GenSpark Java Interview Guide

Interview guide includes following technologies:

Core Java

Spring Framework

Spring-boot

JPA

Hibernate

Thymeleaf

- NOTE: You're not expected to know every single question at this point try to cover as much as you can.

- References: Questions has been taken from various sources from internet. GenSpark isn't an author of this questions. It is solely used for study & interview preparation purpose.

1. What is Java?

Java is a general-purpose programming language that is class-based, object-oriented and is very popular. It's one of the most popular programming languages in the world.

Hello World in Java:

```
public class FileName {
  public static void main(String args[]) {
    System.out.println("Hello World!");
  }
}
```

2. How to install Java?

Install Java through command prompt so that it can generate necessary log files to troubleshoot the issue.

Go to java.com and click on the Free Java Download button.

Click on the Save button and save Java software on the Desktop

Verify that Java software is saved on the desktop.

Open Windows Command Prompt window.

Windows XP: Click Start -> Run -> Type: cmd

Windows Vista and Windows 7: Click Start -> Type: cmd in the Start Search field.
cd <Java download directory> (for example Downloads or Desktop etc.)

IRun the installer and follow onscreen instructions.

3. How to reverse a string in Java?

```
"String str = ""Hello"";

String reverse(String str){

  StringBuilder sb = new StringBuilder();

  sb.append(str);

  sb.reverse();

  return sb.toString();

}"
```

4. What is thread in Java?

Threads allow 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.

It can be created by extending the Thread class and overriding its run() method:

Extend Syntax

```
public class MyClass extends Thread {
  public void run() {
    System.out.println("This code is running in a thread");
  }
}
```

5. How to take input in Java?

```
"Scanner in = new Scanner(System.in);

      System.out.print(""Please enter hour 1: "");

      int hour1 = in.nextInt();

      System.out.print(""Please enter hour 2: "");

      int hour2 = in.nextInt();

      System.out.print(""Please enter minute 1: "");

      int min1 = in.nextInt();

      System.out.print(""Please enter minute 2: "");

      int min2 = in.nextInt();"
```

6. How to set path in Java?

Windows 10 and Windows 8

- In Search, search for and then select: System (Control Panel)
- Click the Advanced system settings link.
- Click Environment Variables. In the section System Variables, find the PATH environment variable and select it. Click Edit. If the PATH environment variable does not exist, click New.
- In the Edit System Variable (or New System Variable) window, specify the value of the PATH environment variable. Click OK. Close all remaining windows by clicking OK.
- Reopen Command prompt window, and run your java code.

Mac OS X

To run a different version of Java, either specify the full path or use the java_home tool:

% /usr/libexec/java_home -v 1.8.0_73 –exec javac -version

Solaris and Linux

To find out if the path is properly set:
In a terminal windows, enter:
% java -version
This will print the version of the java tool, if it can find it. If the version is old or you get the error java: Command not found, then the path is not properly set.
Determine which java executable is the first one found in your PATH
In a terminal window, enter:
% which java

7. What is enumeration in Java?

Enumeration means a list of named constant. In Java, enumeration defines a class type. An Enumeration can have constructors, methods and instance variables. It is created using enum keyword. Each enumeration constant is public, static and final by default. Even though enumeration defines a class type and have constructors, you do not instantiate an enum using new. Enumeration variables are used and declared in much a same way as you do a primitive variable.

8. What is inheritance in Java?

The process by which one class acquires the properties(data members) and functionalities(methods) of another class is called inheritance. The aim of inheritance is to provide the reusability of code so that a class has to write only the unique features and rest of the common properties and functionalities can be extended from another class.

Child Class:
The class that extends the features of another class is known as child class, sub class or derived class.

Parent Class:
The class whose properties and functionalities are used(inherited) by another class is known as parent class, super class or Base class.

9. How to compare two strings in Java?

```
"// These two have the same value

new String(""test"").equals(""test"") // --> true



// ... but they are not the same object

new String(""test"") == ""test"" // --> false



// ... neither are these

new String(""test"") == new String(""test"") // --> false
```

```
// ... but these are because literals are interned by

// the compiler and thus refer to the same object

""test"" == ""test"" // --> true "
```

10. What is abstraction in Java?

Objects are the building blocks of Object-Oriented Programming. An object contains some properties and methods. We can hide them from the outer world through access modifiers. We can provide access only for required functions and properties to the other programs. This is the general procedure to implement abstraction in OOPS.

11. What is encapsulation in java

The idea behind encapsulation is to hide the implementation details from users. If a data member is private it means it can only be accessed within the same class. No outside class can access private data member (variable) of other class.

However if we setup public getter and setter methods to update (for example void setName(String Name))and read (for example String getName()) the private data fields then the outside class can access those private data fields via public methods.

12. What is collection in java?

Collections are like containers that group multiple items in a single unit. For example, a jar of chocolates, list of names, etc.

Collections are used in every programming language and when Java arrived, it also came with few Collection classes – Vector, Stack, Hashtable, Array.

13. What is api in java?

Java application programming interface (API) is a list of all classes that are part of the Java development kit (JDK). It includes all Java packages, classes, and interfaces, along with their methods, fields, and constructors. These pre-written classes provide a tremendous amount of functionality to a programmer.

14. How to initialize array in java?

```
"int[] arr = new int[5];  // integer array of size 5 you can
also change data type

String[] cars = {""Volvo"", ""BMW"", ""Ford"", ""Mazda""};"
```

15. How to take input from user in java?

```
"import java.util.Scanner;

  Scanner console = new Scanner(System.in);

  int num = console.nextInt();
```

```
  console.nextLine() // to take in the enter after the
nextInt()

  String str = console.nextLine();"
```

16. What is static in java?

In Java, a static member is a member of a class that isn't associated with an instance of a class. Instead, the member belongs to the class itself. As a result, you can access the static member without first creating a class instance.

17. What is package in java?

A package in Java is used to group related classes. Think of it as a folder in a file directory. We use packages to avoid name conflicts, and to write a better maintainable code. Packages are divided into two categories:

Built-in Packages (packages from the Java API)
User-defined Packages (create your own packages)

18. How to sort an array in java?

```
"import java. util. Arrays;

Arrays. sort(array);"
```

19. What is an abstract class in java?

A class that is declared using the “abstract” keyword is known as abstract class. It can have abstract methods(methods without body) as well as concrete methods (regular methods with body). A normal class(non-abstract class) cannot have abstract methods.

20. What is method in java?

A method is a block of code which only runs when it is called. You can pass data, known as parameters, into a method. Methods are used to perform certain actions, and they are also known as functions.

21. How to check java version?

Execute java -version on a command prompt/terminal.

22. What is a class in java?

A class–the basic building block of an object-oriented language such as Java–is a template that describes the data and behaviour associated with instances of that class. When you instantiate a class you create an object that looks and feels like other instances of the same class. The data associated with a class or object is stored in variables; the behaviour associated with a class or object is implemented with methods.

23. What is core java?

“Core Java” is Sun’s term, used to refer to Java SE, the standard edition and a set of related technologies, like the Java VM, CORBA, et cetera. This is

mostly to differentiate from, say, Java ME or Java EE. Also, note that they're talking about a set of libraries rather than the programming language.

24. How to enable java in chrome?

- In the Java Control Panel, click the Security tab
- Select the option Enable Java content in the browser
- Click Apply and then OK to confirm the changes
- Restart the browser to enable the changes

25. What is string in java?

String is a sequence of characters, for e.g. "Hello" is a string of 5 characters. In java, string is an immutable object which means it is constant and cannot be changed once it has been created.

26. What is exception in java?

An exception is an event, which occurs during the execution of a program, that disrupts the normal flow of the program's instructions.
When an error occurs within a method, the method creates an object and hands it off to the runtime system. The object, called an exception object, contains information about the error, including its type and the state of the program when the error occurred. Creating an exception object and handing it to the runtime system is called throwing an exception.

After a method throws an exception, the runtime system attempts to find something to handle it. The set of possible "somethings" to handle the exception is the ordered list of methods that had been called to get to the

method where the error occurred. The list of methods is known as the call stack.

27. Why multiple inheritance is not supported in java?

Java supports multiple inheritance through interfaces only. A class can implement any number of interfaces but can extend only one class. Multiple inheritance is not supported because it leads to a deadly diamond problem.

28. How to take string input in java?

```
"import java.util.Scanner;  // Import the Scanner class



class MyClass {

  public static void main(String[] args) {

    Scanner myObj = new Scanner(System.in);  // Create a
Scanner object

    System.out.println(""Enter username"");



    String userName = myObj.nextLine();  // Read user input

    System.out.println(""Username is: "" + userName);  //
Output user input

  }

}"
```

29. What is singleton class in java?

The singleton design pattern is used to restrict the instantiation of a class and ensures that only one instance of the class exists in the JVM. In other words, a singleton class is a class that can have only one object (an instance of the class) at a time per JVM instance.

30. What is array in java?

An array is a container object that holds a fixed number of values of a single type. The length of an array is established when the array is created. After creation, its length is fixed. You have seen an example of arrays already, in the main method of the “Hello World!” application. This section discusses arrays in greater detail.

Illustration of an array as 10 boxes numbered 0 through 9; an index of 0 indicates the first element in the array.

An array of 10 elements.
Each item in an array is called an element, and each element is accessed by its numerical index. As shown in the preceding illustration, numbering begins with 0. The 9th element, for example, would therefore be accessed at index 8.

31. What is garbage collection in java?

Java garbage collection is an automatic process. The programmer does not need to explicitly mark objects to be deleted. The garbage collection implementation lives in the JVM. Each JVM can implement garbage collection however it pleases; the only requirement is that it meets the JVM

specification. Although there are many JVMs, Oracle's HotSpot is by far the most common. It offers a robust and mature set of garbage collection options.

32. Why we need encapsulation in java?

Encapsulation in Java is a mechanism of wrapping the code and data (variables)acting on the data (methods) together as a single unit. In encapsulation, the variables of a class will be hidden from other classes and can be accessed only through the methods of their current class.

33. What is jvm in java?

A Java virtual machine (JVM) is a virtual machine that enables a computer to run Java programs as well as programs written in other languages that are also compiled to Java bytecode. The JVM is detailed by a specification that formally describes what is required in a JVM implementation.

34. What is java programming?

Java is a powerful general-purpose programming language. It is used to develop desktop and mobile applications, big data processing, embedded systems, and so on. According to Oracle, the company that owns Java, Java runs on 3 billion devices worldwide, which makes Java one of the most popular programming languages.

35. How hashmap works internally in java?

HashMap in Java works on hashing principles. It is a data structure which allows us to store object and retrieve it in constant time O(1) provided we know the key. In hashing, hash functions are used to link key and value in HashMap.

36. Who invented java?

Java was originally developed by James Gosling at Sun Microsystems (which has since been acquired by Oracle) and released in 1995 as a core component of Sun Microsystems' Java platform.

37. How to execute a java program?

Open a command prompt window and go to the directory where you saved the java program (HelloWorld. java). …
Type 'javac HelloWorld. java' and press enter to compile your code.
Now, type ' HelloWorld ' to run your program.
You will be able to see the result printed on the window.

38. How to get input from user in java?

```
"import java.util.Scanner;  // Import the Scanner class



class MyClass {

  public static void main(String[] args) {

    Scanner myObj = new Scanner(System.in);  // Create a
Scanner object
```

```
    System.out.println(""Enter username"");



    String userName = myObj.nextLine();  // Read user input

    System.out.println(""Username is: "" + userName);  //
Output user input

  }

}"
```

39. What is bytecode in java?

Bytecode is the compiled format for Java programs. Once a Java program has been converted to bytecode, it can be transferred across a network and executed by Java Virtual Machine (JVM). Bytecode files generally have a .class extension.

40. How to set classpath in java?

- Select Start, select Control Panel, double click System, and select the Advanced tab.
- Click Environment Variables. In the section System Variables, find the PATH environment variable and select it.
- In the Edit System Variable (or New System Variable) window, specify the value of the PATH environment variable. Click OK.

41. How to connect database in java?

- Install or locate the database you want to access.
- Include the JDBC library.
- Ensure the JDBC driver you need is on your classpath.
- Use the JDBC library to obtain a connection to the database.
- Use the connection to issue SQL commands.

42. What is enum in java?

An enum is a special “class” that represents a group of constants (unchangeable variables, like final variables). To create an enum, use the enum keyword (instead of class or interface), and separate the constants with a comma.

43. How to uninstall java?

- Click Start
- Select Settings
- Select System
- Select Apps & features
- Select the program to uninstall and then click its Uninstall button
- Respond to the prompts to complete the uninstall

44. How to find duplicate characters in a string in java?

```
"public class Example {

   public static void main(String argu[]) {

      String str = ""beautiful beach"";

      char[] carray = str.toCharArray();

      System.out.println(""The string is:"" + str);

      System.out.print(""Duplicate Characters in above string
are: "");

      for (int i = 0; i < str.length(); i++) {

         for (int j = i + 1; j < str.length(); j++) {

            if (carray[i] == carray[j]) {

               System.out.print(carray[j] + "" "");

               break;

            }

         }

      }

   }

}"
```

45. How to take character input in java?

```
"import java.util.Scanner;

public class CharacterInputExample1

{

public static void main(String[] args)

{

Scanner sc = new Scanner(System.in);

System.out.print(""Input a character: "");

// reading a character

char c = sc.next().charAt(0);

//prints the character

System.out.println(""You have entered ""+c);

}

}  "
```

46. how to read string in java?

```
"import java.util.Scanner;  // Import the Scanner class



class MyClass {

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

```
    Scanner myObj = new Scanner(System.in);  // Create a
Scanner object

    System.out.println(""Enter username"");



    String userName = myObj.nextLine();  // Read user input

    System.out.println(""Username is: "" + userName);  //
Output user input

  }

}"
```

47. How to round off numbers in java?

```
"import java.lang.Math; // Needed to use Math.round()



class Program {

    public static void main( String args[] ) {

      double num1 = 74.65;

      System.out.println(Math.round(num1));



      float num2 = 1337.345523f;

      System.out.println(Math.round(num2));

    }

}"
```

48. How to get current date in java?

```
"DateFormat df = new SimpleDateFormat(""dd/MM/yy HH:mm:ss"");

Date dateobj = new Date();

System.out.println(df.format(dateobj));"
```

49. What is dao in java?

Dao is a simple java class which contains JDBC logic. The Java Data Access Object (Java DAO) is an important component in business applications. Business applications almost always need access to data from relational or object databases and the Java platform offers many techniques for accessing this data.

50. What is awt in java?

The Abstract Window Toolkit (AWT) is Java's original platform-dependent windowing, graphics, and user-interface widget toolkit, preceding Swing. The AWT is part of the Java Foundation Classes (JFC) — the standard API for providing a graphical user interface (GUI) for a Java program. AWT is also the GUI toolkit for a number of Java ME profiles. For example, Connected Device Configuration profiles require Java runtimes on mobile telephones to support the Abstract Window Toolkit.

51. What is framework in java?

Frameworks are large bodies (usually many classes) of prewritten code to which you add your own code to solve a problem in a specific domain. Perhaps you could say that the framework uses your code because it is usually the framework that is in control. You make use of a framework by calling its methods, inheritance, and supplying “callbacks”, listeners, or other implementations of the Observer pattern.

52. How to update java?

Manually updating Java on Windows is typically done through the Java Control Panel.

Windows 10: Type “java” into the Windows/Cortana search box, located in the lower left-hand corner of your screen. When the pop-out menu appears select Configure Java, located in the Apps section.

53. How to run java program in command prompt?

```
execute java <programName.java>
```

54. What is variable in java?

A Java variable is a piece of memory that can contain a data value. A variable thus has a data type. Data types are covered in more detail in the text on Java data types. Variables are typically used to store information which your Java program needs to do its job.

55. What is the difference between java and javascript?

The main differences between JavaScript and Java are:

1. JavaScript is used for Front End development while java is used for Back End Development. i.e.

JavaScript is responsible for the dynamic behaviour of a webpage. Mainly, JavaScript handles events, cookies, ajax (Asynchronous JavaScript and XML), etc. in a website. JavaScript is the heart of a Dynamic User Interface of a Web Page while Java is the best programming language for software engineers and can be used with JSP (Java Server pages) for handling back end.

2. Java Script is dynamically typed language and Java is Statically typed language: i.e

In JavaScript, datatype of one variable can be changed:

var string = “hello world”;
string = 4;
document.write(string); //OUTPUT IS 4
document.write( ) will now print ‘4’ on the browser.

But in Java, the datatype of one variable cannot be changed and Java shows the error.

int number = 45;
number = “hello world”; //ERROR!!!!!!!

3. JavaScript is a scripting language while Java is a programming language:

Like other languages, Java also needs a compiler for building and running the programs while JavaScript scripts are read and manipulated by the browser.

4. Java and JavaScript are very different in their SYNTAX.

For example:

Hello World Program in JAVA:

```
public class hello

{

        public static void main(String[] args)

        {

                System.out.println("Hello World");

        }

}

Hello World Program in JavaScript:



<script>

        document.write("Hello World");

</script>
```

5. Both languages are Object Oriented but JavaScript is a Partial Object Oriented Language while Java is a fully Object Oriented Langauge. JavaScript can be used with or without using objects but Java cannot be used without using classes.

56. How to count the number of occurrences of a character in a string in java?

