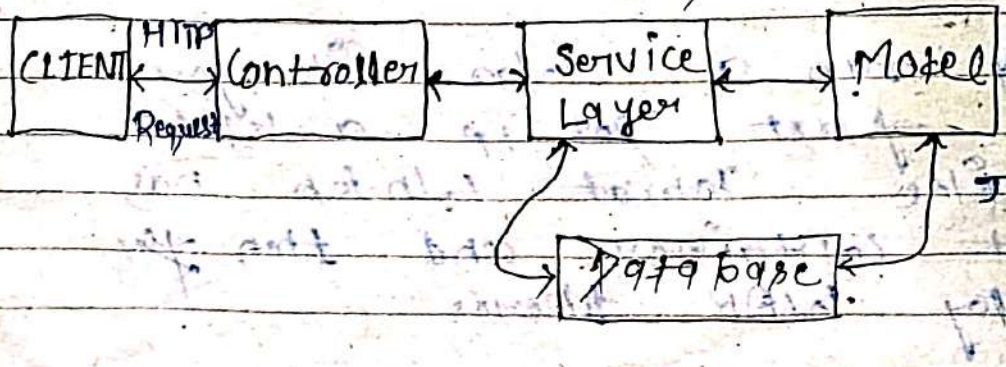
Ans:-

Controller

ServiceLayer

Model

Dependency Injection



Q4:- What is the difference b/w @RequestMapping and @GetMapping?

Ans:-

A) @RequestMapping can be used with GET, POST, PUT, and many other request methods using the method attribute on the annotation.

Whereas -
@GetMapping is only an extension of @RequestMapping with Get Method which helps you to improve on clarity on Request.

```
@RequestMapping(value="/user/{userId}", method = RequestMethod.GET)
```

```
@GetMapping(value="/user/{userId}")
```

Q5:- What is the difference b/w Embedded container and a WAR?

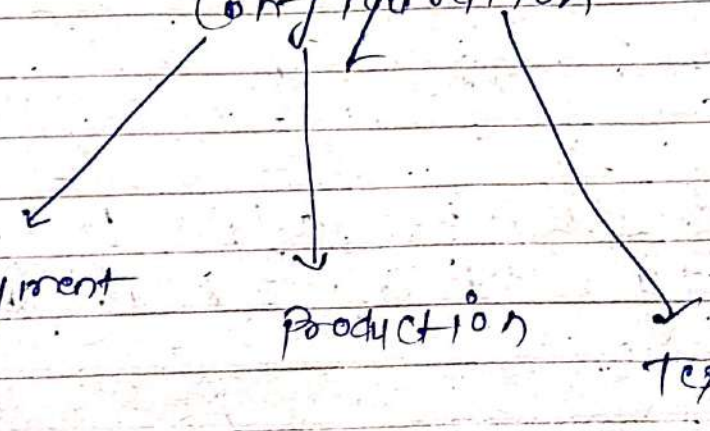
Ans:-

A) The main difference on embedded container and a WAR file is that you can spring boot application as a JAR file from the command prompt without getting up a live server.

But to run a war file, you need to first set up a Web servers like Tomcat which has a servlet container and then you need to deploy WAR there.

Qus:- What is the use of profiles in spring boot?

Ans:- * spring has the provision of profiles to keep the seperate configration of environment



Qus:- What is spring Actuator? What are its advantages

Ans:- * provides special feature to monitor and manage your application when you push it to production.

* Monitor and Manage all everything

<spring-boot-starter-actuator>

Qus:- How to get the list of all beans of your spring boot Application?

Ans:- A) Spring boot actuator "/Beans" is used to get the list of all spring Beans

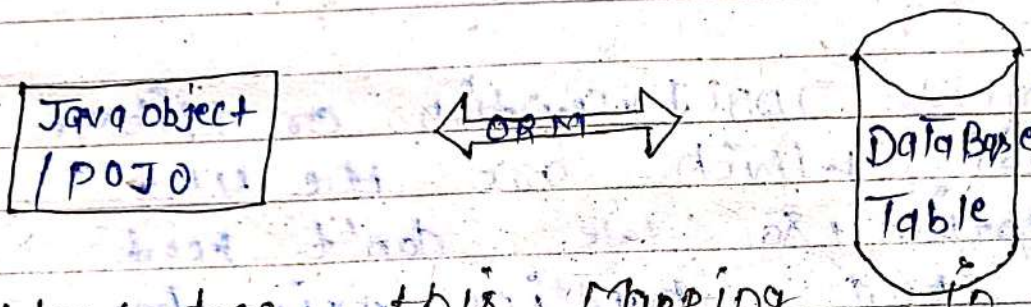
B) "/env" it return the list of all environment properties of running the spring Boot Application.

Hibernate

Qus:- ~~What is JPA~~ How is Hibernate related to JPA?

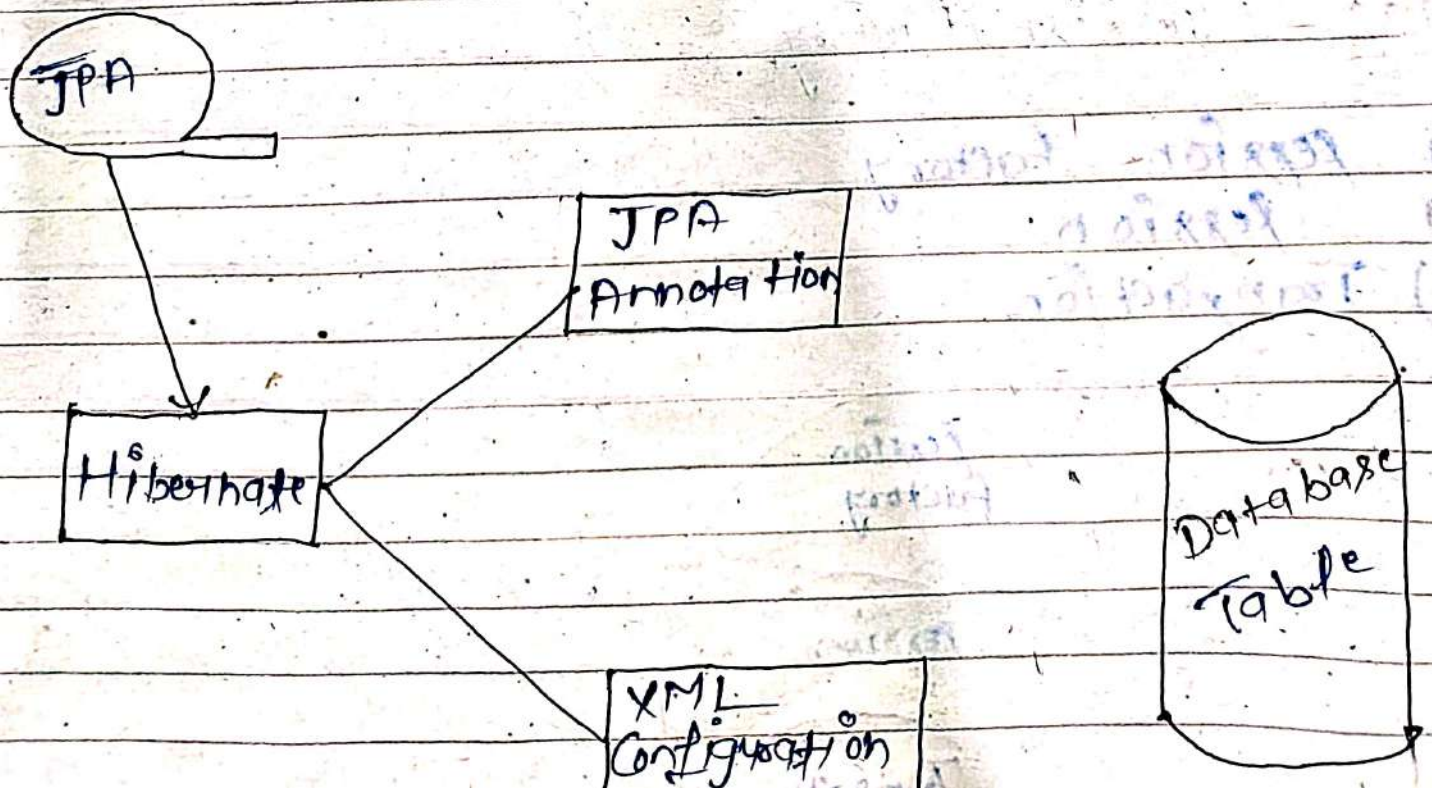
Qus:- What is Hibernate and why to use it?

Ans:- Hibernate is an object-relational mapping (ORM) tool used to map Java objects and database table.



Q How does this Mapping done Table?

* It provides JPA implementation hence we can use JPA annotation as well as xml configuration to achieve this mapping.



Q48:⇒ Why Hibernate?

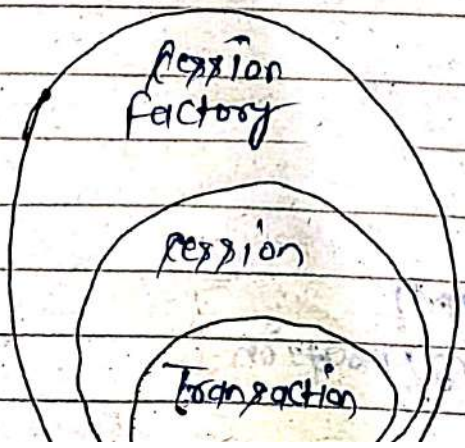
Ans:=

* Hibernate eliminates all the boiler-plate code that comes with JDBC.
* It supports HQL with is more object Oriented.
* It provides transaction management Implicitly.
* Hibernate throws JDBC Exception or Hibernate Exception which are the un-checked exception. So we don't need to worry about handling fixing try and catch.
* It supports caching for better performance.

Qus:- Important Interfaces used in Hibernate

Ans:- Session Factory :-

1) session Factory
2) session
3) Tranxaction



Q:- What are the Annotations used in Hibernate?

Ans:-

* Javax.persistence.Entity :- Used with Model class to specify that they are entity beans.

* Javax.persistence.Table :- Used with entity beans to define the Corresponding table name in database.

* Access :- Used to define the access Type either field or property. and Default value is field and if you want Hibernate to use getter and setter Methods that you need to set it to property.

E.g.- @Access(value= AccessType.PROPERTY)

* Id :- Used to define the primary key in the entity bean.

* Embeded Id :- Used to define Composite primary key in the entity bean.

* Column :- used to define the Column name in database Table.

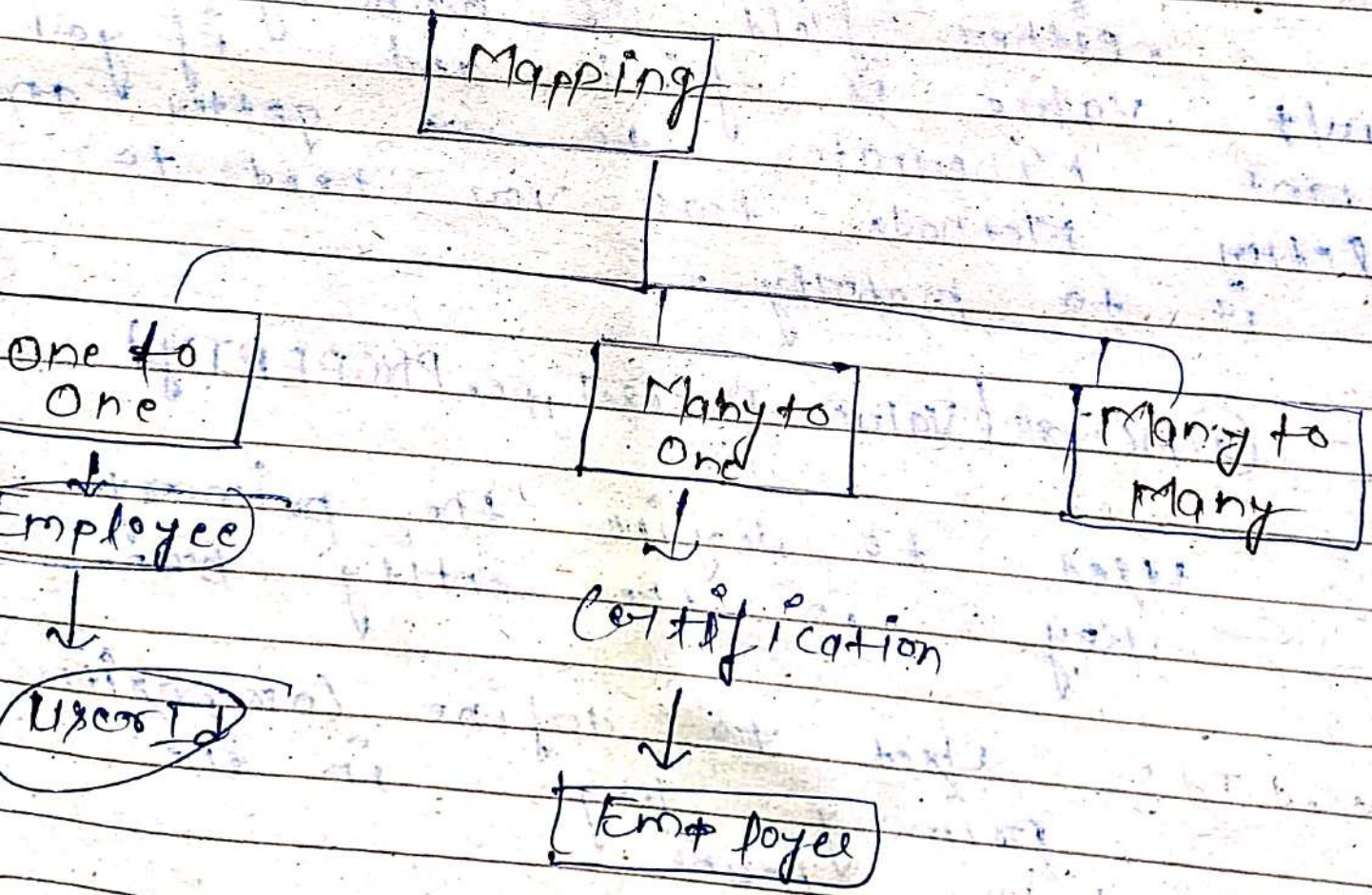
* GeneratedValue :- Used for generation of the primary key.

* Cascade :- Used to define the Cascading between Between two entity beans. Used with Mappings. It works in conjunction with org.hibernate.annotations.CascadeType.

* Primary key JoinColumn :- Using.

Join column Used to define the property for foreign key.

Mappings in Hibernate:



Q ⇒ Difference openSession & getCurrentSession?

Ans:-

★ getCurrentSession() Method returns the session bound to the context.
Since this session object belongs to the ~~object~~ context of Hibernate, it is okay if you don't close it, once the sessionFactory is closed, this session object gets closed.

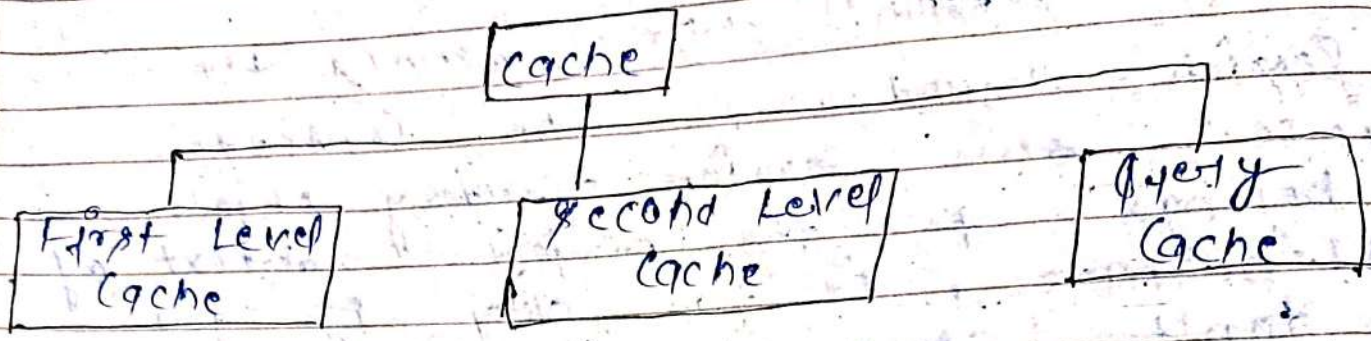
While,

openSession() Method helps in opening a new session.
You should close this session object once you are done with all the database operations. And also, you should open a new session, for each request in a Multi-threaded environment.

Que:- Difference session get() and load() Methods

Ans:- ★ get() loads the data as soon as it is called whereas load() returns a proxy object and loads data only when it's actually required, so load() is better because it support lazy loading.
★ Since load() throws exceptions when the data is not found, we should use it only when we know data exist.
★ We should use get() when we want to make sure data exist in Database.

Q=> Hibernate caching - Types



FIRST LEVEL CACHING

* Hibernate caches query data to make our application faster and improve performance.
* The idea behind cache is to reduce the number of database queries.
* Hibernate first level cache is associated with the session object.
* Hibernate first level cache is enabled by default and there is no way to disable it.
* Still hibernate provides through which we can delete selected objects from the cache or clear the cache completely.
* Any object cached in a session will not be visible to other session and when the session is closed all the cached object will also be lost.