```
"import java.util.HashMap;



public class EachCharCountInString

{

    private static void characterCount(String inputString)

    {

        //Creating a HashMap containing char as a key and
occurrences as a value



        HashMap<Character, Integer> charCountMap = new
HashMap<Character, Integer>();



        //Converting given string to char array



        char[] strArray = inputString.toCharArray();
```

```
        //checking each char of strArray


        for (char c : strArray)
        {
            if(charCountMap.containsKey(c))
            {
                //If char 'c' is present in charCountMap,
incrementing it's count by 1


                charCountMap.put(c, charCountMap.get(c)+1);
            }
            else
            {
                //If char 'c' is not present in charCountMap,
                //putting 'c' into charCountMap with 1 as it's
value


                charCountMap.put(c, 1);
            }
        }


        //Printing inputString and charCountMap
```

```
        System.out.println(inputString+"" : ""+charCountMap);
    }

    public static void main(String[] args)
    {
       characterCount(""Java J2EE Java JSP J2EE"");

       characterCount(""All Is Well"");

       characterCount(""Done And Gone"");
    }
}"
```

57. how to read excel file in java?

```
"FileInputStream fis = new FileInputStream(new
File(""WriteSheet.xlsx""));

  XSSFWorkbook workbook = new XSSFWorkbook(fis);
  XSSFSheet spreadsheet = workbook.getSheetAt(0);
  Iterator < Row >  rowIterator = spreadsheet.iterator();
```

```
  while (rowIterator.hasNext()) {

     row = (XSSFRow) rowIterator.next();

     Iterator < Cell >  cellIterator = row.cellIterator();



     while ( cellIterator.hasNext()) {

        Cell cell = cellIterator.next();



        switch (cell.getCellType()) {

           case Cell.CELL_TYPE_NUMERIC:

              System.out.print(cell.getNumericCellValue() + ""
\t\t "");

              break;



           case Cell.CELL_TYPE_STRING:

              System.out.print(

              cell.getStringCellValue() + "" \t\t "");

              break;

        }

     }

     System.out.println();

  }
```

```
  fis.close();"
```

58. What is a method in java?

Functions are also known as methods in java.

59. How to read csv file in java?

```
"public static void readDataLineByLine(String file)

{



    try {



        // Create an object of filereader

        // class with CSV file as a parameter.

        FileReader filereader = new FileReader(file);



        // create csvReader object passing

        // file reader as a parameter

        CSVReader csvReader = new CSVReader(filereader);

        String[] nextRecord;



        // we are going to read data line by line

        while ((nextRecord = csvReader.readNext()) != null) {
```

```
            for (String cell : nextRecord) {
                System.out.print(cell + ""\t"");
            }
            System.out.println();
        }
    }
    catch (Exception e) {
        e.printStackTrace();
    }
} "
```

60. How to check java version in windows?

type java -version on command prompt.

61. What is public static void main in java?

This is the access modifier of the main method. It has to be public so that java runtime can execute this method. Remember that if you make any method non-public then it's not allowed to be executed by any program, there are some access restrictions applied. So it means that the main method has to be public. Let's see what happens if we define the main method as non-public.

When java runtime starts, there is no object of the class present. That's why the main method has to be static so that JVM can load the class into

memory and call the main method. If the main method won't be static, JVM would not be able to call it because there is no object of the class is present.

Java programming mandates that every method provide the return type. Java main method doesn't return anything, that's why it's return type is void. This has been done to keep things simple because once the main method is finished executing, java program terminates. So there is no point in returning anything, there is nothing that can be done for the returned object by JVM. If we try to return something from the main method, it will give compilation error as an unexpected return value.

62. Why we use interface in java?

It is used to achieve total abstraction. Since java does not support multiple inheritance in case of class, but by using interface it can achieve multiple inheritance . It is also used to achieve loose coupling. Interfaces are used to implement abstraction.

63. What is the purpose of serialization in java?

Object Serialization is a process used to convert the state of an object into a byte stream, which can be persisted into disk/file or sent over the network to any other running Java virtual machine. The reverse process of creating an object from the byte stream is called deserialization.

64. What is functional interface in java?

A functional interface in Java is an interface that contains only a single abstract (unimplemented) method. A functional interface can contain

default and static methods which do have an implementation, in addition to the single unimplemented method.

65. What is this keyword in java?

The this keyword refers to the current object in a method or constructor. The most common use of the this keyword is to eliminate the confusion between class attributes and parameters with the same name (because a class attribute is shadowed by a method or constructor parameter).

66. How was java initially named?

The language was at first called Oak after an oak tree that remained external Gosling's office. Later the task passed by the name Green and was at last renamed Java, from Java , a coffee brand, the coffee from Indonesia.

67. How to remove duplicate elements from array in java?

```
"public class Change

{

   public static int removeDuplicate(int[] arrNumbers, int num)

   {

      if(num == 0 || num == 1)

      {

         return num;
```

```
    }
    int[] arrTemporary = new int[num];
    int b = 0;
    for(int a = 0; a < num - 1; a++)
    {
       if(arrNumbers[a] != arrNumbers[a + 1])
       {
          arrTemporary[b++] = arrNumbers[a];
       }
    }
    arrTemporary[b++] = arrNumbers[num - 1];
    for(int a = 0; a < b; a++)
    {
       arrNumbers[a] = arrTemporary[a];
    }
    return b;
 }
 public static void main(String[] args)
 {
    int[] arrInput = {1, 2, 3, 3, 4, 5, 5, 6, 7, 8};
    int len = arrInput.length;
```

```
      len = removeDuplicate(arrInput, len);

      // printing elements

      for(int a = 0; a < len; a++)

      {

         System.out.print(arrInput[a] + "" "");

      }

   }

}

"
```

68. What is difference between throw and throws in java?

Throw is a keyword which is used to throw an exception explicitly in the program inside a function or inside a block of code. Throws is a keyword used in the method signature used to declare an exception which might get thrown by the function while executing the code.

69. What is classpath in java?

The CLASSPATH variable is one way to tell applications, including the JDK tools, where to look for user classes. (Classes that are part of the JRE, JDK platform, and extensions should be defined through other means, such as the bootstrap class path or the extensions directory.)

70. Why is Java Platform Independent?

At the time of compilation, the java compiler converts the source code into a JVM interpretable set of intermediate form, which is termed as byte code. This is unlike the compiled code generated by other compilers and is non-executable. The java virtual machine interpreter processes the non-executable code and executes it on any specific machine. Hence the platform dependency is removed.

71. What is Method overloading? Why is it used in Java?

Method overriding is a process in which methods inherited by child classes from parent classes are modified as per requirement by the child class. It's helpful in hierarchical system design where objects share common properties.

Example: Animal class has properties like fur colour, sound. Now dog and cat class inherit these properties and assign values specific to them to the properties.

```
class Animal {

    void sound(){

    }

}

 class Cat extends Animal{

     void sound(){

         System.out.println("Meow");

     }
```

```
}


class Dog extends Animal{
    void sound(){
        System.out.println("Bark");
    }
}


public class OverRide{
    public static void main(String args[]){
        Cat c=new Cat();
        c.sound();

        Dog d=new Dog();
        d.sound();
    }
}
```

72. What is Method overloading? Why is it used in Java?

If multiple functions in a class have the same name but different function definitions it is called method overloading.

It is used to make a java function serve multiple purposes making the code cleaner and API less complex.

Example:

```
println() prints any data type passed to it as a string.



public class Add_Overload {

    void add(int x, int y){

        System.out.println(x+y);

    }

    void add(double x, double y){

        System.out.println(x+y);

    }

    void add(double x, int y){

        System.out.println(x+y);

    }

    public static void main(String args[]){

        Add_Overload a= new Add_Overload();

        a.add(10,20);

        a.add(20.11,11.22);

        a.add(20.11,2);
```

```
    }
```

73. Why is Java Robust?

Java is termed as robust because of the following features:
– Lack of pointers: Java does not have pointers which make it secure
– Garbage Collection: Java automatically clears out unused objects from memory which are unused
– Java has strong memory management.
– Java supports dynamic linking.

74. Why is Java Secure?

Java does not allow pointers. Pointers give access to actual locations of variables in a system. Also, java programs are bytecode executables that can run only in a JVM. Hence java programs do not have access to the host systems on which they are executing, making it more secure. Java has its own memory management system, which adds to the security feature as well.

75. What is the difference between JDK, JRE, and JVM?

JDK is a software environment used for the development of Java programs. It's a collection of libraries that can be used to develop various applications. JRE (Java Runtime Environment) is a software environment that allows Java programs to run. All java applications run inside the JRE. JVM (java virtual machine) is an environment that is responsible for the conversion of java programs into bytecode executables. JDK and JRE are platform-dependent whereas JVM is platform-independent.

76. What are the features of Java?

Java is a pure Object Oriented Programming Language with the following features:

– High Performance

– Platform Independent

– Robust

– Multi-threaded

– Simple

– Secure

77. Does Java Support Pointers?

Pointers are not supported in java to make it more secure.

78. Why are Static variables used in Java?

Static methods and variables are used in java to maintain a single copy of the entity across all objects. When a variable is declared as static it is shared by all instances of the class. Changes made by an instance to the variable reflect across all instances.

```
public class static_variable {



    static int a;

    static int b;

    static_variable(){

        a=10;
```

```
    }
    int calc_b(){
        b=a+10;
        return b;
    }
void print_val(){
    System.out.println(this.b);
}
public static void main(String args[]){
    static_variable v=new static_variable();
    v.calc_b();
    v.print_val();
    static_variable v1=new static_variable();
    v1.print_val();
}
}
```

79. What are static methods, static variables, and static blocks?

Static methods are methods that can be called directly inside a class without the use of an object.
Static variables are variables that are shared between all instances of a

class.
Static blocks are code blocks that are loaded as the class is loaded in memory.

80. What’s the use of static methods?

Static methods are used when there is no requirement of instantiating a class. If a method is not going to change or overridden then it can be made static.

81. What’s the use of static variables?

Static variables are used for maintaining a common state of certain data which is modifiable and accessible by all instances of a class.

82. What are the interfaces?

An interface is a collection of constants, static methods, abstract methods, and default methods. Methods in an interface do not have a body.

83. How is Abstraction achieved in Java?

Abstraction is achieved in Java by the use of abstract class and abstract methods.

84. Why are strings immutable in Java?

Strings in java are frequently used for hashmap keys. Now if someone changes the value of the string it will cause severe discrepancies. Hence strings are made immutable.

85. What are wrapper classes in Java?

Wrapper classes are a functionality supported by java to accept primitive data types as inputs and then later convert those into string objects so that they can be compared to other objects.

86. Can interfaces in Java be inherited?

Yes, interfaces can be inherited in java. Hybrid inheritance and hierarchical inheritance are supported by java through inheritable interfaces.

87. Are static methods allowed in a Java interface?

Yes, static methods are allowed in java interfaces. They are treated as default methods so they need not be implemented.

88. How is garbage collection done in Java?

Java has an automatic built-in garbage collection mechanism in place. Apart from the built-in mechanism, manual initiation of garbage collection can also be done by using the gc() of system class.

89. Can there be two main methods in a class?

Yes, there can be two main methods. This also means that the main method is overloaded. But at the time of execution, JVM only calls the original main method and not the overloaded main method.

90. Can private variables be inherited?

Private variables have a class-specific scope of availability. They can only be accessed by the methods of the class in which they are present. Hence when the class is inherited, private variables are not inherited by the subclass.

91. Can the size of an array be increased after declaration?

The size of a java array cannot be increased after declaration. This is a limitation of Java arrays.

92. What is the size of the below array in memory? int a[]=new int[10];

Each int block takes a size of 4 bytes and there are 10 such blocks in the array. Hence, the size the array takes in memory is 40 bytes.

93. How many data types does java support?

Java supports 8 primitive data types, namely byte, short, int, long, float, double, char, boolean.

94. How to find out the ASCII value of a character in java?

int c=char('A') would give the ASCII value of A in java.

95. How to get a string as user input from the console?

We have to instantiate an input reader class first. There are quite a few options available, some of which are BufferedReader, InputStreamReader Scanner.
Then the relative functionality of the class can be used. One of the most prevalently used is nextLine() of Scanner class.

96. How to check the size of strings?

The size of strings in java can be checked by using the length() function.

97. How can we sort a list of elements in Java?

The built-in sorting utility sort() can be used to sort the elements. We can also write our custom functions but it's advisable to use the built-in function as its highly optimized.

98. If we sort a list of strings how would be the strings arranged? The strings would be arranged alphabetically in ascending order.

99. The difference between throw and throws in Java?

Throw is used to actually throw an instance of java.lang.Throwable class, which means you can throw both Error and Exception using throw keyword e.g.

```
throw new IllegalArgumentException("size must be multiple of
2")
```

On the other hand, throws is used as part of method declaration and signals which kind of exceptions are thrown by this method so that its caller can handle them. It's mandatory to declare any unhandled checked exception in throws clause in Java. Like the previous question, this is another frequently asked Java interview question from errors and exception topic but too easy to answer.

100. Can we make an array volatile in Java?

Yes, you can make an array volatile in Java but only the reference which is pointing to an array, not the whole array. What I mean, if one thread changes the reference variable to points to another array, that will provide a volatile guarantee, but if multiple threads are changing individual array elements they won't be having happens before guarantee provided by the volatile modifier.

101. Can I store a double value in a long variable without casting?

No, you cannot store a double value into a long variable without casting because the range of double is more than long and we need to type cast. It's not difficult to answer this question but many developer get it wrong due to confusion on which one is bigger between double and long in Java.

102. Which one will take more memory, an int or Integer?

An Integer object will take more memory as Integer is an object and it stores metadata overhead about the object but int is a primitive type, so it takes less space.

103. The difference between nested static class and top-level class?

A public top-level class must have the same name as the name of the source file, there is no such requirement for a nested static class. A nested class is always inside a top-level class and you need to use the name of the top-level class to refer nested static class e.g. HashMap.Entry is a nested static class, where HashMap is a top-level class and Entry is nested, static class.

104. What is the use of the final keyword?

The final keyword is used to declare the final state of an entity in java. The value of the entity cannot be modified at a later stage in the application. The entity can be a variable, class, object, etc.
It is used to prevent unnecessary modifications in a java application.

105. What's the difference between deep copy and shallow copy?

Shallow copy in java copies all values and attributes of an object to another object and both objects reference the same memory locations.

Deep copy is the creation of an object with the same values and attributes of the object being copied but both objects reference different memory locations.

106. What's the use of default constructor?

The default constructor is a constructor that gets called as soon as the object of a class is declared. The default constructor is un-parametrized. The generic use of default constructors is in the initialization of class variables.