SECOND LEVEL CACHE

* Hibernate second Level cache is disabled by default but we can enable it through configuration.

* Currently EHCache and Infinispan provides implementation for Hibernate second level cache and we can use them.

Q4:- How to Integrate Hibernate and spring?

Ans:- * spring is JAVA EE Framework. extensively used in enterprise application.

* Hibernate is the most popular ORM framework.

* This is why spring Hibernate Combination is used in a lot of enterprise Application.

* Steps to Integrate

* Add required Dependencies.

* Create Entity, Controller, Service and DAO layers required.

* And that it, its as simple as that.

Java Interview Question

Ques:- What is Java?
Ans:- Java is the High level programming Language that was developed by James Gosling in the year 1982. It is based on the principels of Object Oriented programing.

Ques:- Why is Java plateform independent Language.
Ans→ Java language has developed in such a way that it does not depend on any hardware or software due to the fact that the compiler compiles the code and then convert it to plateform-Independent byte Code which can be run on Multiple system.

Q→ Diff. : Heap and stack Memory in Java and how Java utilize this)
Ans:- stack Memory is the portion of memory that was assigned to every Individual program. And it was Fixed. On the other Hand.

Heap Memory is the portion that was not allocated to the Java program but it will be available For use by the Java progra. Mostly runtime required the program.

Java utilize the memory as

* When We write a Java pgm then all the variables, methods, etc are

stored in the stack.
* When we create any object in java program then that object was created in the heap Memory. And it was referenced from the stack Memory.

Qus :- Local Variable & Instance Variable?
Ans :- Instance Variable :- Are those Variable that are accessible by all the Methods in the class. They are declared outside the Methods and inside the class.
Local Variable :- Are those Variable present within a block, function or constructor and can be accessed only inside them.

Qus :- JIT Compiler?
Ans → Jit stands for Just In Time Compiler. and it is used for improving the performance during run time.

Qus ⇒ Difference b/w equals() Method and equality (==) operator in Java?
Ans :- We already know (==) equal operator. That we have used this to compare the equality of the values.
But when we talk about the terms of OOP. We deal with the values in the form of object. And this may contains multiple types of data. So using the (==) operator don't work in this case.

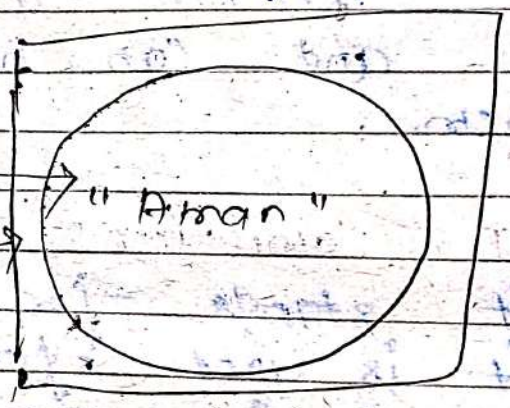
Ex:-

```
string Name1 = "Aman";
string Name2 = "Aman";

o/p (Name1 == Name2)
```

Ans- True

here (==) operator don't compare each characters in this case. It compares the memory location. And the string uses the constant pool for storing the values in the memory both Name1 and Name2 are stored at the same memory location.

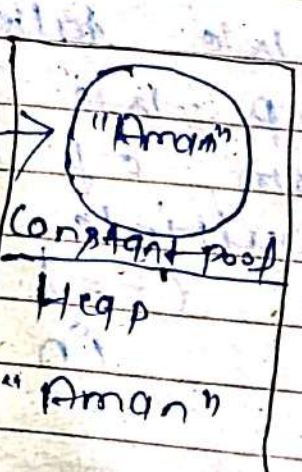


Modify the program

```
String name1 = new String ("Aman");
String name2 = "Aman");

o/p (name1 == name2)
```

Ans → false



Ques:- Constructor Overloading?

Ans → It creating Multiple Constructors in the class consisting of the same name with difference in the constructor parameters.

Ex! → class Hospitial {

int var1, var2;
double var3;

public Hospitial (int doctors, int nurses)
{
var1 = doctors
var2 = nurses
}

public Hospitial (int doctors) {
var1 = doctors;
}

public Hospitial (double salaries);
var3 = salaries
}

}

Ques:- Define Copy Constructor in Java?

Ans → Copy Constructor is the constructor used when we want to initialize the value to the new object from the old object of the same class.

Ex:- class Aman
{
String department;
String service;

Aman (Aman An);
this.department = An.department;
this.services = An.services;
}

Qus → Method overLoad?
Ans → Same Name but different parameter.

Qus → A single try block and Multiple catch blocks can co-exist in a Java prgm?
Ans → yes Multiple Catch block exist but specific approach should come prior to the general approach because only the first catch block satisfying the catch condition is executed.

Qus → use of Final variable. Keyword, Methods and class?
Ans → final variable:

* If any value is declared as Final in Java, the value can't be modified.
* If any value has not assigned to that variable, then it can be assigned only by the Constructor of the class.

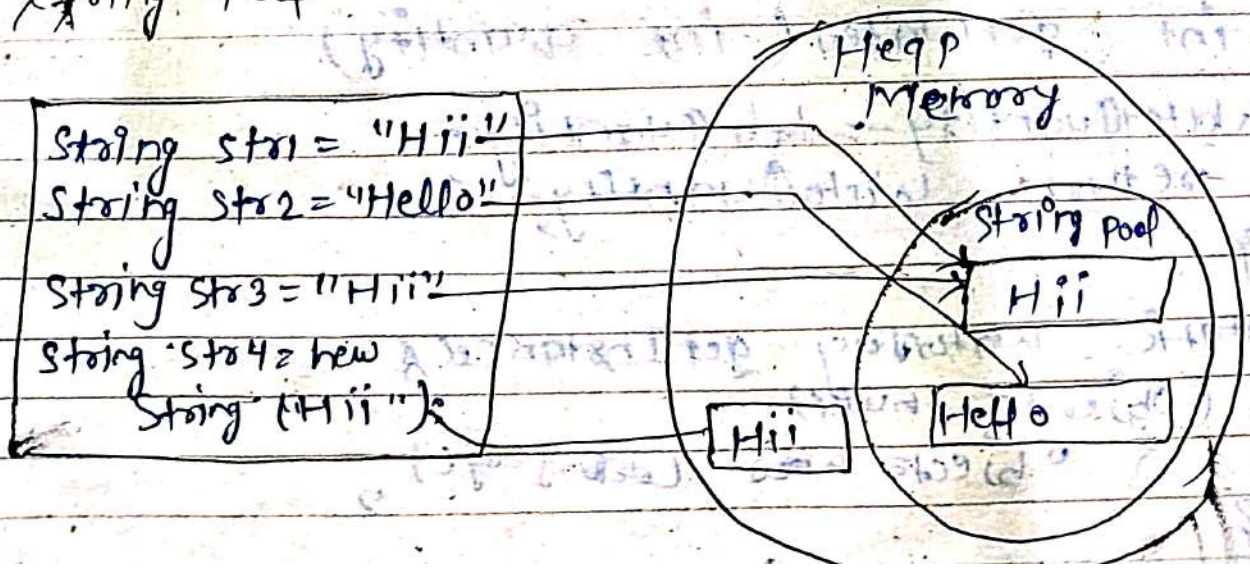
Final Method :- A Constructor declared as final can't marked as final because whenever a class is inherited, the constructor are invoked. Hence marking it final doesn't make sense. Java throws compilation error.

Final class → Final class can extend other class. for its usages

Q → What is classloader,
Ans → ✱ It belongs to JRE (JRE)
Ex :- To get the input from the Console, we require the scanner class. And the scanner class is loaded by the classloader.

Java interview Intermediate Question

Q → Apart from security Aspect, what are the reasons behind making strings immutable in Java;
Ans →
String Pool



Q→ What is singleton class in Java? And how to implement a singleton class?

Ans→ singleton class are those classes, whose objects are created only once, and with only that object the class members can be accessed.

understand this with the help of an example:-

Consider the Water Jug in the office and if every employee wants that water then they will not create a new water jug for drinking water. They will use the existing one with their own references as a glass. So implemented as-

```
class WaterJug {
private int WaterQuantity = 500;
private WaterJug() {}
private WaterJug object = null;

Public int getWater(int quantity)
{ WaterQuantity = WaterQuantity - quantity;
  return WaterQuantity;
}

Public static WaterJug getInstance()
{ if (object == null)
  { object = new WaterJug();
  }
  return object;
}
```

Q.4 → How would you differentiate b/w a String & StringBuffer and StringBuilder?

Ans →

Storage Area :- In String, the String pool serves as the storage area. For SB and SB, heap memory is the storage area.

Mutability :- A String is immutable, whereas both the SB & SB are Mutable.

Efficiency :- It is quiet slow to work with a String. StringBuilder is the fastest. The speed of a SBuffer is more than a StringBuilder.

Thread Safe :- In the case of threaded environment, StringBuilder and StringBuffer are used whereas a String is not used. SBuilder is not used. However, suitable for an environment with a single thread and a SBuffer is suitable for Multiple threads.

Q→ HashMap

* HashMap is not synchronized thereby making it better for non-thread applications.

* Allow only one null key but any number of null in the values.

* Supports Order of Insertion by making use of its subclass Linked HashMap.

HashTable

* It is synchronized. It is suitable for threaded app.

* This doesn't allow null values both keys and values.

* Order of Insertion is not guaranteed in HashTable.

Q→ What are the diff ways to thread usage?

Ans→ Define thread 2 ways.

1) Extending the Thread class.
2) Implementing the Runnable Interface.

Encapsulation :-

```
Class Employee
{
private int empId ; → Data Hiding
public void setEmpId ( int empId1)
    {
    emp_id = empId1 ;
    }
public void getEmpId ()
    {
    return empId ;
    }
```

1) Wrapping up of the data and code acting on that Data.

2) Wrapping up of variables and Methods in to a single unit is known as Encapsulation.

Abstraction :-

Abstraction	Encapsulation
Abstraction is detail hiding (Implementation hiding)	1) Encapsulation is data hiding (information hiding).
2) Data abstraction deals with exposing the Interface to the user and hiding the details of implementation.	2) Encapsulation groups together data and Methods that act upon the data.

Abstraction :- Real World Example

Ex in the car case, relevant parts like steering, gear, horn etc are known to driver because they are necessary for driving. But the driver need not know the any internal functionality of engine, gear etc.

We can achieve 2 ways.

1) Abstract class (0 - 100%)
2) Interfaces (100%)

Ex → abstract class Vehicle
{
int no. of tyre;
abstract void start();
}

* if Method is we will Create Method no body then we will add the word abstract
* When our Mehod is abstract then class will be abstract
* abstract Method can't have any body.

{
class Car extends Vehicle
{
void start()
{
C.O.P ("Starts with key");
}

* If a regular class extends an abstract class, then the class must have to implement all the abstract Methods.

(#) We can't create object of Abstract class.

Interface :-

```
Interface I1
{
// All Methods Abstract
}
```

* It is used to achieve abstraction.
* It supports Multiple Inheritance.
* It can be achieve loose Coupling.

Syntax :-

```
interface InterfaceName
{
Methods  // Abstract
         // public Abstract Method
Fields   // public static final
}
```

(3) 8 Version
(4) → default Concrete Methods
9th Version → static Method
→ Private Methods

Q4)- Method Overriding ?

Ans :- 1) Same Name.

a) different class

3) same argument
   → no. of arg.
   → type of arg.
   → sequence of arg

4) Inheritance

Interface Page Part Any

* We can not create Interface objects

```
interface I1
{
    public void show();
}
class Test implements I1
{
public void show()
  {
    S.o.P("1");
  }
}
P.S.V.M(String arr[])
{
Test t = new Test();
   t.show();
```

polymorphism :-

Many Forms

Types of polymorphism

1. Compile Time polymorphism
   → static polymorphism
   → Method overloading

2. Run time polymorphism
   → Dynamic polymorphism
   → Method overriding

Method overloading	Method overriding
1) Same Name	1) Same Name
2) Within same class	2) Different class
3) Different Argument → No. of argument → Type of arg. → seq. of arg.	3) Same argument → No. of arg. → Type of arg. → seq. of arg.
	4) In heritance (Is-A) Relationship

Method Overriding

```
class Test
{
    void show()
    {
        s.o.p("1");
    }
}
```

```
class Test1 extends Test
    {
        void show()
        {
            s.o.p("2");
        }
    }
p.s.v.m(String arr[])
    {
    Test t = new Test()
        t.show(); // 1
    Test1 t1 = new Test1()
        t1.show(); // 2
```

static keyword

static :-

Access Modifier	Non Access Modifier
Public	static
private	final
protected	abstract
default (No Modifier)	synchronized
	transient
	volatile
	strictfp
	Inner class

use

1) variable (class level) → local variable
2) Methods
3) Blocks
4) Inner class → (outer class X)
(nested class)

Ex :- class Test
{
static int a = 10;

void m1()
{
static int b = 20; //error
}
}

Ex:-

```
class Test
{
    static int a = 10;
}
class Demo
{
    P.S.V.M (String arr[])
    {
        S.o.p (Test.a);   // 10
    }
}
```

* static variable are used for memory Management

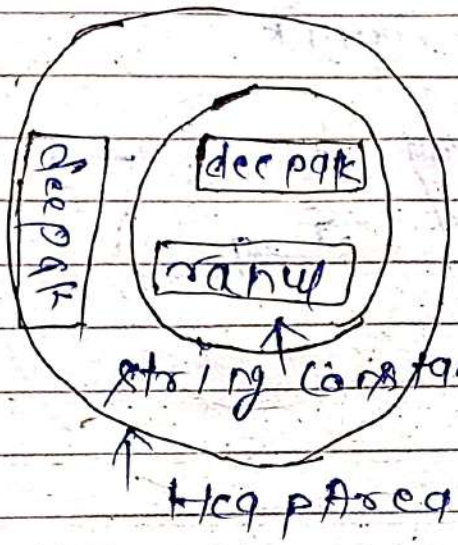
Diff:- String, StringBuffer, StringBuilder

String	StringBuffer	StringBuilder
Storage:-	1) Heap Area	1) Heap Area.
1) Heap Area, String Constant pool	2) Mutable	2) Mutable
Object:-	3) Consumes less Memory	3) Consumes less Memory.
2) Immutable	4) all Methods are Synchronized & it is thread safe	4) Non Synchronized Method Not thread safe.
Memory:-		
3) If we change value in String a lot of times it will allocate more memory.		
ThreadSafe:-		
4) Not		

string :-

```
String s1 = new String ("deepak");
String s2 = "rahul");
```

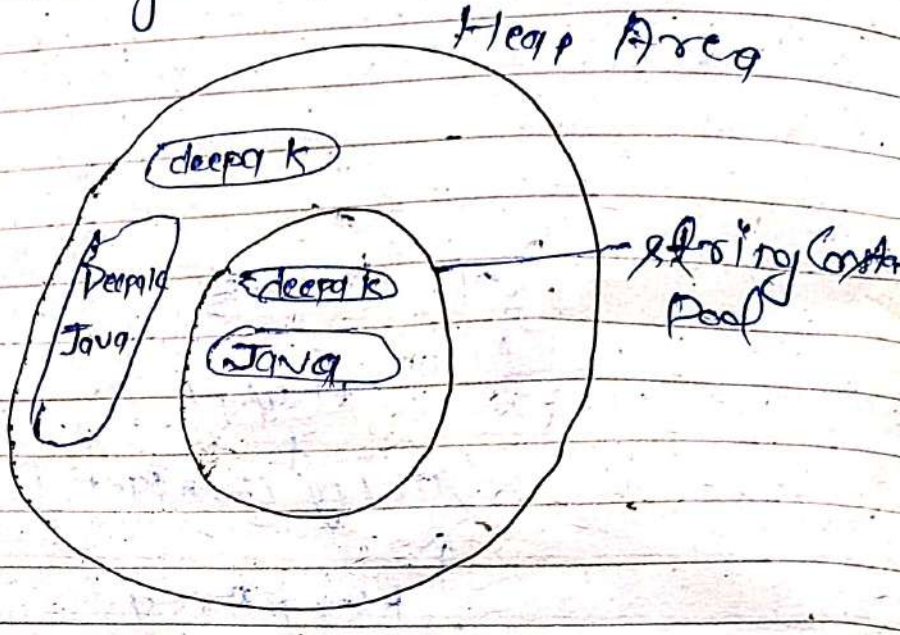
1) It will create 2 object
2) It will create 1 object.