```
class ABC{

    int i,j;

    ABC(){

        i=0;

        j=0;

    }

}
```

Here ABC() is a default constructor.

107. What is object cloning?

Object cloning is the process of creating an exact copy of an object of a class. The state of the newly created object is the same as the object used for cloning.
The clone() method is used to clone objects. The cloning done using the clone method is an example of a deep copy.

108. Why are static blocks used?

They serve the primary function of initializing the static variables. If multiple static blocks are there they are executed in the sequence in which they are written in a top-down manner.

109. What is the use of this keyword in java?

This keyword is used to reference an entity using the current object in java. It's a multi-purpose keyword which serves various functionalities

110. What's the difference between String and String Builder class in java?

Strings are immutable while string Builder class is mutable. The string builder class is also synchronized.

111. How to calculate the size of an object?

The size of an object can be calculated by summing the size of the variables of the class the object is instantiated from.
If a class has an integer, a double variable defined in it then the size of the object of the class is size(int)+size(double).
If there is an array, then the size of the object would be the length of array*size of data type of array.

112. What's the difference between == and .equals()?

"==" is an operator, whereas .equals() is a function.
"==" checks if the references share the same location, whereas .equals() checks if both object values are the same on evaluation.

Advance java interview questions.

1. What is serialization in java?

Object Serialization is a process used to convert the state of an object into a byte stream, which can be persisted into disk/file or sent over the network to any other running Java virtual machine. The reverse process of creating an object from the byte stream is called deserialization.

2. What is synchronization in java?

Synchronization is a process of handling resource accessibility by multiple thread requests. The main purpose of synchronization is to avoid thread interference. At times when more than one thread try to access a shared resource, we need to ensure that resource will be used by only one thread at a time. The process by which this is achieved is called synchronization. The synchronization keyword in java creates a block of code referred to as a critical section.

3. What is spring in java?

The Spring Framework is an application framework and inversion of control container for the Java platform. The framework's core features can be used

by any Java application, but there are extensions for building web applications on top of the Java EE (Enterprise Edition) platform.

4. How to create immutable class in java?

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

5. What is servlet in java?

A servlet is a Java programming language class that is used to extend the capabilities of servers that host applications accessed by means of a request-response programming model. Although servlets can respond to any type of request, they are commonly used to extend the applications hosted by web servers. For such applications, Java Servlet technology defines HTTP-specific servlet classes.

All servlets must implement the Servlet interface, which defines life-cycle methods. When implementing a generic service, you can use or extend the GenericServlet class provided with the Java Servlet API. The HttpServlet class provides methods, such as doGet and doPost, for handling HTTP-specific services.

6. What is xname class in java?

An Expanded Name, comprising of a (discretionary) namespace name and a nearby name. XName examples are changeless and might be shared.

7. Can static methods reference non-static variables?

Yes, static methods can reference non-static variables. It can be done by creating an object of the class the variable belongs to.

8. How do static blocks get executed if there are multiple static blocks?

Multiple static blocks are executed in the sequence in which they are written in a top-down manner. The top block gets executed first, then the subsequent blocks are executed.

9. Can we override static methods?

Static methods cannot be overridden because they are not dispatched to the object instance at run time. In their case, the compiler decides which method gets called.

10. Which class do all classes inherit from in java?

All classes in java inherit from the Object class which is the superclass of all classes.

11. What is classloader?

ClassLoader is a subsystem of JVM which is used to load class files. Whenever we run the java program, it is loaded first by the classloader. There are three built-in classloaders in Java.

- **Bootstrap ClassLoader**: This is the first classloader which is the superclass of Extension classloader. It loads the rt.jar file which contains all class files of Java Standard Edition like java.lang package classes, java.net package classes, java.util package classes, java.io package classes, java.sql package classes, etc.
- **Extension ClassLoader**: This is the child classloader of Bootstrap and parent classloader of System classloader. It loads the jar files located inside $JAVA_HOME/jre/lib/ext directory.
- **System/Application ClassLoader**: This is the child classloader of Extension classloader. It loads the class files from the classpath. By default, the classpath is set to the current directory. You can change the classpath using “-cp” or “-classpath” switch. It is thus also known as Application classloader.

12. The difference between Serializable and Externalizable in Java?

Serializable interface is used to make Java classes serializable so that they can be transferred over a network or their state can be saved on disk, but it leverages default serialization built-in JVM, which is expensive, fragile and not secure. Externalizable allows you to fully control the Serialization process, specify a custom binary format and add more security measure.

13. Can we use String in the switch case?

We can use String in switch case but it is just syntactic sugar. Internally string hash code is used for the switch. See the detailed answer for more explanation and discussion.

14. What are object serialization and deserialization?

The use of java.io.Serializable to convert an object into a sequence of bytes is known as object serialization. Deserialization is the process of recovering back the state of the object from the byte stream.

15. The difference between checked and unchecked exception in Java?

A checked exception is checked by the compiler at compile time. It's mandatory for a method to either handle the checked exception or declare them in their throws clause. These are the ones which are a subclass of Exception but doesn't descend from RuntimeException. The unchecked exception is the descendant of RuntimeException and not checked by the compiler at compile time. This question is now becoming less popular and you would only find this with interviews with small companies, both investment banks and startups are moved on from this question.

16. Is ++ operator is thread-safe in Java?

No, it's not a thread-safe operator because it involves multiple instructions like reading a value, incriminating it and storing it back into memory which can be overlapped between multiple threads.

17. Which class contains the clone method? Cloneable or Object?

java.lang.Cloneable is a marker interface and doesn't contain any method clone method is defined in the object class. It is also knowing that clone() is a native method means it's implemented in C or C++ or any other native language.

Java Coding Interview Questions

1. What is interface in java?

An interface in the Java programming language is an abstract type that is used to specify a behaviour that classes must implement. They are similar to protocols. Interfaces are declared using the interface keyword, and may only contain method signature and constant declarations.

2. How to declare array in java?

```
"To declare an array, define the variable type with square
brackets:



String[] cars;

We have now declared a variable that holds an array of strings.
To insert values to it, we can use an array literal - place the
values in a comma-separated list, inside curly braces:



String[] cars = {""Volvo"", ""BMW"", ""Ford"", ""Mazda""};
```

```
To create an array of integers, you could write:



int[] myNum = {10, 20, 30, 40};"
```

3. What is polymorphism in java?

Polymorphism is one of the OOPs features that allow us to perform a single action in different ways. For example, let's say we have a class Animal that has a method sound(). Since this is a generic class so we can't give it an implementation like Roar, Meow, Oink etc. We had to give a generic message.

```
public class Animal{

   ...

   public void sound(){

      System.out.println(""Animal is making a sound"");

   }

}

Now lets say we two subclasses of Animal class: Horse and Cat
that extends (see Inheritance) Animal class. We can provide the
implementation to the same method like this:



public class Horse extends Animal{

...

    @Override
```

```
    public void sound(){
        System.out.println(""Neigh"");
    }
}
```

and

```
public class Cat extends Animal{
...
    @Override
    public void sound(){
        System.out.println(""Meow"");
    }
}
```

As you can see that although we had the common action for all subclasses sound() but there were different ways to do the same action. This is a perfect example of polymorphism (feature that allows us to perform a single action in different ways). It would not make any sense to just call the generic sound() method as each Animal has a different sound. Thus we can say that the action this method performs is based on the type of object."

4. How to convert string to int in java?

```
"class Scratch{

    public static void main(String[] args){

        String str = ""50"";

        System.out.println( Integer.parseInt( str ));   //
Integer.parseInt()

    }

}"
```

5. How to convert int to string in java?

```
class Convert

{

  public static void main(String args[])

  {

    int a = 786;

    int b = -986;

    String str1 = Integer.toString(a);

    String str2 = Integer.toString(b);

    System.out.println(""String str1 = "" + str1);

    System.out.println(""String str2 = "" + str2);

  }
```

6. Why string is immutable in java?

The string is Immutable in Java because String objects are cached in String pool. Since cached String literals are shared between multiple clients there is always a risk, where one client's action would affect all another client. For example, if one client changes the value of String "ABC" to "abc", all other clients will also see that value as explained in the first example. Since caching of String objects was important from performance reason this risk was avoided by making String class Immutable. At the same time, String was made final so that no one can compromise invariant of String class e.g. Immutability, Caching, hashcode calculation etc by extending and overriding behaviours.

7. how to convert integer to string in java?

```
class ABC

{

  public static void main(String args[])

  {

    int a = 789;

    int b = 123;

    String str1 = Integer.toString(a);

    String str2 = Integer.toString(b);

    System.out.println(""String str1 = "" + str1);

    System.out.println(""String str2 = "" + str2);
```

```
  }
```

8. How to compile java program?

Open a command prompt window and go to the directory where you saved the java program (MyFirstJavaProgram. java). …
Type ‘javac MyFirstJavaProgram. java’ and press enter to compile your code

9. How to convert char to string in java?

```
public class CharToStringExample2{

public static void main(String args[]){

char c='M';

String s=Character.toString(c);

System.out.println(""String is: ""+s);

}}
```

10. What is wrapper class in java?

Wrapper classes are used for converting primitive data types into objects, like int to Integer etc.

11. How to iterate map in java?

```
import java.util.Map;

import java.util.HashMap;
```

```
class IterationDemo

{

    public static void main(String[] arg)

    {

        Map<String,String> m = new HashMap<String,String>();



        // enter name/url pair

        m.put(""a"", 1);

        m.put(""b"", 2);

        m.put(""c"", 3);

        m.put(""d"", 4);



        // using for-each loop for iteration over
Map.entrySet()

        for (Map.Entry<String,String> entry : m.entrySet())

            System.out.println(""Key = "" + entry.getKey() +

                               "", Value = "" +
entry.getValue());

    }

}
```

12. How to convert char to int in java?

```
public class JavaExample{

   public static void main(String args[]){

         char ch = '10';

         int num = Integer.parseInt(String.valueOf(ch));



         System.out.println(num);

   }

}
```

13. What is an interface in java?

An interface in Java is similar to a class, but the body of an interface can include only abstract methods and final fields (constants). A class implements an interface by providing code for each method declared by the interface.

14. How to split string in java?

```
String string = ""004-034556"";

String[] parts = string.split(""-"");

String part1 = parts[0]; // 004

String part2 = parts[1]; // 034556
```

14. How to read a file in java?

```
import java.io.*;

public class Read

{

  public static void main(String[] args)throws Exception

  {

  File file = new File(""C:\\Users\\LBL\\Desktop\\test.txt"");



  BufferedReader br = new BufferedReader(new FileReader(file));



  String st;

  while ((st = br.readLine()) != null)

    System.out.println(st);

  }

}
```

15. How to use scanner in java?

```
import java.util.Scanner;



class classname{

  public methodname(){
```

```
    //Scanner declaration
    Scanner s_name = new Scanner(System.in);
    //Use Scanner to take input
    int val = s_name.nextInt();
  }
}
```

16. How to reverse a number in java?

```
class Reverse
{
   public static void main(String args[])
   {
      int num=564;
      int reverse =0;
      while( num != 0 )
      {
          reverse = reverse * 10;
          reverse = reverse + num%10;
          num = num/10;
      }
```

```
      System.out.println(""Reverse  is: ""+reverse);

   }

}
```

17. What is instance in java?

Instance variable in Java is used by Objects to store their states. Variables that are defined without the STATIC keyword and are Outside any method declaration are Object-specific and are known as instance variables. They are called so because their values are instance specific and are not shared among instances.

18. How to convert char array to string in java?

```
class CharArrayToString

{

   public static void main(String args[])

   {

      // Method 1: Using String object

      char[] ch = {'g', 'o', 'o', 'd', ' ', 'm', 'o', 'r', 'n',
'i', 'n', 'g'};

      String str = new String(ch);

      System.out.println(str);



      // Method 2: Using valueOf method
```

```
        String str2 = String.valueOf(ch);

        System.out.println(str2);

    }

}
```

19. What is maven in java?

Maven is a powerful project management tool that is based on POM (project object model). It is used for project build, dependency and documentation.

It simplifies the build process like ANT. But it is too much advanced than ANT.

20. What is an array in java?

An array is a container object that holds a fixed number of values of a single type. The length of an array is established when the array is created. After creation, its length is fixed. You have seen an example of arrays already, in the main method of the “Hello World!” application.

21. What is applet in java?

An applet is a special kind of Java program that runs in a Java-enabled browser. This is the first Java program that can run over the network using the browser. An applet is typically embedded inside a web page and runs in the browser.

In other words, we can say that Applets are small Java applications that

can be accessed on an Internet server, transported over the Internet, and can be automatically installed and run as apart of a web document.

22. What is method overriding in java?

```
class Human{

   //Overridden method

   public void eat()

   {

      System.out.println(""Human is eating"");

   }

}

class Boy extends Human{

   //Overriding method

   public void eat(){

      System.out.println(""Boy is eating"");

   }

   public static void main( String args[]) {

      Boy obj = new Boy();

      //This will call the child class version of eat()

      obj.eat();

   }
```

```
}
```

22. how to check java is installed or not?

- Click Start
- Select Control Panel
- Select Programs
- Click Programs and Features
- The installed Java version(s) are listed

23. How to return an array in java?

```
import java.util.*;

public class Main

{

public static String[] return_Array() {

       //define string array

       String[] ret_Array = {""Java"", ""C++"", ""Python"",
""Ruby"", ""C""};

      //return string array

      return ret_Array;

   }



public static void main(String args[]) {

      //call method return_array that returns array

     String[] str_Array = return_Array();
```

```
     System.out.println(""Array returned from method:"" +
Arrays.toString(str_Array));



    }

}
```

24. How to generate random number in java?

```
public static double getRandomNumber(){

    double x = Math.random();

    return x;

}
```

25. What is generics in java?

Generics enable types (classes and interfaces) to be parameters when defining classes, interfaces and methods. Much like the more familiar formal parameters used in method declarations, type parameters provide a way for you to re-use the same code with different inputs. The difference is that the inputs to formal parameters are values, while the inputs to type parameters are types.

26. What is a constructor in java?

A constructor in Java is a special method that is used to initialize objects. The constructor is called when an object of a class is created.

27. How to find the length of an array in java?

```
class ArrayLengthFinder {

   public static void main(String[] arr) {

      // declare an array

      int[] array = new int[10];

      array[0] = 12;

      array[1] = -4;

      array[2] = 1;

      // get the length of array

      int length = array.length;

      System.out.println(""Length of array is: "" + length);

   }

}
```

28. What is overriding in java?

Method overriding is a process of overriding base class method by derived class method with a more specific definition.

Method overriding performs only if two classes have an is-a relationship. It means class must have inheritance. In other words, It is performed between two classes using inheritance relation.

In overriding, method of both classes must have the same name and an

equal number of parameters.

Method overriding is also referred to as runtime polymorphism because the calling method is decided by JVM during runtime.

The key benefit of overriding is the ability to define a method that's specific to a particular subclass type.

29. How to create a file in java?

Open a text file, save it with extension .java.

30. What is an instance variable in java?

Instance variable in Java is used by Objects to store their states. Variables that are defined without the STATIC keyword and are Outside any method declaration are Object-specific and are known as instance variables. They are called so because their values are instance specific and are not shared among instances.

31. How to iterate hashmap in java?

```
import java.util.Map;

import java.util.HashMap;



class IterationDemo

{

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

```
    {

        Map<String,String> g = new HashMap<String,String>();



        // enter name/url pair

        g.put(""1"":""One"");

         g.put(""2"":""Two"");

        // using for-each loop for iteration over
Map.entrySet()

        for (Map.Entry<String,String> entry : g.entrySet())

            System.out.println(""Key = "" + entry.getKey() +

                               "", Value = "" +
entry.getValue());

    }

}
```

32. How to split a string in java?

```
public class JavaExample{

   public static void main(String args[]){

      String s = "" ,ab;gh,bc;pq#kk$bb"";

      String[] str = s.split(""[,;#$]"");



      //Total how many substrings? The array length
```

```
      System.out.println(""Number of substrings:
""+str.length);



      for (int i=0; i < str.length; i++) {

            System.out.println(""Str[""+i+""]:""+str[i]);

      }

   }

}
```

33. How to sort array in java?

```
public class InsertSort {

  public static void main (String [] args) {

   int [] array = {10,20,30,60,70,80,2,3,1};

   int temp;

   for (int i = 1; i < array.length; i++) {

    for (int j = i; j > 0; j--) {

     if (array[j] < array [j - 1]) {

      temp = array[j];

      array[j] = array[j - 1];

      array[j - 1] = temp;

     }

    }
```

```
   }
   for (int i = 0; i < array.length; i++) {
     System.out.println(array[i]);
   }
  }
}
```

34. Why main method is static in java?

Java main() method is always static, so that compiler can call it without the creation of an object or before the creation of an object of the class. In any Java program, the main() method is the starting point from where compiler starts program execution. So, the compiler needs to call the main() method.