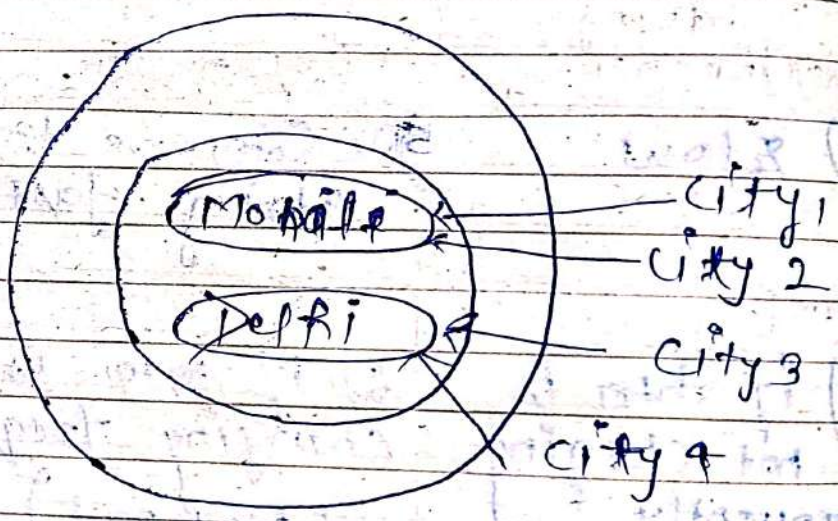
String	String Buffer	String Builder
Performance		
5) Slow	5) Compare to String fast	5) fast as Compared to String Buffer.
Use :-		
6) if data is not changing frequently. (Notepad)	6) if data is changing frequently. (Notepad)	6) change frequently. (Notepad)

Qu:- Why string Immutable
Ans:- Immutable means unchangable.

String s = new String ("deepak");
s.concat("Java");



Ex:-



String city1 = "Mohali".
String city2 = "Mohali".
City 3 = "delhi".
City 4 = "delhi";

Spring framework

* Beans is an object.
* IoC can achieved loosely achieved.
* It Reduce line of code.

Here is 2 class

BeanFactory ApplicationContext

* ApplicationContext is a superset of BeanFactory.

(Interface)
ApplicationContext

(Abstract class)
AbstractApplicationContext

(Class)
ClassPathXMLApplicationContext

Spring Boot Framework

Singleton pattern :-

* Object will create only one time.

* Container create the object when we will write @ Component in our class.

* @Qualifier.
It will search by the name

private Laptop laptop;

@Qualifier("laptop")
private Laptop laptop;

Java

Lambda Expression

Q→ What is Lambda Expression?

Ans→ Lambda is anonymous function (No Name) नामरहित।

* No Name
* No Modifier
* No Return Type

```
() →
{
   s.o.p("1");
}
```

Benefits of Lambda

* Reduces the lines of code.
* Sequential and Parallel execution support by passing behaviour as an argument in Methods.
* To call API's very effectively
* To write More Readable, Maintable and Concise Code.

How to write Lambda

```
() →
{
   s.o.p("1");
}
```

```
(int a, int b) →
{
   return a+b;
}
```

Important Rules of Lambda

* If the body of LE contain only one statement then curly braces are optional.

* Java Compiler also inner the type of variable passed in arg, hence type is optional.

```
public int getLength(String str)
{
    return str.length();
}
```

```
(str) -> str.length();
```

Functional Interface

If the interface Contains only one abstract Methods then it's Functional Interface.

Ex:- Runnable, Callable, Comparable etc.

* To call lambda we require Functional Interface.

* Lambda is used to implement functional interface in very simple and short Manner.

Ex:-

```
package work;

public class Main{
p.s.v.M(String ar[])
{
s.o.p("My System starts");
}
}
```

// create Interface

```
package work
@ functional Interface
public interface myInter{

public abstract void sayHello();

}
```

* If we declare functional then Interface we will use @FunctionInterfa

* Functional Interface only one abstra Method.

// create & create class and implemente interface.

```
Package work

public class myInterImpl implements
    myInter{
@Override
public void sayHello{
    s.o.p("I am saying Hello");
}
```

Main

```
public class Main {
    p.s.v.M(String arr[]) {
    s.o.p("My system stands");

    MyInter myInter = new MyInterImpl();
    myInter.sayHello();

// Anonymous class for implementing
   interface

    MyInter i = new MyInter() {
        s.o.p("first Anonymous");
    };
    i.sayHello();

// Lambda expression
// using our interface with the
      help of Lambda

    MyInter i = () -> {
        s.o.p("first");
    };

// }
    i.sayHello();
```

Remove Curly Braces.

```
MyInter i2 = () -> s.o.p("second")
    i2.sayHello();
```

Ex:-

```
public Interface SumInter{
    int sum(int a, int b);
}
// using Lambda
SumInter sumInter1=(int a, int b) -> {
    return a+b;
};
s.o.p(sumInter1.sum(a:2, b:6));
}
}
```

Second Rule

```
SumInter sumInter1=(a,b) -> a+b;
```

Java (1.8) ⇒ Stream API

* It Reduce the lines of code and improve performance.

Introduction to Stream

* These streams are related to collection frameworks (Group of objects). These streams are very different from io streams are the sequence of data.

* Stream API is basically bulk operations and process the objects of collection.

* Streams reduce the code length.

Ex1:-

```
public class StreamMain1 {
    p.s.v.m(String arg[])
// Create a list and filter all even numbers
List<Integer> list1 = List.of(2,4,6,50,21,22);
// Immutable List → List.of
List<Integer> list2 = Arrays.asList(12,34,23,78);

// without Stream
List<Integer> listEven = new ArrayList<>;
for(Integer i : list1)
{
    if(i%2==0)
    {
        listEven.add(i);
    }
}
s.o.p(listEven);
// using Stream API
1) list1.stream.filter(i -> i%2==0).collect
List<Integer> newList = (Collectors.toList());
2) list1.stream.filter(i -> i%2==0).collect
(collectors.toList);
s.o.p(newList);
```

(2) fo > Number
List<Integer> newList =
list.stream.filter(i -> i>50).i.collect
(Collectors.toList());
s.o.p (newList);

Q:- How to create stream object

public class streamObject {
P.S.V.M(String arr[])
{

// stream API - Collection process
// Collection - group of objects

☆ filter Means ide inbill work on boolean values : (return boolean value
☆ And Map Means ide inbill work on elements. (It return value)

Ex:- List<String> names = List.of("Aman", "Ankit", "Abhinav", "Durgesh");

names.stream.filter(e -> e.startsWith("A")).
collect(collectors.toList());

#:-

List<Integer> numbers = List.of
(23, 4, 2, 5, 7, 9);

(v) numbers.stream.map(i -> i*i).collect(Collectors.
toList);

# names.stream.forEach(e->{
s.o.p(e);
});

newNames.stream.forEach(System.out::println);

numbers.stream.sorted.forEach(System.out
::println);

# numbers.stream.min((x,y)->x.compare(y)).
get();
Integer integer1 =
# numbers.stream.max((x,y)->x.compare(y)).get();
s.o.p("Max!"+integer1);

(8)

Java Optional class

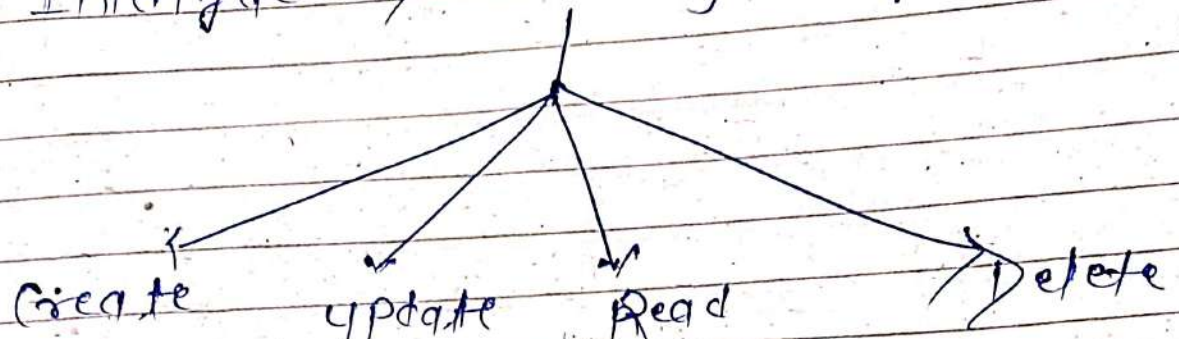
public class OptionalExample {
p.s.v.m(String arr[])
{

String str = null;

Optional<String> optional = Optional.ofNullable(str);
s.o.p(optional.isPresent()); //True
s.o.p(optional.orElse(other:"No object"));
// No object

* Entity Manager Factory (JPA) will Provide EntityManager object.

Interface → EntityManager



* Spring Boot makes easier to perform operation with JPA.

// Add the Dependency

~~spring-boot-starter-data-jpa~~

{ spring-boot-starter-data-jpa }

SPRING

Life cycle of spring Methods

* spring provide two important Methods to every bean.

public void init() → Initialization
public void destroy() → Destroy.

* We can change (init and destroy) name according to our name.

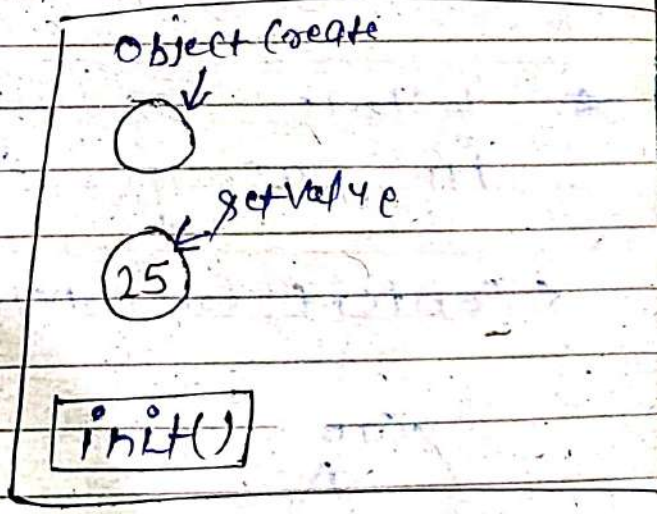
if we will load configuration File, Database connectivity, and Webservice connectivity and loading and initialization code

* When Bean will create then we can add here.

Destroy for cleanup Method.

① spring bean

② Configuration XML File



* Initialization after property setting.

Configure Technique

1) XML
2) Spring Interface
3) Annotation

Stereotype Annotation

* @Component
* It will create object at a Runtime.

```
@Component
class Student
{
}
```

* Spring container will create object student class.
* Where is my class we will tell in package name → XML file

```
<context:component-scan base-package="
```

scan

Package And Sub Package

@value

* put the name, id using @value="

* If we want to change My Bean name , will use

#@Component ("ob")

Hibernate Caxicading

Qus:- How Cascading works?

Ans:-

~~session s = fc~~

SessionFactory factory = new Configuration().configure().buildSessionfactory();

Session s = factory.openSession();

* if we want to save any Answer in Database.

I will use session object.

s.save(object);

#cascade = CascadeType.ALL

(1) if we will change parent it will automatically effect children class

~~cascade = CascadeType.persist~~

= save

Spring Boot

Annotation

@Qualifier inject Bean name:-

```
@Qualifier("temp3");
private Address address;
```

Java

Qus:- Diff. b/w JDK, JRE, JVM?

Ans:- JVM: (abstract)

* It is a Java Virtual Machine.
* It is called a Virtual Machie It doesn't exist.
* It is a specification that provides a runtime Environment.
* It Convert Bytecode can be executed.

It perform main tasks.

* Loads Code
* Verifies Code
* Executes Code
* Provide runtime environment

JRE:- * It is an acronym for Java Runtime Environment.

* It is a set of tools Software tools which are used to develop Java Application.
* It is the implementation of JVM.
* It physically Exist.
* Contains set of Libraries + other File.

JDK:- * It is a Java Development kit.

* It physically exist.
* It Contains JRE + development tools.
* It contains private (JVM) and a few other resources such as interpreter/loader.

Java variables

* 2 types of datatype

1) Primitive Datatypes
2) Non-primitive Datatypes

Types of Variable

* Local Variable
* Instance Variable
* Static Variable

* **Local Variable:** * Local Variable: declared inside the body of Method is called LV.
* It can't be defined with static keyword.

Instance variable:

* A variable declared inside the class but outside the body of Method.
* It is not declared as static.

① primitive → int, char, byte, short, long
float, Double

② Non-Primitive data type :-
Class, Interface, Array

operators in java :-

- ★ Unary operator
- ★ Arithmetic operator
- ★ Shift operator
- ★ Relation "
- ★ Bitwise "
- ★ Logical "
- ★ Ternary "
- ★ Assignment "
- ★

List of Java Keywords :-

1) abstract
2) boolean
3) break
4) byte
5) ~~class~~ case
6) catch
7) char
8) class
9) continue
10) default

Collections framework

Collection:- Any group of individual object which are represented as a single unit is known as collection of the objects.

Ex:- Collection of books (objects).

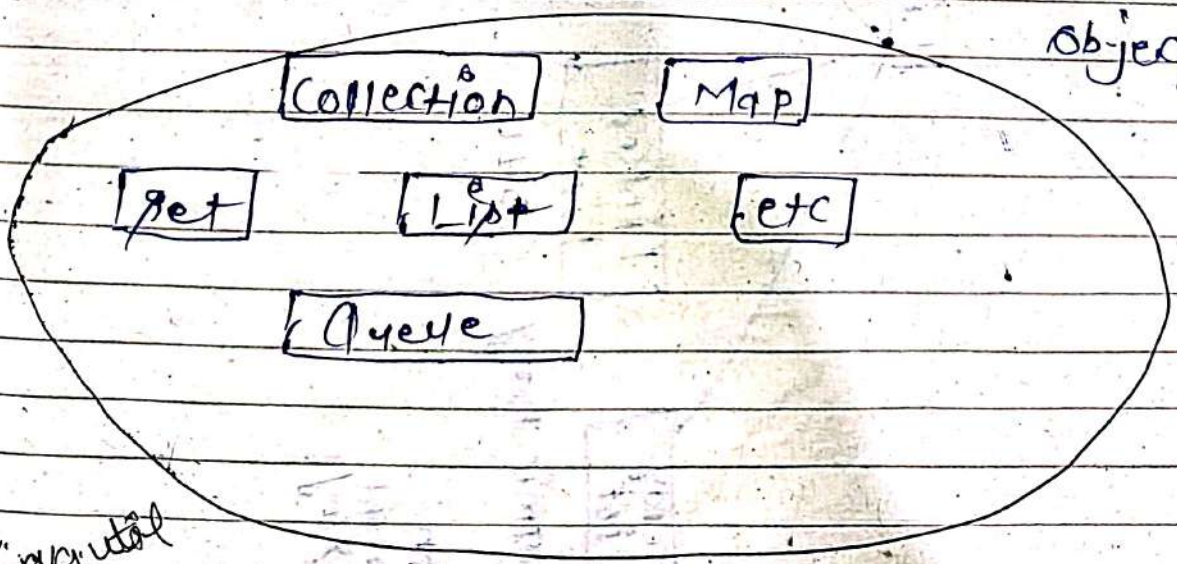
Framework:- A Framework is a set of classes and interfaces which provide a ready-made architecture.

Classes + Interfaces

Collection framework:-

Collection Framework is java API which provides architecture to store and manipulate group of objects.

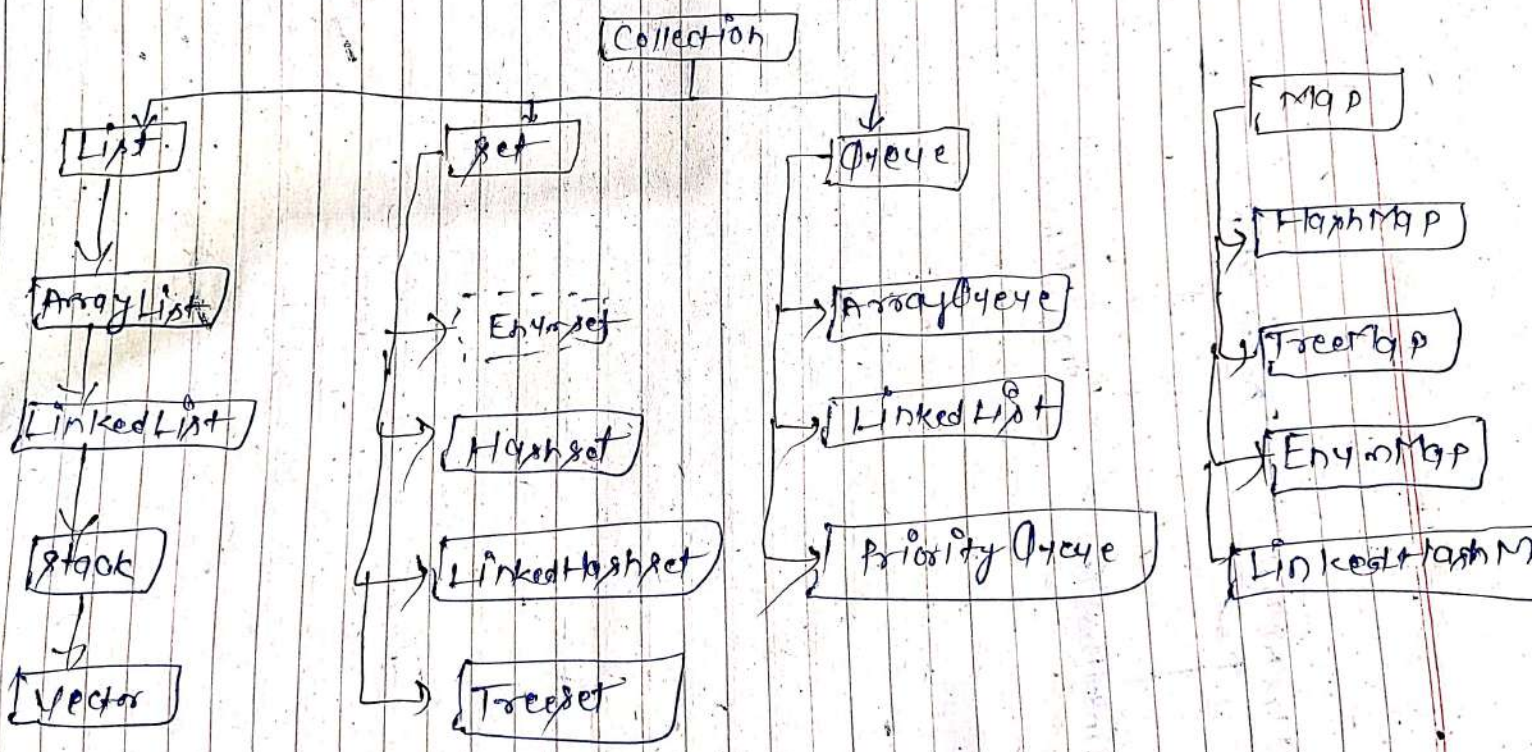
Class and Interface → store and Manipulation → group of objects