35. How to reverse a string in java word by word?

```
import java.util.*;
class ReverseString
{
  public static void main(String args[])
  {
    String original, reverse = """";
    Scanner in = new Scanner(System.in);
```

```
    System.out.println(""Enter a string to reverse"");

    original = in.nextLine();



    int length = original.length();



    for (int i = length - 1 ; i >= 0 ; i--)

      reverse = reverse + original.charAt(i);



    System.out.println(""Reverse of the string: "" + reverse);

  }

}
```

36. How to convert string to date in java?

```
String string = ""January 2, 2010"";

DateTimeFormatter formatter =
DateTimeFormatter.ofPattern(""MMMM d, yyyy"", Locale.ENGLISH);

LocalDate date = LocalDate.parse(string, formatter);

System.out.println(date); // 2010-01-02
```

37. How to read a string in java?

```
Scanner sc= new Scanner(System.in); //System.in is a standard
input stream.

System.out.print(""Enter a string: "");
```

```
String str= sc.nextLine(); //reads string.
```

38. How to convert string to integer in java?

```
String string1 = ""100"";

String string2 = ""50"";

String string3 = ""20"";



int number1 = Integer.decode(string1);

int number2 = Integer.decode(string2);

int number3 = Integer.decode(string3);



System.out.println(""Parsing String \""" + string1 + ""\"": ""
+ number2);

System.out.println(""Parsing String \""" + string2 + ""\"": ""
+ number2);

System.out.println(""Parsing String \""" + string3 + ""\"": ""
+ number3);
```

39. How to sort arraylist in java?

```
import java.util.*;

public class ArrayListOfInteger  {



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

```
        ArrayList<Integer> arraylist = new
ArrayList<Integer>();

        arraylist.add(11);

        arraylist.add(2);

        arraylist.add(7);

        arraylist.add(3);

        /* ArrayList before the sorting*/

        System.out.println(""Before Sorting:"");

        for(int counter: arraylist){

                System.out.println(counter);

           }



        /* Sorting of arraylist using Collections.sort*/

        Collections.sort(arraylist);



        /* ArrayList after sorting*/

        System.out.println(""After Sorting:"");

        for(int counter: arraylist){

                System.out.println(counter);

           }

      }
```

```
}
```

40. How to find the length of a string in java?

To calculate the length of a string in Java, you can use an inbuilt length() method of the Java string class.

In Java, strings are objects created using the string class and the length() method is a public member method of this class. So, any variable of type string can access this method using the . (dot) operator.

The length() method counts the total number of characters in a String.

41. What is data type in java?

Data type in java specifies the type of value a variable in java can store.

42. What is hashmap in java?

HashMap is a Map-based collection class that is used for storing Key & value pairs, it is denoted as HashMap<Key, Value> or HashMap<K, V>. This class makes no guarantees as to the order of the map. It is similar to the Hashtable class except that it is unsynchronized and permits nulls(null values and null key).

43. What is stream in java?

A Stream in Java can be defined as a sequence of elements from a source. Streams supports aggregate operations on the elements. The source of

elements here refers to a Collection or Array that provides data to the Stream.

Stream keeps the ordering of the elements the same as the ordering in the source. The aggregate operations are operations that allow us to express common manipulations on stream elements quickly and clearly.

44. How to convert double to string in java?

```
public class D2S{

public static void main(String args[]){

double d=1.2222222;

String s=Double. toString(d);

System. out. println(s);

}}
```

45. How to declare an array in java?

```
int arr[]=new int[10];
```

46. How to replace a character in a string in java?

String replace(char oldChar, char newChar): It replaces all the occurrences of a oldChar character with newChar character. For e.g. “pog pance”.replace(‘p’, ‘d’) would return dog dance.

47. What is lambda expression in java?

A lambda expression (lambda) describes a block of code (an anonymous function) that can be passed to constructors or methods for subsequent execution. The constructor or method receives the lambda as an argument. Consider the following example:

System.out.println(“Hello”)
This example identifies a lambda for outputting a message to the standard output stream. From left to right, () identifies the lambda’s formal parameter list (there are no parameters in the example), -> indicates that the expression is a lambda, and System.out.println(“Hello”) is the code to be executed.

48. What is microservices java?

Microservices are a form of service-oriented architecture style (one of the most important skills for Java developers) wherein applications are built as a collection of different smaller services rather than one whole app.

49. What is jsp in java?

A JSP page is a text document that contains two types of text: static data, which can be expressed in any text-based format (such as HTML, SVG, WML, and XML), and JSP elements, which construct dynamic content.

The recommended file extension for the source file of a JSP page is .jsp. The page can be composed of a top file that includes other files that contain either a complete JSP page or a fragment of a JSP page. The recommended extension for the source file of a fragment of a JSP page is .jspf.

The JSP elements in a JSP page can be expressed in two syntaxes, standard and XML, though any given file can use only one syntax. A JSP page in XML syntax is an XML document and can be manipulated by tools and APIs for XML documents.

50. What is the use of constructor in java?

A constructor is a block of code that initializes the newly created object. A constructor resembles an instance method in java but it's not a method as it doesn't have a return type. In short constructor and method are different(More on this at the end of this guide). People often refer constructor as special type of method in Java.

A constructor has same name as the class and looks like this in java code.

51. How to convert list to array in java?

The best and easiest way to convert a List into an Array in Java is to use the .toArray() method.

Likewise, we can convert back a List to Array using the Arrays.asList() method.

The examples below show how to convert List of String and List of Integers to their Array equivalents.

```
Convert List to Array of String

import java.util.ArrayList;
```

```
import java.util.List;


public class ConvertArrayListToArray {

    public static void main(String[] args) {

        List<String> itemList = new ArrayList<String>();

        itemList.add(""item1"");

        itemList.add(""item2"");

        itemList.add(""item3"");


        String[] itemsArray = new String[itemList.size()];

        itemsArray = itemList.toArray(itemsArray);


        for(String s : itemsArray)

            System.out.println(s);

    }

}"
```

52. How many ways to create object in java?

There are five different ways to create an object in Java:

- Java new Operator
- Java Class. newInstance() method
- Java newInstance() method of constructor

- Java Object. clone() method
- Java Object Serialization and Deserialization

53. Why java is becoming functional (java 8)?

Java 8 adds functional programming through what are called lambda expressions, which is a simple way of describing a function as some operation on an arbitrary set of supplied variables.

54. Which inheritance is not supported in java?

Multiple inheritance is not supported in Java.

55. How to convert double to int in java?

```
double d=1.2

int i=int(d)
```

56. How to get ASCII value of char in java?

```
char character = 'a';

int ascii = (int) character;
```

In your case, you need to get the specific Character from the String first and then cast it.

```
char character = name.charAt(0); // This gives the character
'a'

int ascii = (int) character; // ascii is now 97.
```

57. How to declare a string array in java?

```
String[] array = new String[] {"a", "b"};
```

58. What is marker interface in java?

An empty interface in Java is known as a marker interface i.e. it does not contain any methods or fields by implementing these interfaces a class will exhibit a special behaviour with respect to the interface implemented. If you look carefully on marker interface in Java e.g. Serializable, Cloneable and Remote it looks they are used to indicate something to compiler or JVM. So if JVM sees a Class is Serializable it done some special operation on it, similar way if JVM sees one Class is implement Clonnable it performs some operation to support cloning. Same is true for RMI and Remote interface. In simplest Marker interface indicate, signal or a command to Compiler or JVM.

–> Practically we can create an interface like a marker interface with no method declaration in it but it is not a marker interface at all since it is not instructing something to JVM that provides some special behaviour to the class when our program is going to execute.

For example Serializable, Cloneable etc..are marker interfaces

When my program gets executed, JVM provides some special powers to my class which has implemented the Marker Interfaces.

59. How to take multiple string input in java using a scanner?

```
class MyClass {

    public static void main(String[ ] args) {

        Scanner a = new Scanner(System.in);

        //Scanner b = new Scanner(System.in);

        System.out.println (a.nextLine());

        System.out.println(a.nextLine());

    }

}



Then type this way:

a

b
```

60. How to concatenate two strings in java?

s3=s1+s2 where s1 and s2 are java strings.

61. How to convert string to char array in java?

```
public class StringToCharArrayExample{

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

```
String s1=""hello"";

char[] ch=s1.toCharArray();

for(int i=0;i<ch.length;i++){

System.out.print(ch[i]);

}

}}
```

62. What is type casting in java?

The process of converting the value of one data type (int, float, double, etc.) to another data type is known as typecasting.

63. How to sort a string in java?

```
import java.util.Arrays;



public class Test

{

    public static void main(String[] args)

    {

        String original = ""edcba"";

        char[] chars = original.toCharArray();

        Arrays.sort(chars);

        String sorted = new String(chars);
```

```
        System.out.println(sorted);

    }

}
```

64. How to input string in java?

```
import java.util.*;

class Inp

{

public static void main(String[] args)

{

Scanner sc= new Scanner(System.in); //System.in is a standard
input stream

System.out.print(""Enter a string: "");

String str= sc.nextLine();              //reads string

System.out.print(""You have entered: ""+str);

}

}
```

65. How to import scanner in java?

```
import java.utils.Scanner

Scanner sc=new Scanner();
```

66. How to remove special characters from a string in java?

```
class New

{

public static void main(String args[])

{

String str= ""This#string%contains^special*characters&."";

str = str.replaceAll(""[^a-zA-Z0-9]"", "" "");

System.out.println(str);

}

}
```

67. How to find string length in java?

To figure the length of a string in Java, you can utilize an inbuilt length() technique for the Java string class.

In Java, strings are objects made utilizing the string class and the length() strategy is an open part technique for this class. Along these lines, any factor of type string can get to this strategy utilizing the . (dot)

administrator.

The length() technique tallies the all outnumber of characters in a String.

68. How to add elements in array in java?

Convert array to arraylist. Then elements can be added.

69. What is exception handling in java?

Exception Handling in Java is a way to keep the program running even if some fault has occurred. An exception is an error event that can happen during the execution of a program and disrupts its normal flow. Java provides a robust and object-oriented way to handle exception scenarios, known as Java Exception Handling.

70. How to scan string in java?

```
import java.util.*;

public class ScannerExample {

public static void main(String args[]){

Scanner in = new Scanner(System.in);

System.out.print(""Enter your name: "");

String name = in.nextLine();

System.out.println(""Name is: "" + name);

in.close();
```

71. When to use comparable and comparator in java with example?

In case one wants a different sorting order then he can implement comparator and define its own way of comparing two instances. If sorting of objects needs to be based on natural order then use Comparable whereas if your sorting needs to be done on attributes of different objects, then use Comparator in Java.

72. How to create jar file in java?

The basic format of the command for creating a JAR file is:

jar cf jar-file input-file(s)
The options and arguments used in this command are:

- The c option indicates that you want to create a JAR file
- The f option indicates that you want the output to go to a file rather than to stdout

jar-file is the name that you want the resulting JAR file to have. You can use any filename for a JAR file. By convention, JAR filenames are given a .jar extension, though this is not required.
The input-file(s) argument is a space-separated list of one or more files that you want to include in your JAR file. The input-file(s) argument can contain the wildcard * symbol. If any of the “input-files” are directories, the contents of those directories are added to the JAR archive recursively.
The c and f options can appear in either order, but there must not be any space between them.

73. How to call a method in java?

To call a method in Java, write the method's name followed by two parentheses () and a semicolon; The process of method calling is simple. When a program invokes a method, the program control gets transferred to the called method.

74. What is the difference between next () and nextline () in java?

next() can read the input only till space. It can't read two words separated by space. Also, next() places the cursor in the same line after reading the input. nextLine() reads input including space between the words (that is, it reads till the end of line \n).

75. what is mvc in java?

MVC Pattern represents the Model-View-Controller Pattern. This example is utilized to isolate the application's interests. Model – Model speaks to an item or JAVA POJO conveying information. It can likewise have a rationale to refresh regulator if its information changes.

76. How to iterate a map in java?

```
for (Map.Entry<Integer, String> entry : hm.entrySet()) {

    Integer key = entry.getKey();

    String value = entry.getValue();
```

```
}
```

77. What is the diamond problem in java?

The “diamond problem” is an uncertainty that can emerge as a result of permitting various legacy. It is a significant issue for dialects (like C++) that take into account numerous legacy of the state. In Java, nonetheless, numerous legacy doesn’t take into account classes, just for interfaces, and these don’t contain state.

78. How to swap two strings in java?

```
String a = ""one"";

String b = ""two"";



a = a + b;

b = a.substring(0, (a.length() - b.length()));

a = a.substring(b.length());



System.out.println(""a = "" + a);

System.out.println(""b = "" + b);
```

79. How to convert string to date in java in yyyy-mm-dd format?

```
String start_dt = ""2011-01-01"";

DateFormat formatter = new SimpleDateFormat(""yyyy-MM-DD"");
```

```
Date date = (Date)formatter.parse(start_dt);

SimpleDateFormat newFormat = new SimpleDateFormat(""MM-dd-
yyyy"");

String finalString = newFormat.format(date);
```

80. What is getname in java with example?

```
import java.io.*;



public class solution {

    public static void main(String args[])

    {



        // try-catch block to handle exceptions

        try {



            // Create a file object

            File f = new File(""new.txt"");



            // Get the Name of the given file f

            String Name = f.getName();



            // Display the file Name of the file object
```

```
            System.out.println(""File Name : "" + Name);
        }
        catch (Exception e) {
            System.err.println(e.getMessage());
        }
    }
}
```

getName returns the name of the file.

81. What is bufferreader in java?

The Java.io.BufferedReader class peruses text from a character-input stream, buffering characters to accommodate the proficient perusing of characters, clusters, and lines. Following are the significant focuses on BufferedReader – The cradle size might be determined, or the default size might be utilized.

82. How to create a package in java?

To make a bundle, you pick a name for the bundle (naming shows are talked about in the following area) and put a bundle articulation with that name at the head of each source record that contains the sorts (classes, interfaces, lists, and explanation types) that you need to remember for the bundle.

83. What is aggregation in java?

The case of Aggregation is Student in School class when School shut, Student despite everything exists and afterwards can join another School or something like that. In UML documentation, a structure is signified by a filled precious stone, while conglomeration is indicated by an unfilled jewel, which shows their undeniable distinction regarding the quality of the relationship.

84. How to use switch case in java?

```
int amount = 9;



switch(amount) {

    case     0 : System.out.println(""amount is  0""); break;

    case     5 : System.out.println(""amount is  5""); break;

    case    10 : System.out.println(""amount is 10""); break;

    default    : System.out.println(""amount is something
else"");

}
```

85. What is recursion in java?

Recursion is simply the strategy of settling on a capacity decision itself. This method gives an approach to separate entangled issues into straightforward issues which are simpler to settle.

86. How to print array in java?

System.out.println(Arrays.toString(a));

87. What is autoboxing and unboxing in java?

Autoboxing is the automatic conversion that the Java compiler makes between the primitive types and their corresponding object wrapper classes. For example, converting an int to an Integer, a double to a Double, and so on. If the conversion goes the other way, this is called unboxing.

88. A java constructor returns a value, but what?

Java constructor does not return any values.

89. What is method overloading in java?

Method Overloading is a feature that allows a class to have more than one method having the same name if their argument lists are different. It is similar to constructor overloading in Java, that allows a class to have more than one constructor having different argument lists.

90. How to create an array of objects in java?

One way to initialize the array of objects is by using the constructors. When you create actual objects, you can assign initial values to each of the objects by passing values to the constructor. You can also have a separate member method in a class that will assign data to the objects.

91. When to use abstract class and interface in java?

Java provides four types of access specifiers that we can use with classes and other entities.

These are:

#1) Default: Whenever a specific access level is not specified, then it is assumed to be 'default'. The scope of the default level is within the package.

#2) Public: This is the most common access level and whenever the public access specifier is used with an entity, that particular entity is accessible throughout from within or outside the class, within or outside the package, etc.

#3) Protected: The protected access level has a scope that is within the package. A protected entity is also accessible outside the package through inherited class or child class.

#4) Private: When an entity is private, then this entity cannot be accessed outside the class. A private entity can only be accessible from within the class.

92. How to create singleton class in java?

Singleton class means you can create only one object for the given class. You can create a singleton class by making its constructor as private so that you can restrict the creation of the object. Provide a static method to

get an instance of the object, wherein you can handle the object creation inside the class only. In this example, we are creating an object by using a static block.

```
public class MySingleton {

    private static MySingleton myObj;

    static{
        myObj = new MySingleton();
    }

    private MySingleton(){

    }

    public static MySingleton getInstance(){
        return myObj;
    }

    public void testMe(){
```

```
        System.out.println(""Hey.... it is working!!!"");
    }

    public static void main(String a[]){
        MySingleton ms = getInstance();
        ms.testMe();
    }
}
```

93. What is a static method in java?

Java Programming Java8Object Oriented Programming. The static keyword is used to create methods that will exist independently of any instances created for the class. Static methods do not use any instance variables of any object of the class they are defined in.

94. Explain the exception handling mechanism of Java?

Exception class inherits from the Throwable class in java. Java has a try-catch mechanism for handling exceptions without them being generated as errors.

```
public class Exception_Handling {
    String gender;
    Exception_Handling(String s){
```

```
        gender=s;
    }
     void Check_Gender(String s) throws GenderException{
        if (gender!="Male" || gender!="Female")
            throw new GenderException("Gender Invalid");
        else
        System.out.println("Gender valid");
        }
    public static void main(String args[]){
        Exception_Handling n=new Exception_Handling("None");
        try{
            n.Check_Gender("Female");
        }catch (Exception e){
            System.out.println("Exception : "+e);
        }
    }
    }
class GenderException extends Exception{
    GenderException(String s){
        super(s);
    }
```

```
}
```

95. When do we use the Array list?

Whenever there is a need for random access of elements in java we use ArrayList. Get and set methods provide really fast access to the elements using the array list.

96. What is the use of generics in Java?

Generics allow classes and interfaces to be a type for the definition of new classes in java which enables stronger type checking. It also nullifies the probability of type mismatch of data while insertion.

97. What is an iterator?

An iterator is a collection framework functionality which enables sequential access of elements. The access can be done in one direction only. Java supports two types of iterators:
1. Enumeration Iterator
2. List Iterator

98. What is a hashmap?

Hashmap is a collection framework functionality which is used for storing data into key-value pairs. To access data we need the key. A hashmap uses linked lists internally for supporting the storage functionality.

99. What is a stack?

A stack is a data structure that supports LAST IN FIRST OUT methodology. The element pushed last is at the top of the stack. A stack supports the following functionality:

- Push-operation to push an element into the stack
- Pop-operation to push an element out of the stack
- Peek-An option to check the top element

100. What is a treemap?

Treemap is a navigable map interpretation in java which is built around the concepts of red and black trees. The keys of a treemap are sorted in ascending order by their keys.

101. What is a vector?

A vector is an ArrayList like data structure in java whose size increases as per the demands. Moreover, it also supports some legacy functions not supported by collections.
You should also know that a vector is more suitable to work with threads, unlike collection objects.

102. What is the difference between ArrayList and vector?

An ArrayList is not suitable for working in a thread based environment. A vector is built for thread-based executions. ArrayList does not support legacy functions whereas a vector has support for legacy functions.

103. Write a program to calculate the factorial of a number in java.

```
import java.util.Scanner;



public class star {

     public static void main(String[] args) {

          Scanner sc=new Scanner(System.in);

          int fact=1;

          int n=sc.nextInt();



          for (int i=1;i<=n;i++)

          fact=fact*i;



          System.out.println(fact);







        }



}
```

104. Write a program to check if a number is prime.

```
import java.util.Scanner;

public class star {

     public static void main(String[] args) {

          Scanner sc=new Scanner(System.in);

          int n=sc.nextInt();

          int count=0;

          for (int i=1;i<=n;i++)

          {

              if (n%i==0)

              count++;

          }

          if (count==2)

          System.out.println("Prime");

          else

          System.out.println("Not Prime");

         }

}
```

105. Write a program to convert decimal numbers to binary.

```
import java.util.Scanner;



class star

{

public static void main(String arg[])

{

    Scanner sc=new Scanner(System.in);

    System.out.println("Enter a decimal number");

    int n=sc.nextInt();

    int  bin[]=new int[100];

    int i = 0;

    while(n > 0)

    {

    bin[i++] = n%2;

       n = n/2;

    }

   System.out.print("Binary number is : ");

    for(int j = i-1;j >= 0;j--)
```

```
   {
       System.out.print(bin[j]);
   }
}
}
```

106. Write a program to convert decimal numbers to octal.

```
import java.util.Scanner;
class star
{
  public static void main(String args[])
  {
    Scanner sc = new Scanner( System.in );
    System.out.print("Enter a decimal number : ");
    int num =sc.nextInt();
    String octal = Integer.toOctalString(num);
    System.out.println("Decimal to octal: " + octal);
  }
}
```

107. Which utility function can be used to extract characters at a specific location in a string?

The charAt() utility function can be used to achieve the above-written functionality.

108. How to get the length of a string in java?

Length of string in java can be found using the .length() utility.

109. Which of the following syntax for defining an array is correct?

– Int []=new int[];
– int a[]=new int[];
– int a[] =new int [32]

int a[]=new int[32] is the correct method.

110. What will this return 3*0.1 == 0.3? true or false?

This is one of the really tricky questions and can be answered only if your concepts are very clear. The short answer is false because some floating-point numbers can not be represented exactly.

111. Write a program to do bubble sort on an array in java.

```
import java.util.Scanner;

class star
```

```
{
  public static void main(String args[])
  {
    int arr[] =new int [10];
    Scanner sc = new Scanner( System.in );
    System.out.println("Enter size of array");
    int n=sc.nextInt();
    System.out.print("Enter an arry : ");
    for (int i=0;i<n;i++)
        arr[i]=sc.nextInt();
    for (int i=0;i<n;i++)
    {
        for (int j=0;j<n-i-1;j++)
        {
            if (arr[j]>arr[j+1])
            {
                int t=arr[j];
                arr[j]=arr[j+1];
                arr[j+1]=t;
            }
        }
```

```
    }
 for (int i=0;i<n;i++)
    {
        System.out.println(arr[i]);
    }
  }
}
```

112. Write a program to generate the following output in java?

*

**

****

*****

******

```
public class star {
     public static void main(String[] args) {
         int i;
         int count=1;
        for (i=1;i<=5;i++){
            for (int j=1;j<=i;j++)
                System.out.print("*");
```

```
            System.out.println(" ");


        }


}
}
```

113. Write a program to generate the following output.

****

***

**

*

```
public class star {

     public static void main(String[] args) {

         int i;

         int count=1;

        for (i=5;i>=1;i--){

            for (int j=1;j<=i;j++)

                System.out.print("*");

            System.out.println(" ");


        }
```

```
}

}
```

114. Write a program in java to remove all vowels from a string.

```
import java.util.Scanner;



public class star {

     public static void main(String[] args) {

          Scanner sc=new Scanner(System.in);

          String n=sc.nextLine();

          String n1=n.replaceAll("[AEIOUaeiou]", "");

          System.out.println(n1);



          }

         }
```

115. Write a program in java to check for palindromes.

```
String str, rev = "";

       Scanner sc = new Scanner(System.in);
```

```
System.out.println("Enter a string:");

str = sc.nextLine();


int length = str.length();


for ( int i = length - 1; i >= 0; i-- )

   rev = rev + str.charAt(i);


if (str.equals(rev))

   System.out.println(str+" is a palindrome");

else

   System.out.println(str+" is not a palindrome");
```

116. What is the underlying mechanism in java's built-in sort?

Java's built-in sort function utilizes the two pivot quicksort mechanism. Quicksort works best in most real-life scenarios and has no extra space requirements.

117. Which utility function is used to check the presence of elements in an ArrayList?

hasNext() is used for the presence of the next element in an ArrayList.

118. How to remove an element from an array?

To remove an element from an array we have to delete the element first and then the array elements lying to the right of the element are shifted left by one place.

119. Difference between a = a + b and a += b ?

The += operator implicitly cast the result of addition into the type of the variable used to hold the result. When you add two integral variables e.g. variable of type byte, short, or int then they are first promoted to int and them addition happens. If the result of the addition is more than the maximum value of a then a + b will give a compile-time error but a += b will be ok as shown below

```
byte a = 127;
byte b = 127;
b = a + b; // error : cannot convert from int to byte
b += a; // ok
```

Java OOPS Interview Questions

1. What is class in Java?

In the real world, you often have many objects of the same kind. For example, your bicycle is just one of many bicycles in the world. Using object-oriented terminology, we say that your bicycle object is an instance (in the glossary) of the class of objects known as bicycles. Bicycles have some state (current gear, current cadence, two wheels) and behaviour (change gears, brake) in common. However, each bicycle's state is

independent of and can be different from that of other bicycles.
When building bicycles, manufacturers take advantage of the fact that bicycles share characteristics, building many bicycles from the same blueprint. It would be very inefficient to produce a new blueprint for every individual bicycle manufactured.

In object-oriented software, it's also possible to have many objects of the same kind that share characteristics: rectangles, employee records, video clips, and so on. Like the bicycle manufacturers, you can take advantage of the fact that objects of the same kind are similar and you can create a blueprint for those objects. A software blueprint for objects is called a class (in the glossary).

2. What is a constructor in java?

```
"A constructor in Java is a special method that is used to
initialize objects. The constructor is called when an object of
a class is created. It can be used to set initial values for
object attributes:



Example

Create a constructor:



// Create a MyClass class

public class MyClass {

  int x;  // Create a class attribute
```

```
  // Create a class constructor for the MyClass class

  public MyClass() {

    x = 5;  // Set the initial value for the class attribute x

  }



  public static void main(String[] args) {

    MyClass myObj = new MyClass(); // Create an object of class
MyClass (This will call the constructor)

    System.out.println(myObj.x); // Print the value of x

  }

}



// Outputs 5
```

3. What is object in java?

An object is a software bundle of variables and related methods.
You can represent real-world objects using software objects. You might want to represent real-world dogs as software objects in an animation program or a real-world bicycle as a software object within an electronic exercise bike. However, you can also use software objects to model abstract concepts. For example, an event is a common object used in GUI window systems to represent the action of a user pressing a mouse button or a key on the keyboard.

4. How to create object in java?

- Declaration: The code set in bold are all variable declarations that associate a variable name with an object type.
- Instantiation: The new keyword is a Java operator that creates the object.
- Initialization: The new operator is followed by a call to a constructor, which initializes the new object.

5. What is an object in java?

Software objects are conceptually similar to real-world objects: they too consist of state and related behaviour. An object stores its state in fields (variables in some programming languages) and exposes its behaviour through methods (functions in some programming languages). Methods operate on an object's internal state and serve as the primary mechanism for object-to-object communication. Hiding internal state and requiring all interaction to be performed through an object's methods is known as data encapsulation — a fundamental principle of object-oriented programming.

6. What is oops in java?

Object-oriented programming System(OOPs) is a programming paradigm based on the concept of "objects" that contain data and methods. The primary purpose of object-oriented programming is to increase the flexibility and maintainability of programs. Object-oriented programming brings together data and its behaviour(methods) in a single location(object) makes it easier to understand how a program works. We

will cover each and every feature of OOPs in detail so that you won't face any difficultly understanding OOPs Concepts.

7. Who executes the byte code in java?

Bytecode is the compiled format for Java programs. Once a Java program has been converted to bytecode, it can be transferred across a network and executed by Java Virtual Machine (JVM).

8. Why java is secure?

Java has no concept of pointers. Hence java is secure. Java does not provide access to actual memory locations.

9. Why java does not support multiple inheritance?

Java supports multiple inheritance through interfaces only. A class can implement any number of interfaces but can extend only one class.

Multiple inheritance is not supported because it leads to a deadly diamond problem. However, it can be solved but it leads to a complex system so multiple inheritance has been dropped by Java founders.

10. Why java doesn't support multiple inheritance?

Java supports multiple inheritance through interfaces only. A class can implement any number of interfaces but can extend only one class. Multiple inheritance is not supported because it leads to a deadly diamond problem.

11. Why we can’t create the object of abstract class in java?

Because an abstract class is an incomplete class (incomplete in the sense it contains abstract methods without body and output) we cannot create an instance or object; the same way we say for an interface.

12. What is Constructor Overloading?

When a class has multiple constructors with different function definitions or different parameters it is called constructor overloading.

```
import java.io.*;

import java.lang.*;

public class constructor_overloading {

    double sum;

    constructor_overloading(){

        sum=0;

    }

    constructor_overloading(int x,int y){

        sum=x+y;

    }

    constructor_overloading(double x,double y){

        sum=x+y;

    }
```

```
    void print_sum(){

        System.out.println(sum);

    }

    public static void main(String args[]){

        constructor_overloading c=new
constructor_overloading();

        c.print_sum();

        constructor_overloading c1=new
constructor_overloading(10,20);

        c1.print_sum();

        constructor_overloading c2=new
constructor_overloading(10.11,20.11);

        c2.print_sum();

    }

}
```

13. How many types of Inheritance are possible in Java?

Single, multiple, multilevel, hybrid and hierarchical inheritance are possible in java. Hybrid inheritance and hierarchical inheritance are only possible through interfaces.

14. How many types of constructor does Java support?

Java supports the following types of constructors:
– Non-Parameterized or Default Constructors
– Parameterized Constructors
– Copy constructor

15. What is a singleton class in Java? What's the benefit of making a class singleton?

A singleton class is a class in Java that can at most have one instance of it in an application. If new instances are created for the same class they point to the first instance created and thus have the same values for all attributes and properties.
Singleton classes are created to create global points of access to objects. Singleton classes find their primary usages in caching, logging, device drivers which are all entities for universal access.

16. What is the role of finalize()?

Finalize() is used for garbage collection. It's called by the Java run environment by default to clear out unused objects. This is done for memory management and clearing out the heap.

17. Explain encapsulation in Java.

Encapsulation is the process of wrapping variables and functions together into a single unit in order to hide the unnecessary details. The wrapped up

entities are called classes in java. Encapsulation is also known as data hiding because it hides the underlying intricacies.

18. Explain abstraction in Java.

Abstraction is the process of revealing the essential information and hiding the trivial details across units in java. Java has abstract classes and methods through which it does data abstraction.

19. If a child class inherits base class then are the constructor of the base class also inherited by the child class?

Constructors are not properties of a class. Hence they cannot be inherited. If one can inherit constructors then it would also mean that a child class can be created with the constructor of a parent class which can later cause referencing error when the child class is instantiated. Hence in order to avoid such complications, constructors cannot be inherited. The child class can invoke the parent class constructor by using the super keyword.

20. What is the use of super?

super() is used to invoke the superclass constructor by the subclass constructor. In this way, we do not have to create different objects for super and subclasses.

21. How is encapsulation achieved in Java?

Encapsulation is achieved by wrapping up data and code into simple wrappers called classes. Objects instantiate the class to get a copy of the class data.

22. What is an abstract class in Java?

An abstract class is a class that can only be inherited and it cannot be used for object creation. It's a type of restricted class with limited functionality.

23. How is polymorphism achieved in Java?

An example of polymorphism is the == operator which can be used to compare both numerics and strings.

24. Can the main method be declared as private in Java?

Yes, the main method can be declared as private.

25. What is an object in Java?

An object is an instance of a class in java. It shares all attributes and properties of a class.

26. What happens if we make the constructor final?

If we make the constructors final then the class variables initialized inside the constructor will become unusable. Their state cannot be changed.

27. What is constructor chaining?

constructor chaining is the process of invoking constructors of the same class or different classes inside a constructor. In this way, multiple objects are not required for constructor invocation with constructors having different parameters.

Java Multithreading Interview Questions

1. What is multithreading in java?

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

2. What is thread-safe in java?

Thread-safety or thread-safe code in Java refers to code which can safely be used or shared in concurrent or multi-threading environment and they will behave as expected. any code, class, or object which can behave differently from its contract on the concurrent environment is not thread-safe.

3. What is a thread in java?

A thread is a lightweight program that allows multiple processes to run concurrently. Every java program has at least one thread called the main thread, the main thread is created by JVM. The user can define their own

threads by extending the Thread class (or) by implementing the Runnable interface. Threads are executed concurrently.

```
public static void main(String[] args){//main thread starts
here

}
```

4. What is volatile in java?

Volatile keyword is used to modify the value of a variable by different threads. It is also used to make classes thread-safe. It means that multiple threads can use a method and instance of the classes at the same time without any problem.

5. How to generate random numbers in java within range?

```
import java.util.concurrent.ThreadLocalRandom;



// nextInt is normally exclusive of the top value,

// so add 1 to make it inclusive

int randomNum = ThreadLocalRandom.current().nextInt(min, max +
1);
```

6. If we clone objects using the assignment operator does the references differ?

When objects are cloned using the assignment operator, both objects share the same reference. Changes made to the data by one object would also be reflected in the other object.

7. Can we start a thread twice in java?

Once a thread is started, it can never be started again. Doing so will throw an IllegalThreadStateException

8. How can java threads be created?

Threads can be created by implementing the runnable interface.
Threads can also be created by extending the thread class

This brings us to the end of the Java Interview Questions. Glad to see you are now better equipped to face an interview.

Java Interview Questions FAQs

1. What should I prepare for Java interview?

There is no fixed method through which you can prepare for your upcoming Java Interview. However, understanding the basic concepts of Java is important for you to do well. The next step would be to take up a Java Beginners Course that will help you understand the concepts well, or read the top books for self-learning. Apart from learning the basic concepts

through courses, books, and blogs, you can also work on projects that will help you gain a hands-on experience.