"javautil

Java Collections

Collection Hierarchy



Q.45 → Difference List and Set?

Ans →

List and Set both extend the Collection Interface.

List	Set
★ It contain duplicate element.	★ It contain unique elements.
★ List is an Ordered Collection which maintains the insertion Order.	★ Set is an unOrdered Collections.
★ List allow n: number of null values.	★ Set allows a single null value.

Q.46 → Diff. between TreeSet & HashSet?

Ans →

TreeSet	HashSet
★ It maintain ascending Order.	★ It maintains no Order.
★ It implemented by a Tree structure	★ It implement by hashtable.
★ It slower.	★ It fastey
★ It TreeSet is backed by TreeMap.	★ Backed by HashMap.
	★

Qus → Diff. Set and Map?

Ans:

Set	Map
★ Set Contain Value	★ Map Contains key and value both.
★ unique Values.	★ unique key with duplicate Values.
★ It holds a single number of null Values.	★ Map include a single null key with n number of null Values.

Qus:- Diff. Hashset and HashMap?

Ans:-

Hashset	HashMap
★ Contain Value.	★ Contain key, Value.
★ Implement Set Interface.	★ Map Interface.
★ Hashset can not have any duplicate Values.	★ Can Contain duplicate Values with unique keys.
★ single number of null Value	★ single null key diff. null key with n number of null Values.

Ques → Diff. HashMap & TreeMap?

Ans →

HashMap	TreeMap
★ It maintains No order.	★ It maintains ascending order.
★ It implemented by Hashtable.	★ It implemented by a Tree structure.
★ It Can be sorted by key or Value.	★ sorted by key.
★ Contain a null keys with Multiple values.	★ TreeMap can't hold a null key but can have multiple null Values.

Ques → Diff. HashMap and Hashtable?

Ans :-

HashMap	HashTable
★ HashMap is not synchronized.	★ synchronized.
★ HashMap can Contain one null key and multiple null Values.	★ HashTable Can't Contain any null key or null Value.
★ Not threadsafe.	★ threadsafe.
★ HashMap inherits the AbstractMap Class.	★ Hashtable inherits the Dictionary Class.

Que → What is the serialization?
Ans → Serialization is the process for converting the object in to stream.

Que → What is Deserialization?
Ans → Deserialization is the reverse process of serialization it convert the Byte Code in to object and save.

Serialization
↓
Object
↓ Convert
stream
↓ Deserialized
stream
↓ Convert
object

OneEnd ← → Destination End

☆ Implements Serializable Interface.

Serizable :- It is an interface
It is in java.io package
if any those class implements serializable interface in those class that part of the serialization.

☆ It is a Marker Interface.

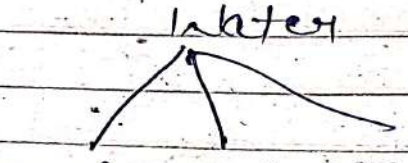
Ques? What is Marker Interface?
Ans? No Any Method Available

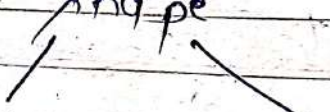
Ques? OOPs Concept

Polymorphism :-

* polymorphism means many forms.
* Ability of an object to behave differently.

Ex:-





Types

1. Compile Time polymorphism
   - Achieve
     - Static polymorphism
     - Method overloading
2. Run Time polymorphism
   - Achieve
     - Dynamic polymorphism
     - Method overriding

* Compile time polymorphism - handle the Compiler.

* Run Time polymorphism - handle the JVM.

Qu:- What is Inheritance

Ans → * It is inheriting the properties of parent class in to child class.

Ex:- class Animal
{ void eat() }
{ S.o.p("Eating"); }

class Dog extends Animal {
P.S.V.M (String ar[]) {
{ Dog d = new Dog();
d.eat();
} } } is a Animal

= Is-A Relationship.

* Spring Container autowires the bean by matching data type.

Que → * Spring Boot Annotations?
Ans →

* Autowired :- It is used to autowire spring Bean on Setter Method, Instance variable, and Constructor.

* 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 throwh an exception BeanInitializationException.

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

* @ComponentScan :- It is used when we want to scan a package for Beans. It is used with the annotation @Configuration.

* @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.

@Component :- It is a class level annotation. The class annotated with @Configuration

★ @Component :- It is a class level annotation. It is used to mark as a Java class as a Bean. A Java class annotated with @Component is found during the classpath.

★ @Controller :- The @Controller is a class level annotation. It is a specialization of @Component It marks a class as a web request handler.

★ @Service :- It is also used at class level. It tells the spring that class contains business logic.

★ @Repository :- It is a class level annotation. The Repository is a DAO's (Data Access object) that access the database directly. The repository does all the operations related to the database.

★ @EnableAutoConfigure :- It autoConfigure the bean that is present in the classpath and Configures it to run the

Que → Difference b/w @Controller and @RestController?

Ans →

@Controller	@RestController
* @Controller is used to marks classes as Spring MVC Controller.	* This annotation is a special Controller used in Restful web services, and it's the combination of @Controller and @ResponseBody annotation.
* It is a specialized version of @Component annotation.	* It is a specialized version of @Controller annotation.
* We can return a view in spring Web MVC.	* We can not Return a view.
* @Controller annotation indicates that the class is a "Controller" like a web container.	* @RestController annotation indicates that class is a Controller where @RequestMapping Methods assume @ResponseBody semantic by default
* We need to use @ResponseBody on every handler.	* We don't need to use @ResponseBody on every Method.
* It was added to spring 2.5 version.	* It was added to spring 4.0 version.

Que: What is Marker Interface in Java?

Ans → An interface that does not contain Methods, fields and constants is known as Marker Interface. In other word, an Empty Interface is known as Marker Interface.

Que → What is serialization?

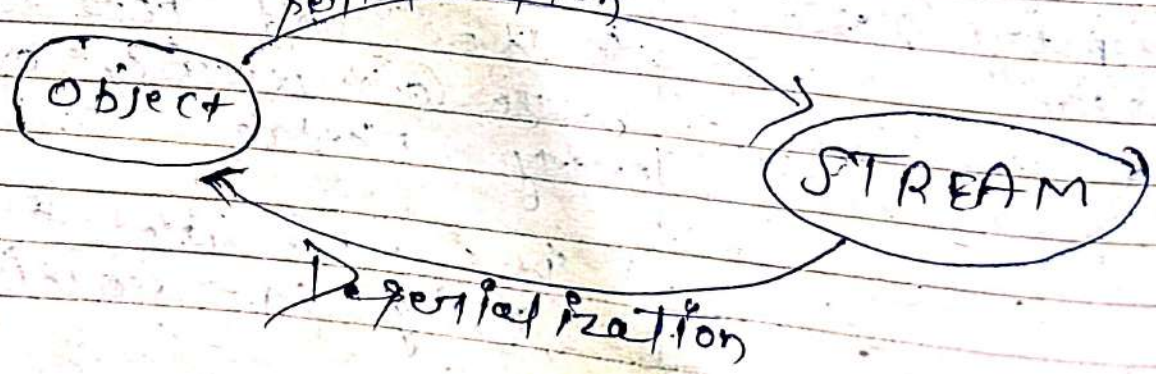
Ans → Serialization in java is a mechanism of writing the state of object in to a byte stream. It is mainly used in Hibernate, RMI, JPA and JMS technologies.