2. What are the basics of Java?

Java is a object-oriented general-purpose programming language. It is a popular programming language because of it's easy-to-use syntax. The basics of Java include understanding what Java is, how to install Java and Java IDE, variables and data types in Java, Operators in Java, Arrays, Functions, Flow Control Statements, and basic programs. To learn the basics of Java, you can take up a Java for Beginners Course and understand the concepts required for you to build a successful career in Java Programming.

3. Why is string immutable in Java?

String is immutable in Java to ensure that the value of the string does not change. Java uses string literals, if there are multiple variables that refer to one object and the value of the object is changed, it would affect all reference variables. Hence, strings are immutable in Java.

4. Why pointers are not used in Java?

Java doesn't allow using pointers. Pointers are not used in Java to ensure that it is more secure and robust.

5. Is Java 100% object oriented language?

No. Java is not a 100% object oriented language. It follows some principles of object oriented language, but not all.

6. What are the features of Java?

The main features of Java include: multithreaded, platform independent, simple, secure, architecture neutral, portable, robust, dynamic, high-performance, and interpreted.

7. How can I learn Java easily?

Any method of learning that suits you and your learning style should be considered as the best way to learn. Different people learn well through different methods. Some individuals may prefer taking up online courses, reading books or blogs, watching YouTube videos to self-learn. And some people may also learn through practice and hands-on experience. Choose what works best for you!

8. What is API in Java?

In Java, API stands for Application Programming Interface. It is a software that allows two applications to talk to each other. It is a list of classes that is part of a JDK or Java Development Kit and includes interfaces, packages, classes, fields, methods and constructors.

9. What is Java full form?

Java is a programming language that is originally developed by James Gosling at Sun Microsystems. While the developers were drinking coffee and brainstorming names for their programming language, they chose to name it Java back in 1995. It does not have any full form.

Spring Framework Interview Question

1) What is Spring?

It is a lightweight, loosely coupled and integrated framework for developing enterprise applications in java.

---

2) What are the advantages of spring framework?

1. Predefined Templates
2. Loose Coupling
3. Easy to test
4. Lightweight
5. Fast Development
6. Powerful Abstraction
7. Declarative support

---

3) What are the modules of spring framework?

1. Test
2. Spring Core Container
3. AOP, Aspects and Instrumentation
4. Data Access/Integration
5. Web

---

4) What is IOC and DI?

IOC (Inversion of Control) and DI (Dependency Injection) is a design pattern to provide loose coupling. It removes the dependency from the program.

Let's write a code without following IOC and DI.

```
1. public class Employee{
2. Address address;
3. Employee(){
4. address=new Address();//creating instance
5. }
6. }
```

Now, there is dependency between Employee and Address because Employee is forced to use the same address instance.

Let's write the IOC or DI code.

```
1. public class Employee{
2. Address address;
3. Employee(Address address){
4. this.address=address;//not creating instance
5. }
6. }
```

Now, there is no dependency between Employee and Address because Employee is not forced to use the same address instance. It can use any address instance.

---

5) What is the role of IOC container in spring?

IOC container is responsible to:

- create the instance
- configure the instance, and
- assemble the dependencies

---

6) What are the types of IOC container in spring?

There are two types of IOC containers in spring framework.

1. BeanFactory
2. ApplicationContext

---

7) What is the difference between BeanFactory and ApplicationContext?

BeanFactory is the **basic container** whereas ApplicationContext is the **advanced container**. ApplicationContext extends the BeanFactory interface. ApplicationContext provides more facilities than BeanFactory such as integration with spring AOP, message resource handling for i18n etc.

---

8) What is the difference between constructor injection and setter injection?

No.	Constructor Injection	Setter Injection
1)	No Partial Injection	Partial Injection
2)	Desn't override the setter property	Overrides the constructor property if both a defined.
3)	Creates new instance if any modification occurs	Doesn't create new instance if you change t property value
4)	Better for too many properties	Better for few properties.

---

9) What is autowiring in spring? What are the autowiring modes?

Autowiring enables the programmer to inject the bean automatically. We don't need to write explicit injection logic.

Let's see the code to inject bean using dependency injection.

```
1. <bean id="emp" class="com.javatpoint.Employee" autowire="byName" />
```

The autowiring modes are given below:

No.	Mode	Description
1)	no	this is the default mode, it means autowiring is not enabled.
2)	byName	injects the bean based on the property name. It uses setter method.
3)	byType	injects the bean based on the property type. It uses setter method.
4)	constructor	It injects the bean using constructor

The "autodetect" mode is deprecated since spring 3.

---

10) What are the different bean scopes in spring?

There are 5 bean scopes in spring framework.

No.	Scope	Description
1)	singleton	The bean instance will be only once and same instance will be returned by t IOC container. It is the default scope.
2)	prototype	The bean instance will be created each time when requested.
3)	request	The bean instance will be created per HTTP request.
4)	session	The bean instance will be created per HTTP session.
5)	globalsession	The bean instance will be created per HTTP global session. It can be used in port context only.

---

11) In which scenario, you will use singleton and prototype scope?

Singleton scope should be used with EJB **stateless session bean** and prototype scope with EJB **stateful session bean**.

---

12) What are the transaction management supports provided by spring?

Spring framework provides two type of transaction management supports:

1. **Programmatic Transaction Management**: should be used for few transaction operations.
2. **Declarative Transaction Management**: should be used for many transaction operations.

» Spring JDBC Interview Questions

---

13) What are the advantages of JdbcTemplate in spring?

Less code: By using the JdbcTemplate class, you don't need to create connection,statement,start transaction,commit transaction and close connection to execute different queries. You can execute the query directly.

---

14) What are classes for spring JDBC API?

1. JdbcTemplate
2. SimpleJdbcTemplate
3. NamedParameterJdbcTemplate
4. SimpleJdbcInsert
5. SimpleJdbcCall

---

15) How can you fetch records by spring JdbcTemplate?

You can fetch records from the database by the **query method of JdbcTemplate**. There are two interfaces to do this:

1. ResultSetExtractor
2. RowMapper

---

16) What is the advantage of NamedParameterJdbcTemplate?

NamedParameterJdbcTemplate class is used to pass value to the named parameter. A named parameter is better than ? (question mark of PreparedStatement).

It is **better to remember**.

---

17) What is the advantage of SimpleJdbcTemplate?

The **SimpleJdbcTemplate** supports the feature of var-args and autoboxing.

» Spring AOP Interview Questions

---

18) What is AOP?

AOP is an acronym for Aspect Oriented Programming. It is a methodology that divides the program logic into pieces or parts or concerns.

It increases the modularity and the key unit is Aspect.

---

19) What are the advantages of spring AOP?

AOP enables you to dynamically add or remove concern before or after the business logic. It is **pluggable** and **easy to maintain**.

---

20) What are the AOP terminology?

AOP terminologies or concepts are as follows:

- JoinPoint
- Advice
- Pointcut
- Aspect
- Introduction
- Target Object
- Interceptor
- AOP Proxy
- Weaving

---

21) What is JoinPoint?

JoinPoint is any point in your program such as field access, method execution, exception handling etc.

---

22) Does spring framework support all JoinPoints?

No, spring framework supports method execution joinpoint only.

---

23) What is Advice?

Advice represents action taken by aspect.

---

24) What are the types of advice in AOP?

There are 5 types of advices in spring AOP.

1. Before Advice

2. After Advice
3. After Returning Advice
4. Throws Advice
5. Around Advice

---

25) What is Pointcut?

Pointcut is expression language of Spring AOP.

---

26) What is Aspect?

Aspect is a class in spring AOP that contains advices and joinpoints.

---

27) What is Introduction?

Introduction represents introduction of new fields and methods for a type.

---

28) What is target object?

Target Object is a proxy object that is advised by one or more aspects.

---

29) What is interceptor?

Interceptor is a class like aspect that contains one advice only.

---

30) What is weaving?

Weaving is a process of linking aspect with other application.

31) Does spring perform weaving at compile time?

No, spring framework performs weaving at runtime.

32) What are the AOP implementation?

There are 3 AOP implementation.

1. Spring AOP
2. Apache AspectJ
3. JBoss AOP

» Spring MVC Interview Questions

33) What is the front controller class of Spring MVC?

The **DispatcherServlet** class works as the front controller in Spring MVC.

34) What does @Controller annotation?

The **@Controller** annotation marks the class as controller class. It is applied on the class.

35) What does @RequestMapping annotation?

The **@RequestMapping** annotation maps the request with the method. It is applied on the method.

36) What does the ViewResolver class?

The **View Resolver** class resolves the view component to be invoked for the request. It defines prefix and suffix properties to resolve the view component.

---

37) Which ViewResolver class is widely used?

The **org.springframework.web.servlet.view.InternalResourceViewResolver** class is widely used.

---

38) Does spring MVC provide validation support?

Yes.

Spring-boot Interview Question

Q1. What Is Spring Boot and What Are Its Main Features?

Spring Boot is essentially a framework for rapid application development built on top of the Spring Framework. With its auto-configuration and embedded application server support, combined with the extensive documentation and community support it enjoys, Spring Boot is one of the most popular technologies in the Java ecosystem as of date.

Here are a few salient features:

- **Starters** – a set of dependency descriptors to include relevant dependencies at a go
- **Auto-configuration** – a way to automatically configure an application based on the dependencies present on the classpath
- **Actuator** – to get production-ready features such as monitoring
- **Security**
- **Logging**

Q2. What Are the Differences Between Spring and Spring Boot?

The Spring Framework provides multiple features that make the development of web applications easier. These features include dependency injection, data binding, aspect-oriented programming, data access and many more.

Over the years, Spring has been growing more and more complex, and the amount of configuration such application requires can be intimidating. This is where Spring Boot comes in handy — it makes configuring a Spring application a breeze.

Essentially, while Spring is unopinionated, **Spring Boot takes an opinionated view of the platform and libraries, letting us get started quickly.**

Here are two of the most important benefits Spring Boot brings in:

- Auto-configure applications based on the artifacts it finds on the classpath
- Provide non-functional features common to applications in production, such as security or health checks

Q3. How Can We Set Up a Spring Boot Application With Maven?

We can include Spring Boot in a Maven project just like we would any other library. However, the best way is to inherit from the *spring-boot-starter-parent* project and declare dependencies to Spring Boot starters. Doing this lets our project reuse the default settings of Spring Boot.

Inheriting the *spring-boot-starter-parent* project is straightforward — we only need to specify a *parent* element in *pom.xml*:

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.4.0.RELEASE</version>
</parent>
```

We can find the latest version of *spring-boot-starter-parent* on Maven Central.

Using the starter parent project is convenient but not always feasible. For instance, if our company requires all projects to inherit from a standard POM, we can still benefit from Spring Boot's dependency management using a custom parent.

Q4. What Is Spring Initializr?

Spring Initializr is a convenient way to create a Spring Boot project.

We can go to the Spring Initializr site, choose a dependency management tool (either Maven or Gradle), a language (Java, Kotlin or Groovy), a packaging scheme (Jar or War), version and dependencies, and download the project.

This **creates a skeleton project for us** and saves setup time so that we can concentrate on adding business logic.

Even when we use our IDE's (such as STS or Eclipse with STS plugin) new project wizard to create a Spring Boot project, it uses Spring Initializr under the hood.

Q5. What Spring Boot Starters Are Available Out There?

Each starter plays a role as a one-stop shop for all the Spring technologies we need. Other required dependencies are then transitively pulled in and managed in a consistent way.

All starters are under the *org.springframework.boot* group and their names start with *spring-boot-starter-*. **This naming pattern makes it easy to find starters, especially when working with IDEs that support searching dependencies by name.**

At the time of this writing, there are more than 50 starters at our disposal. Here, we'll list the most common:

- *spring-boot-starter*: core starter, including auto-configuration support, logging and YAML
- *spring-boot-starter-aop*: for aspect-oriented programming with Spring AOP and AspectJ
- *spring-boot-starter-data-jpa:* for using Spring Data JPA with Hibernate
- *spring-boot-starter-security*: for using Spring Security
- *spring-boot-starter-test*: for testing Spring Boot applications
- *spring-boot-starter-web*: for building web, including RESTful, applications using Spring MVC

Q6. How to Disable a Specific Auto-Configuration?

If we want to disable a specific auto-configuration, we can indicate it using the *exclude* attribute of the *@EnableAutoConfiguration* annotation.

For instance, this code snippet neutralizes *DataSourceAutoConfiguration*:

```
// other annotations
@EnableAutoConfiguration(exclude = DataSourceAutoConfiguration.class)
public class MyConfiguration { }
```

If we enabled auto-configuration with the *@SpringBootApplication* annotation — which has *@EnableAutoConfiguration* as a meta-annotation — we could disable auto-configuration with an attribute of the same name:

```
// other annotations
@SpringBootApplication(exclude = DataSourceAutoConfiguration.class)
public class MyConfiguration { }
```

We can also disable an auto-configuration with the *spring.autoconfigure.exclude* environment property. This setting in the *application.properties* file does the same thing as before:

```
spring.autoconfigure.exclude=org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration
```

Q7. How to Register a Custom Auto-Configuration?

To register an auto-configuration class, we must have its fully qualified name listed under the *EnableAutoConfiguration* key in the *META-INF/spring.factories* file:

```
org.springframework.boot.autoconfigure.EnableAutoConfiguration=com.baeldung.autoconfigure.CustomAutoConfiguration
```

If we build a project with Maven, that file should be placed in the *resources/META-INF* directory, which will end up in the mentioned location during the *package* phase.

Q8. How to Tell an Auto-Configuration to Back Away When a Bean Exists?

To instruct an auto-configuration class to back off when a bean already exists, we can use the *@ConditionalOnMissingBean* annotation.

The most noticeable attributes of this annotation are:

- *value* – the types of beans to be checked
- *name* – the names of beans to be checked

When placed on a method adorned with *@Bean*, the target type defaults to the method's return type:

```
@Configuration
public class CustomConfiguration {
```

```
    @Bean
    @ConditionalOnMissingBean
    public CustomService service() { ... }
}
```

Q9. How to Deploy Spring Boot Web Applications as Jar and War Files?

Traditionally, we package a web application as a WAR file and then deploy it into an external server. Doing this allows us to arrange multiple applications on the same server. When CPU and memory were scarce, this was a great way to save resources.

But things have changed. Computer hardware is fairly cheap now, and the attention has turned to server configuration. A small mistake in configuring the server during deployment may lead to catastrophic consequences.

Spring tackles this problem by providing a plugin, namely *spring-boot-maven-plugin*, to package a web application as an executable JAR.

To include this plugin, just add a *plugin* element to *pom.xml*:

```
<plugin>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-maven-plugin</artifactId>
</plugin>
```

With this plugin in place, we'll get a fat JAR after executing the *package* phase. This JAR contains all the necessary dependencies, including an embedded server. So, we no longer need to worry about configuring an external server.

We can then run the application just like we would an ordinary executable JAR.

Notice that the *packaging* element in the *pom.xml* file must be set to *jar* to build a JAR file:

```
<packaging>jar</packaging>
```

If we don't include this element, it also defaults to *jar*.

To build a WAR file, we change the *packaging* element to *war*:

```
<packaging>war</packaging>
```

and leave the container dependency off the packaged file:

```
<dependency>
```

```
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-tomcat</artifactId>
    <scope>provided</scope>
</dependency>
```
After executing the Maven *package* phase, we'll have a deployable WAR file.

Q10. How to Use Spring Boot for Command-Line Applications?

Just like any other Java program, a Spring Boot command-line application must have a *main* method.

This method serves as an entry point, which invokes the *SpringApplication#run* method to bootstrap the application:

```
@SpringBootApplication
public class MyApplication {
    public static void main(String[] args) {
        SpringApplication.run(MyApplication.class);
        // other statements
    }
}
```
The *SpringApplication* class then fires up a Spring container and auto-configures beans.

Notice we must pass a configuration class to the *run* method to work as the primary configuration source. By convention, this argument is the entry class itself.

After calling the *run* method, we can execute other statements as in a regular program.

Q11. What Are Possible Sources of External Configuration?

Spring Boot provides support for external configuration, allowing us to run the same application in various environments. **We can use properties files, YAML files, environment variables, system properties and command-line option arguments to specify configuration properties.**

We can then gain access to those properties using the *@Value* annotation, a bound object via the *@ConfigurationProperties* annotation, or the *Environment* abstraction.

Q12. What Does It Mean That Spring Boot Supports Relaxed Binding?

Relaxed binding in Spring Boot is applicable to the type-safe binding of configuration properties.

With relaxed binding, **the key of a property doesn't need to be an exact match of a property name.** Such an environment property can be written in camelCase, kebab-case, snake_case, or in uppercase with words separated by underscores.

For example, if a property in a bean class with the *@ConfigurationProperties* annotation is named *myProp*, it can be bound to any of these environment properties: *myProp*, *my-prop*, *my_prop*, or *MY_PROP*.

Q13. What Is Spring Boot DevTools Used For?

Spring Boot Developer Tools, or DevTools, is a set of tools making the development process easier.

To include these development-time features, we just need to add a dependency to the *pom.xml* file:

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

The *spring-boot-devtools* module is automatically disabled if the application runs in production. The repackaging of archives also excludes this module by default. So, it won't bring any overhead to our final product.

By default, DevTools applies properties suitable to a development environment. These properties disable template caching, enable debug logging for the web group, and so on. As a result, we have this sensible development-time configuration without setting any properties.

Applications using DevTools restart whenever a file on the classpath changes. This is a very helpful feature in development, as it gives quick feedback for modifications.

By default, static resources, including view templates, don't set off a restart. Instead, a resource change triggers a browser refresh. Notice this can only happen if the LiveReload extension is installed in the browser to interact with the embedded LiveReload server that DevTools contains.

Q14. How to Write Integration Tests?

When running integration tests for a Spring application, we must have an *ApplicationContext*.

To make our life easier, Spring Boot provides a special annotation for testing — *@SpringBootTest*. This annotation creates an *ApplicationContext* from configuration classes indicated by its *classes* attribute.

In case the *classes* attribute isn't set, Spring Boot searches for the primary configuration class. The search starts from the package containing the test until it finds a class annotated with *@SpringBootApplication* or *@SpringBootConfiguration*.

Q15. What Is Spring Boot Actuator Used For?

Essentially, Actuator brings Spring Boot applications to life by enabling production-ready features. **These features allow us to monitor and manage applications when they're running in production.**

Integrating Spring Boot Actuator into a project is very simple. All we need to do is include the *spring-boot-starter-actuator* starter in the *pom.xml* file:

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

Spring Boot Actuator can expose operational information using either HTTP or JMX endpoints. But most applications go for HTTP, where the identity of an endpoint and the */actuator* prefix form a URL path.

Here are some of the most common built-in endpoints Actuator provides:

- *env* exposes environment properties
- *health* shows application health information
- *httptrace* displays HTTP trace information
- *info* displays arbitrary application information
- *metrics* shows metrics information
- *loggers* shows and modifies the configuration of loggers in the application
- *mappings* displays a list of all *@RequestMapping* paths

Q16. Which Is Better to Configure a Spring Boot Project — Properties or YAML?

YAML offers many advantages over properties files:

- More clarity and better readability
- Perfect for hierarchical configuration data, which is also represented in a better, more readable format
- Support for maps, lists and scalar types
- Can include several **profiles** in the same file (since Spring Boot 2.4.0, this is possible for properties files too)

However, writing it can be a little difficult and error-prone due to its indentation rules.

Q17. What Basic Annotations Does Spring Boot Offer?

The primary annotations that Spring Boot offers reside in its *org.springframework.boot.autoconfigure* and its sub-packages.

Here are a couple of basic ones:

- *@EnableAutoConfiguration* – to make Spring Boot look for auto-configuration beans on its classpath and automatically apply them
- *@SpringBootApplication* – to denote the main class of a Boot Application. This annotation combines *@Configuration*, *@EnableAutoConfiguration* and *@ComponentScan* annotations with their default attributes.
-

Q18. How to Change the Default Port in Spring Boot?

We can change the default port of a server embedded in Spring Boot using one of these ways:

- Using a properties file – We can define this in an *application.properties* (or *application.yml*) file using the property *server.port*.
- Programmatically – In our main *@SpringBootApplication* class, we can set the *server.port* on the *SpringApplication* instance.
- Using the command line – When running the application as a jar file, we can set the server.port as a java command argument:
- `java -jar -Dserver.port=8081 myspringproject.jar`

Q19. Which Embedded Servers Does Spring Boot Support, and How to Change the Default?

As of date, **Spring MVC supports Tomcat, Jetty and Undertow.** Tomcat is the default application server supported by Spring Boot's *web* starter.

Spring WebFlux supports Reactor Netty, Tomcat, Jetty and Undertow with Reactor Netty as default.

In Spring MVC, to change the default, let's say to Jetty, we need to exclude Tomcat and include Jetty in the dependencies:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
    <exclusions>
        <exclusion>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-tomcat</artifactId>
        </exclusion>
    </exclusions>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-jetty</artifactId>
</dependency>
```

Similarly, to change the default in WebFlux to UnderTow, we need to exclude Reactor Netty and include UnderTow in the dependencies.

Q20. Why Do We Need Spring Profiles?

When developing applications for the enterprise, we typically deal with multiple environments such as Dev, QA and Prod. The configuration properties for these environments are different.

For example, we might be using an embedded H2 database for Dev, but Prod could have the proprietary Oracle or DB2. Even if the DBMS is the same across environments, the URLs would definitely be different.

To make this easy and clean, **Spring has the provision of profiles to help separate the configuration for each environment.** So, instead of maintaining this programmatically, the properties can be kept in separate files such as *application-dev.properties* and *application-prod.properties*. The default *application.properties* points to the currently active profile using *spring.profiles.active* so that the correct configuration is picked up.

JPA Interview Question

1. What is JPA?

JPA (Java Persistence API) is a Java EE and Java SE specification that describes a management system for saving java objects to relational database tables in a convenient form. Java itself does not contain JPA implementations, however there are many implementations of this specification from different companies (open and not). This is not the only way to save java objects in databases (ORM systems), but one of the most popular in the Java world.

2. What is the difference between JPA and Hibernate?

Hibernate is one of the most popular open source implementations of the latest specification (JPA 2.1). Even more likely the most popular, almost standard de facto. That is, JPA only describes rules and APIs, and Hibernate implements these descriptions, however Hibernate (like many other JPA implementations) has additional features not described in JPA (and not portable to other JPA implementations).

3. Is it possible to use JPA with noSQl databases?

In general, the JPA specification says only about mapping java objects into relational database tables, but there are a number of implementations of this standard for noSql databases: Kundera, DataNucleus, ObjectDB, and a number of others. Naturally, not all specification-specific features for relational databases are transferred to nosql databases completely.

4. What is the difference between JPA and JDO?

JPA (Java Persistence API) and Java Data Objects (JDO) are two specifications for storing java objects in databases. If JPA is concentrated only on relational databases, then JDO is a more general specification that describes the ORM for any possible bases and repositories.

In principle, JPA can be viewed as part of the JDO specification specialized in relational databases, even though the API of these two specifications does not completely match. The

“developers” of specifications also differ – if JPA is developed as JSR, then JDO was first developed as JSR, now it is developed as an Apache JDO project.

5. What is Entity?

Entity is a lightweight persistent domain object. The main program entity is the entity class, which can also use additional classes that can be used as auxiliary classes or to maintain state of the entity.

6. Can the Entity class inherit from non-Entity classes?

Can.

7. Can an Entity class inherit from other Entity classes?

The same can.

8. Can a non-Entity class inherit from an Entity class?

And this is also permissible.

9. Can Entity be an abstract class?

Perhaps, at the same time, it retains all the properties of the Entity, except that it cannot be directly initialized.

Java Persistence API Interview Questions

10. What JPA requirements for Entity classes can you list (at least six requirements)?

1) Entity class must be annotated with Entity or described in the XML configuration file JPA,

2) Entity class must contain a public or protected constructor with no arguments (it can also have constructors with arguments),

3) Entity class must be a top-level class (top -level class),

4) Entity class cannot be enum or interface,

5) Entity class cannot be final class,

6) Entity class cannot contain final fields or methods if they participate in mapping (persistent final methods or persistent final instance variables),

7) If an Entity class object is passed by value as a separate object (detached object), for example through a remote interface (through a remote interface), it must also implement a Serializable interface,

8) The Entity class fields should be directly accessible only to the methods of the Entity class and should not be directly accessible to other classes using this entity. Such classes should refer only to methods (getter / setter methods or other business logic methods in the Entity class),

9) The Enity class must contain a primary key, that is, an attribute or group of attributes that uniquely defines the record of this Enity class in the database.

11. What are the two types of elements in Entity classes. Or in other words, list two types of access (access) to the elements of the Entity classes.

JPA indicates that it can work both with properties of classes (property), designed in the style of JavaBeans, or with fields (field), that is, class variables (instance variables). Accordingly, the type of access will be either property access or field access.

12. What is the attribute of the Entity class in JPA terminology?

JPA indicates that it can work both with properties of classes (property), designed in the style of JavaBeans, or with fields (field), that is, class variables (instance variables). Both types of elements of the Entity class are called attributes of the Entity class.

13. What data types are allowed in the attributes of the Entity class (fields or properties)?

Valid attribute types for Entity classes are:

1. primitive types and their Java wrappers,
2. strings,
3. any Java serializable types (implementing the Serializable interface),
4. enums;
5. entity types;
6. embeddable classes
7. and collection types 1-6

14. What data types can be used in the attributes included in the primary key of the Entity class (composite or simple) so that the resulting primary key can be used for any database? And in the case of auto-generated primary key (generated primary keys)?

Valid attribute types included in the primary key are:

1. primitive types and their Java wrappers,
2. strings,
3. BigDecimal and BigInteger,
4. java.util.Date and java.sql.Date

In the case of an auto-generated primary key (generated primary keys), only numeric types are allowed.

If other types of data are used in the primary key, it can work only for some databases, i.e. becomes not portable (not portable).

Question 15. What is the Embeddable class?

An embeddable class is a class that is not used by itself, only as part of one or more Entity classes. An entity class can contain both single embedded classes and collections of such classes. Also such classes can be used as keys or map values. At run time, each embedded class belongs to only one object of the Entity class and cannot be used to transfer data between the objects of the Entity classes (that is, such a class is not a common data structure

for different objects). In general, such a class serves to make the definition of common attributes for several Entity, we can assume that JPA simply embeds into the Entity instead of an object of this class, the attributes it contains.

16. Can the Embeddable class contain another Embeddable class?

Yes maybe.

17. Can the Embeddable class contain relationships with other Entity or Entity collections? If it can, are there any restrictions on such relationships?

Maybe, but only if such a class is not used as the primary key or the map key.

18. What requirements does JPA set for Embeddable classes?

1. Such classes must satisfy the same rules as the Entity classes, except that they do not have to contain a primary key and be marked with the Entity annotation (see question 10),
2. The Embeddable class must be marked with the Embeddable annotation or described in the XML configuration file JPA.

19. What types of connections (relationship) between Entity do you know (list eight types, or specify four types of connections, each of which can be further divided into two types)?

There are four types of connections

1. OneToOne (one-to-one connection, that is, one Entity object can be associated with no more than one object of another Entity),

2. OneToMany (one-to-many connection, one Entity object can be associated with a whole collection of Entity)
3. ManyToOne (many to one link, feedback for OneToMany),
4. ManyToMany (many to many link) Each of which can be divided into two types:
5. Bidirectional

6. Unidirectional – a link to a link is set for all Entity, that is, in the case of OneToOne AB, Entity A has a link to Entity B, Entity B has a link to Entity A, Entity A is considered the owner of this link (this is important for cases of cascading data deletion , then deleting A will also delete B, but not vice versa). Undirectional- link to link is established only on one side, that is, in the case of OneToOne AB only Entity A will have link to Entity B, Entity B will not have link to A .

Java Persistence API Interview Questions Ans Answers

20. What is Mapped Superclass?

Mapped Superclass is a class from which Entity is inherited, it may contain JPA annotations, however such a class is not Entity, it does not have to fulfill all the requirements set for Entity (for example, it may not contain a primary key). Such a class cannot be used in EntityManager or Query operations. Such a class should be marked with the MappedSuperclass annotation or, respectively, described in the xml file.

21. What are the three types of Inheritance Mapping Strategies described in JPA?

JPA describes three inheritance mapping strategies (Inheritance Mapping Strategies), that is, how JPA will work with the classes derived from the Entity:

1) one table for the entire inheritance hierarchy (**a single table per class hierarchy**) – all enity one table, to identify the type of entity is determined by a special column **“discriminator column”** . For example, if there is an entity Animals with the descendant classes Cats and Dogs, with such a strategy, all entities are recorded in the Animals table, but they have an additional column, animalType, in which the value “cat” or “dog” is written respectively. **Minus**is that in the general table, all fields unique for each of the descendant classes will be created, which will be empty for all other descendant classes. For example, in the table of animals there will be the speed of climbing a tree from cats and whether the dog can bring sneakers from dogs, which will always be null for a dog and cat, respectively.

2) unifying strategy (**joined subclass strategy**) – in this strategy, each enity class stores data in its own table, but only unique columns (not inherited from ancestor classes) and the primary key, and all inherited columns are written to ancestor class tables a relationship is established between these tables, for example, in the case of the Animals classes (see above), there will be three tables of animals, cats, dogs, and only the key and speed of climbing will be recorded in cats, in dogs – the key and whether the dog can bring a stick , and in animals all the rest of the data of cats and dogs are c links th to the appropriate table. **The downside** is the performance loss from joining tables (join) for any operations.

3) one table for each class (**table per concrete class strategy**) – everything is simple here every individual class of successor has its own table, i.e. for cats and dogs (see above), all data will be recorded simply in the tables cats and dogs as if they did not have a common superclass at all. **The downside** is poor support for polymorphism (polymorphic relationships) and the fact that selecting all the classes in the hierarchy will require a large number of separate sql queries or using a UNION query.

22. What are the two types of fetch strategies in JPA do you know?

JPA describes two types of fetch strategies:

1) LAZY – these fields will be loaded only during the first access to this field,

2) EAGER – these fields will be loaded immediately.

23. What is EntityManager and what are its main functions you can list?

EntityManager is an interface that describes an API for all basic operations on Enitity, data retrieval and other JPA entities. Essentially the main API for working with JPA. Basic operations:

1) For operations on Entity: persist (adding Entity under JPA control), merge (updating), remove (delete), refresh (updating data), detach (removing from management JPA), lock (blocking Enity from changes in other thread),

2) data Preparation: find (search and retrieval entity), createQuery, createNamedQuery, createNativeQuery , contains, createNamedStoredProcedureQuery, createStoredProcedureQuery

3) Preparation of other entities JPA: getTransaction, getEntityManagerFactory, getCriteriaBuilder, getMetamodel, getDelegate

4) Work with EntityGraph: createEntityGraph, getEntityGraph

4) General operations on EntityManager or all Entities: close, isOpen, getProperties, setProperty, clear.

24. What are the four lifecycle status of an Entity Instance's Life Cycle you can list?

An Entity object has four lifecycle status: new, managed, detached, or removed. Their description

1) new – the object has been created but has not yet generated primary keys and has not yet been saved in the database,

2) managed – the object has been created, managed by JPA, has generated primary keys,

3) detached – the object has been created, but not managed (or no longer managed) JPA,

4) removed – the object is created, managed by JPA, but will be deleted after commit'a transaction.

Advanced JPA Interview Questions And Answers

25. How does the operation persist on Entity objects of each of the four statuses?

1) If the status is Entity new, then it changes to managed and the object will be saved to the database when a transaction is committed or as a result of flush operations,

2) If the status is already managed, the operation is ignored, but dependent Entity can change the status to managed, if there are annotations of cascading changes,

3) If the status is removed, then it changes to managed,

4) If the status is detached, the exception will be thrown right away or at the commit stage of the transaction.

26. How does the remove operation affect the Entity objects of each of the four statuses?

1) If the status is Entity new, the operation is ignored, however dependent Entity can change the status to removed, if they have cascading change annotations and they had the status managed,

2) If the status is managed, then the status changes to removed and the object is recorded in the database removed during commit commit (remove operations will also occur for all cascade-dependent objects),

3) If status is removed, then the operation is ignored,

4) If detached, exception is thrown right away or at the commit stage of a transaction.

27. How does the operation merge affect the Entity objects of each of the four statuses?

1) If the detached state, then either the data will be copied to the existing managed entity with the same primary key, or a new managed is created into which the data is copied,

1) If the Entity is new, then a new managed entity will be created into which the past data will be copied object,

2) If the status is managed, the operation is ignored, but the operation merge will work on the cascade-dependent Entity, if their status is not managed,

3) If the status is removed, exception will be thrown immediately or at the commit stage of the transaction.

28. How does the refresh operation affect the Entity objects of each of the four statuses?

1) If the status is Entity managed, then as a result of the operation all changes from the database of this Entity will be restored, also a refresh of all cascade-dependent objects will occur,

2) If the status is new, removed or detached, exception will be thrown out.