Que → What is Deserialization?

Ans → The reverse operation of serialization is called Deserialization.

Advantages of Java serialization

* It is mainly used to travel object's state on the network (that is known as Marshalling).



Ques→ What is Java Threads?

Ans→ Threads allows a program to operate more efficiently by doing multiple things at the same Time.

Threads can be used to perform complicated tasks in the background without interrupting the main program.

Creating a Thread

* 2 ways to create a thread.

1) Thread class
2) Runnable Interface

1) Thread class → * It can be created by extending the Thread class and Overriding its run() Method.

Ex→ public class Main extends Thread {
    public void run() {
        S.o.p("Running in Thread");
    }
}

2) Runnable Interface: * Create a thread * is to Implement the Runnable interface.

Ex→ public class Main implements Runnable {
    public void run() {
        S.o.p("Running in thread");
    }
}

Qus→ Exception & Error ?

Ans →

Exception → ★ It is an occurs during the program and Interrupts the normal flow of program instruction. These are the errors that occurs at the compile time and run time. Ex

★ It recovered by using try and catch blocks and throws keyword.

Error → ★ Errors are problem that mainly occur that due to the lack of system resources.

★ It can not be caught or handled.

★ It occurs at RunTime.

Ex→ Out of Memory Error, LinkageError.

Qus→ Difference First level cache & second level cache)

Ans→

First Level Cache	Second Level Cache
★ It is associated with session.	★ It is associated with session Factory.
★ It is enabled by default.	★ It is not Enabled by default.

Ques → Difference get and load Method?

Ans →

get()	load()
★ It Returns null if an object is not found.	★ It throws Object NotFoundException if an Object is not found.
★ It always hit the database.	★ It does not hit the database.
★ It return the real object not the proxy.	★ It return the proxy object.
★ It should be used if you are not sure about the existence of instance.	★ It should be used if you are sure that instance exist

Ques → What is session?

Ans → ★ It maintain a connection b/w Hibernate application and database.

★ It provides methods to store, update, delete or fetch data from the database.

★ It is a factory of Query, Criteria and Transaction.

Qus→ What is session factory?

Ans→ ★ Sessionfactory provides the instance of session.

★ It is a factory of session.

★ It holds the data of second level cache that is not enabled by default.

Qus→ Cache in Hibernate?

Ans→ Hibernate caching improves the performance of the application by pooling the object in the cache. It is useful when we have to fetch the same data multiple Times.

There are 2 types of caching.

★ First level cache

★ Second level cache

★ First level cache :-

Session object holds the first level cache data. It is enabled by default. The First level cache data will not be available to entire application. An application use Many session object.

★ Second level cache :- Session factory object holds the second level cache data. The data stored in the second level cache will be available to entire application But we need to enable it explicitly

Q Qus → What is Lazy Loading in Hibernate?

Ans → ★ Lazy loading in hibernate improves the performance.
★ It loads the child object on demand.
Since Hibernate 3, Lazy loading is enabled by default, and you don't need to do lazy = "true"
It means to load the child objects when the parent obj is loaded.

Qus → What is default scope of Bean?

Ans → The default scope of bean is a singleton. Singleton means that the Spring Container creates only one instance of the bean, and cached in the Memory and all the requests of the bean will return a shared reference to the same bean.

Qus → Use of Autowired?

Ans → ★ Spring provides annotation based autowiring by providing
① Autowired annotation.
★ It is used to autowire spring bean on setter Method, instance variable, and Constructor.
★ When we use ① Autowired annotation the Spring Container autowires the bean by matching data type.

Que→ What are the Methods to create beans in spring?

Ans→ Creates a spring Bean in 3 diff. ways.

* Creating bean inside an XML configuration File (beans.xml).
* Using @Bean Annotation.

Que→ Spring MVC and Rest Annotations?

Ans→

① RequestMapping → * It is used to Map the Web requests. It has many optional elements like Consumes, header, methods, name and value.

Qus → What is Interface?

Ans → An Interface in java is a blueprint of a class. It has static constants and abstract Methods. Interface in java is a Mechanism to achieve Abstraction. There can be only abstract Methods in Java interface, not method body. It is used to achieve abstraction and Multiple Interface in Java.

* Since Java 8, We can have default and static Method.
* Java 9, Can have private Methods in an interface.

Diff.

Interface	Abstraction
* Only abstract Methods.	* It can have Abstract & Non Abstract Method.
* Support Multiple Inheritance.	* It doesn't support Multiple Inheritance.
* static and final variable.	* Abstract class can have Final, non-Final, static and non static Method.
* Can't provide the implementation of abstract class.	* It can provide the implementation of interface
* Interface can extend another Java interface only.	* abstract keyword used to declare Abstract class.
* implemented by implements	

Qus → What is Abstraction?
Ans → It is a feature of OOPs. It is used to hide the unnecessary information or data from the user but shows the essential data that is useful for the user. It can be achieved by using Interface and Abstract Class.

Ex:- Remote → The user only interact with the outer Interface that is nothing but keys. User only knows which key to press for what function.

Abstract Child class → extends

Qus → What is Encapsulation?
Ans → It is also a feature of OOPs. It is used to bind up the data into a single unit called Class. It provides the mechanism which is Data Hiding. It is an Imp. features of OOPs. It prevents to access data members from the outside the class. It is also necessary from the security point of the view.

Ex → private variable → access getter and setter

Q →

Diff

Abstraction	Encapsulation
★ Feature of OOPs. Hides the unnecessary details. Show essential Info. It solves an issues at design level. Focus on External lookout. Implemented by Abstract class and Interface.	★ Feature of OOPs. It hides the code data into a single entity or unit so that the data can be protected outside the world.
★ It is the process of gaining Information.	★ Implementation level. ★ Focus on internal working. ★ Implemental by access Modifier (Public, Prote., Pri.). ★ It is the process of containing Information.

Fibonacci series

```
class Fibonacci
P.S.V.M(String arr[]){
int a=0, b=1, c, num=10;
S.O.p (a+" "+b);
for(i=2; i<num; i++)
{
c=a+b;
S.O.p(" "+c);
a=b;
b=c;
}
```

Palindrom number

```
int r, sum=0, temp;
int n=454;
temp=n;
while(n>0){
r=n%10;
sum=(sum*10)+r;
n=n/10;
}
if(temp==sum)
S.O.p("yes");
else
S.O.p("no");
```

Find Duplicate in String

```
String str = "beautiful beach";
char[] carray = str.toCharArray();
for(int i=0; i<str.length(); i++){
for(int j=i+1; j<str.length(); j++){
if(carray[i]==carray[j]){
S.O.p(carray[j]+" ");
break;
}
```

```
class Fibonacci{
int
o o
class Fibonacci{
static int n1=0, n2=1, n3=0;
if(count>0){
static void printFibo(int count)
{
if((count>0){
n3=n1+n2;
n1=n2;
n2=n3;
S.O.p(" "+n3);
printFibo(count-1);
}
}
P.S.V.M(String arr[]){
int num=10;
S.O.p(n1+" "+n2);
printFibo((count-2);
}
}
```

Print Unique in String

```
String S="Hello World";
char a[]=S.toCharArray();
List duplicates=new ArrayList();
List uniqueElements=new ArrayList();
for(int i=0; i<a.length; i++){
uniqueElements.add(a[i]);
for(int j=i+1; j<a.length; j++)
if(a[i]==a[j]){
duplicates.add(a[i]);
break;
}}}
uniqueElements.removeAll(dupl);
S.O.p(
```

Swap Number in Java

```
① class SwapNumber{
P.S.V.M(String arr[]){
int x,y;
S.O.p("Enter X and Y");
Scanner sc=new Scanner();
x&y = sc.nextInt();
S.O.p("Before Swap"+x+y);
x=x+y;
y=x-y;
x=x-y;
S.O.p(+x" "+y);
}
```

Reverse String

```
① String name="Aman";
char[] reverseName=name.toCharArray();
for(int i=reverseName.length-1; i>=0; i--)
{
S.O.p(reverseName[i]);
```

```
② String name="Aman";
String revName="";
for(int i=name.length()-1; i>=0; i--)
revName=revName+name.charAt(i);
}
S.O.p(revName);
```

Armstrong In Java

```
153 = 1*1*1 + 5*5*5 + 3*3*3
153 = 153
public class Armstrong
{
int no=371, origi=number
,remainder, result=0;
originalNumber=num
while(origNum!=0)
{
rem=org.Num%10;
res=Math.Pow(rem,3);
Org.Num/=10;
}
if(result==number)
S.O.p(Armst
else{
rad)
```