29. How does the detach operation affect the Entity objects of each of the four statuses?

1) If the status is Entity managed or removed, then as a result of the operation, the status of Entity (and all cascade-dependent objects) will become detached.

2) If the status is new or detached, then the operation is ignored.

30. What is the Basic annotation for?

Basic – indicates the simplest type of data mapping on a database table column. Also in the annotation parameters you can specify fetch field access strategy and whether this field is required or not.

JPA Interview Questions And Answers For Experienced

31. What is the Access annotation for?

It defines the access type for an entity class, superclass, embeddable, or individual attributes, that is, how JPA will refer to entity attributes, like class fields (FIELD) or class properties (PROPERTY) that have getters (getter) and setters.

32. What annotations can overlap connections (override entity relationship) or attributes inherited from the superclass, or specified in the embeddable class when using this embeddable class in one of the entity classes and do not overlap in the others?

There are four annotations for this overlap:

1. AttributeOverride to overlap fields, properties and primary keys,
2. AttributeOverrides can similarly

override fields, properties and primary keys with multiple values, 3. AssociationOverride to override entity relationship,

4. AssociationOverrides to overlap multiple relationships.

33. What annotation can I manage JPA caching for this Entity?

Cacheable – allows you to enable or disable the use of a second-level cache (second-level cache) for this Entity (if the JPA provider supports caching and cache settings (second-level cache) are ENABLE_SELECTIVE or DISABLE_SELECTIVE, see question 41). Note that the property is inherited and if it is not blocked by the heirs, then caching will change for them too.

34. What annotations are used to set the class for converting the basic attribute of the Entity to another type when saving / retrieving data of their base (for example, work with the attribute of the Entity boolean type, but save it as a number to the base)?

Convert and Converts – allow you to specify a class for converting the Basic attribute of an Entity into another type (Converts – allow you to specify several conversion classes). Classes for conversion must implement the AttributeConverter interface and can be marked (but not required) by the Converter annotation.

35. What annotation can be used to define a class, the methods of which should be executed under certain JPA operations on an Enitity or Mapped Superclass data (such as deleting, changing data, etc.)?

The EntityListeners annotation allows you to set a Listener class that will contain event handling methods (callback methods) defined by the Entity or Mapped Superclass.

36. What are callback methods in JPA for? To which entities do callback method annotations apply? List seven callback methods (or, equivalently, annotations of callback methods)

Callback methods are used to call on certain Entity events (that is, add processing, for example, deletion of Entity methods by JPA), can be added to the entity class, to the mapped superclass, or to the callback listener class specified by the EntityListeners annotation (see previous question). There are seven callback methods (and annotations with the same names):

1) PrePersist

2) PostPersist

3) PreRemove

4) PostRemove

5) PreUpdate

6) PostUpdate

7) PostLoad

37. What annotations are used to establish the procedure for issuing elements of the Entity collections?

The OrderBy and OrderColumn annotations are used for this.

38. What annotation can exclude fields and properties of Entity from mapping (property or field is not persistent)?

For this is the abstract Transient.

39. What are the six types of locks (lock) described in the JPA specification (or what values does the enum LockModeType in JPA have)?

JPA has six types of locks, list them in order of increasing reliability (from the most unreliable and fast, to the most reliable and slow):

1) NONE – without blocking

2) OPTIMISTIC (or READ synonym for JPA 1) – optimistic blocking

3) OPTIMISTIC_FORCE_INCREMENT (or WRITE synonym, remaining from JPA 1) – optimistic lock with forced increase of the version field,

4) PESSIMISTIC_READ – pessimistic lock for reading,

5) PESSIMISTIC_WRITE – pessimistic lock for write (and read),

6) PESSIMISTICI-PII-IZIETIHI, IHIETI , I, I, I, I, I, I, I, I; on record (and th with forced increase in the field of versioning.

Java Persistence API (JPA) Interview Questions And Answers

40. What are the two types of cache (cache) you know in JPA and what are they for?

JPA talks about two kinds of caches (cache):

1) first-level cache (first-level cache) —caches data from a single transaction;

2) second-level cache (second-level cache) —caches data for more than one transaction. The JPA provider can, but is not required to implement work with the second-level cache. This kind of cache can save access time and improve performance, but the downside is the ability to get outdated data.

41. What are the options for setting the second-level cache (second-level cache) in JPA or what similarly describe what values can the shared-cache-mode element take from persistence.xml?

JPA speaks of the five shared-cache-mode values from persistence.xml, which defines how the second-level cache will be used:

1) ALL – all Entity can be cached in the second-level cache ,

2) NONE – caching is disabled for all Entity,

3) ENABLE_SELECTIVE — caching only works for those Entities that have the Cacheable (true) or their xml equivalent set, for all others, caching is disabled,

4) DISABLE_SELECTIVE — caching works for all Entity, except for those that have the Cacheable annotation (false) or its xml equivalent is

5) UNSPECIFIED – caching is not defined, each provider EPA uses its default value for caching,

42. How can you change the fetch strategy settings of any Entity attributes for individual queries (query) or search methods (find), then if Enity has an attribute with fetchType = LAZY, but for a specific query you need to make it EAGER or vice versa?

For this, the EntityGraph API exists, it is used like this: using the NamedEntityGraph annotation for an Entity, it creates named EntityGraph objects that contain a list of attributes that need to change fetchType to EAGER, and then this name is specified in hits queries or the find method. As a result, the fetchType attribute of the Entity changes, but only for this request. There are two standard properties for specifying EntityGraph in hit:

1) javax.persistence.fetchgraph – all attributes listed in EntityGraph change fetchType to EAGER, all others to LAZY

2) javax.persistence.loadgraph – all attributes listed in EntityGraph change fetchType to EAGER, all others retain their fetchType (that is, if the attribute not specified in EntityGraph has fetchType was EAGER, then it will remain EAGER) With NamedSubgraph you can also change fetchType nested Entity objects.

43. How is it possible in the code to work with the second-level cache (delete all or certain Entity from the cache, find out if this Entity has been cached, etc.)?

To work with the second level cache (second level cache) in JPA, the Cache interface is described which contains a large number of methods for managing the second level cache (second level cache), if supported by the JPA provider, of course. The object of this interface can be retrieved using the getCache method from EntityManagerFactory.

44. How can I get JPA metadata (information about Entity types, Embeddable and Managed classes, etc.)?

JPA uses the Metamodel interface to get this information. The object of this interface can be obtained by the getMetamodel method from an EntityManagerFactory or EntityManager.

45. What is JPQL (Java Persistence query language) and how is it different from SQL?

JPQL (Java Persistence query language) is a query language, almost the same as SQL, but instead of the names and columns of database tables, it uses the names of the Entity classes and their attributes. The query parameters also use the data types of the attributes of the Entity, and not the database fields. Unlike SQL, JPQL has automatic polymorphism (see the next question). JPQL also uses functions that are not found in SQL: such as KEY (Map key), VALUE (Map value), TREAT (to cast a superclass to its heir object, downcasting), ENTRY, etc.

46. What is meant by polymorphism (polymorphism) in JPQL queries (Java Persistence query language) and how to “turn it off”?

Unlike SQL, JPQL queries have automatic polymorphism, that is, each Entity request returns not only the objects of this Entity, but also objects of all its descendant classes, regardless of the inheritance strategy (for example, the select * from Animal query will return not only Animal objects, but also objects of Cat and Dog classes that are inherited from Animal). To eliminate this behavior, the TYPE function is used in the where condition (for example, select * from Animal a where TYPE (a) IN (Animal, Cat) will not return objects of the class Dog).

47. What is the Criteria API and what is it used for?

The Criteria API is also a query language, similar to JPQL (Java Persistence query language), but the queries are based on methods and objects, that is, the queries look like this:

```
1  CriteriaBuilder cb = …
2  CriteriaQuery&lt;Customer&gt; q = cb.createQuery(Customer.class);
3  Root&lt;Customer&gt; customer = q.from(Customer.class);
4  q.select(customer);
```

48. What is the difference in Hibernate Entity requirements from the Entity requirements in the JPA specification (see question 10)?

1) A constructor with no arguments in Hibernate is not required to be public or protected, it is recommended that it be at least a package of visibility, but this is only a recommendation, if the Java security settings allow access to private fields, then it can be private.

2) JPA categorically requires not to use final classes, Hibernate only recommends not using such classes so that it can create a proxy for lazy loading, but it allows you to either turn off the Proxy proxy (lazy = false), or use an interface containing all mapping methods of this class (proxy annotation (proxyClass = interface.class))

49. What is the unique inheritance strategy in Hibernate, but not in the JPA specification?

Unlike JPA, Hibernate has a unique inheritance strategy called implicit polymorphism.

50. What are the main new features in the JPA 2.1 specification compared with JPA 2.0 (list at least five or six new features)?

In the JPA 2.1 specification, there are:

1) Entity Graphs – a dynamic change fetchType mechanism for each request,

2) Converters – a mechanism for defining converters for specifying functions for converting Entity attributes into database fields,

3) DDL generation – automatic generation of tables, indexes and schemas,

4) Stored Procedures – a mechanism for calling stored procedures from JPA,

5) Criteria Update / Delete – a mechanism for invoking bulk updates or deletes using the Criteria API,

6) Unsynchronized persistence contexts – an opportunity to specify SynchronizationType,

7) New features in JPQL / Criteria API : arithmetic subqueries, generic database functions, join ON clause, TREAT function,

8) Dynamic creation of named queries (Details queries) Learn more about changing interfaces and APIs in JPA 2.1:

1) The EntityManager interface received new methods createStoredProcedureQuery, isJoinedToTransaction and createQuery (CriteriaUpdate or CriteriaDelete)

2) The Abstract AbstractQuery class inherited from the CommonAbritCritery class interfaces CriteriaUpdate, CriteriaDelete inherited CommonAbstractCriteria,

3) PersistenceProvider got new features generateSchema possible to generate diagrams,

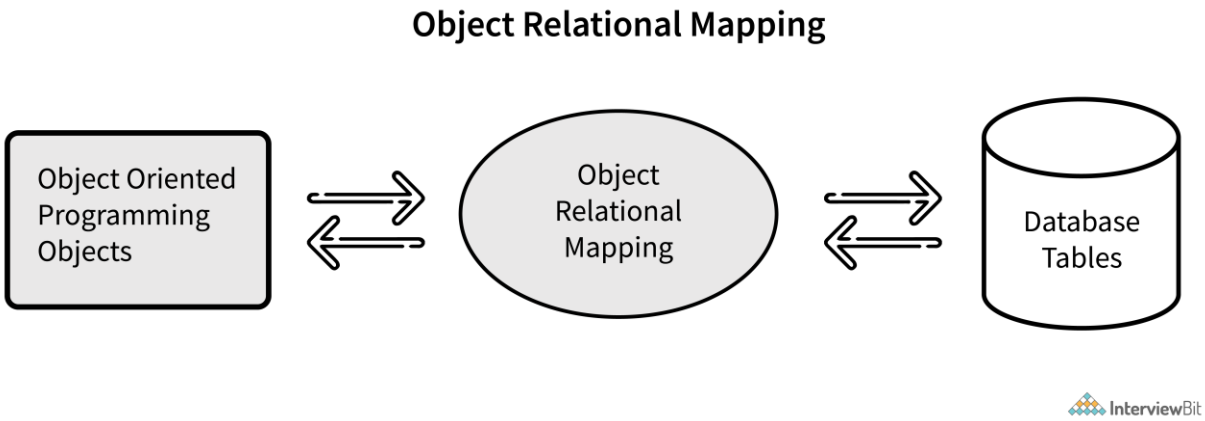
4) received EntityManagerFactory methods addNamedQuery, unwrap, addNamedEntityGraph, createEntityManager (indicating SynchronizationType)

5) a new enum SynchronizationType, Entity Graphs, StoredProced ureQuery and AttributeConverter interfaces.

Hibernate Interview Question

1. What is ORM in Hibernate?

Hibernate ORM stands for **Object Relational Mapping**. This is a mapping tool pattern mainly used for converting data stored in a relational database to an object used in object-oriented programming constructs. This tool also helps greatly in simplifying data retrieval, creation, and manipulation.



Object Relational Mapping

2. What are the advantages of Hibernate over JDBC?

- The advantages of Hibernate over JDBC are listed below:

- **Clean Readable Code:** Using hibernate, helps in eliminating a lot of JDBC API-based boiler-plate codes, thereby making the code look cleaner and readable.
- **HQL (Hibernate Query Language):** Hibernate provides HQL which is closer to Java and is object-oriented in nature. This helps in reducing the burden on developers for writing database independent queries. In JDBC, this is not the case. A developer has to know the database-specific codes.
- **Transaction Management:** JDBC doesn't support implicit transaction management. It is upon the developer to write transaction management code using commit and rollback methods. Whereas, Hibernate implicity provides this feature.
- **Exception Handling:** Hibernate wraps the JDBC exceptions and throws unchecked exceptions like JDBCException or HibernateException. This along with the built-in transaction management system helps developers to avoid writing multiple try-catch blocks to handle exceptions. In the case of JDBC, it

throws a checked exception called SQLException thereby mandating the developer to write try-catch blocks to handle this exception at compile time.
- **Special Features:** Hibernate supports OOPs features like inheritance, associations and also supports collections. These are not available in JDBC.

3. What are some of the important interfaces of Hibernate framework?

Hibernate core interfaces are:

- Configuration
- SessionFactory
- Session
- Criteria
- Query
- Transaction

4. What is a Session in Hibernate?

A session is an object that maintains the connection between Java object application and database. Session also has methods for storing, retrieving, modifying or deleting data from database using methods like persist(), load(), get(), update(), delete(), etc. Additionally, It has factory methods to return Query, Criteria, and Transaction objects.

5. What is a SessionFactory?

SessionFactory provides an instance of **Session**. It is a factory class that gives the Session **objects** based on the configuration parameters in order to establish the connection to the database.
As a good practice, the application generally has a single instance of SessionFactory. The internal state of a SessionFactory which includes metadata about ORM is immutable, i.e once the instance is created, it cannot be changed.

This also provides the facility to get information like statistics and metadata related to a class, query executions, etc. It also holds second-level cache data if enabled.

6. What do you think about the statement - “session being a thread-safe object”?

No, Session is not a thread-safe object which means that any number of threads can access data from it simultaneously.

7. Can you explain what is lazy loading in hibernate?

Lazy loading is mainly used for improving the application performance by helping to load the child objects on demand.

It is to be noted that, since Hibernate 3 version, this feature has been enabled by default. This signifies that child objects are not loaded until the parent gets loaded.

8. What is the difference between first level cache and second level cache?

Hibernate has 2 cache types. First level and second level cache for which the difference is given below:

First Level Cache	Second Level Cache
This is local to the Session object and cannot be shared between multiple sessions.	This cache is maintained at the SessionFactory level and shared among all sessions in Hibernate.
This cache is enabled by default and there is no way to disable it.	This is disabled by default, but we can enable it through configuration.
The first level cache is available only until the session is open, once the session is closed, the first level cache is destroyed.	The second-level cache is available through the application's life cycle, it is only destroyed and recreated when an application is restarted.

If an entity or object is loaded by calling the get() method then Hibernate first checked the first level cache, if it doesn't find the object then it goes to the second level cache if configured. If the object is not found then it finally goes to the database and returns the object, if there is no corresponding row in the table then it returns null.

9. What can you tell about Hibernate Configuration File?

Hibernate Configuration File or **hibernate.cfg.xml** is one of the most required configuration files in Hibernate. By default, this file is placed under the src/main/resource folder.
The file contains database related configurations and session-related configurations. Hibernate facilitates providing the configuration either in an XML file (like hibernate.cfg.xml) or a properties file (like hibernate.properties).

This file is used to define the below information:

- Database connection details: Driver class, URL, username, and password.
- There must be one configuration file for each database used in the application, suppose if we want to connect with 2 databases, then we must create 2 configuration files with different names.
- Hibernate properties: Dialect, show_sql, second_level_cache, and mapping file names.

10. How do you create an immutable class in hibernate?

Immutable class in hibernate creation could be in the following way. If we are using the XML form of configuration, then a class can be made immutable by markingmutable=false. The default value is true there which indicating that the class was not created by default.

In the case of using annotations, immutable classes in hibernate can also be created by using @Immutable annotation.

11. Can you explain the concept behind Hibernate Inheritance Mapping?

Java is an Object-Oriented Programming Language and Inheritance is one of the most important pillars of object-oriented principles. To represent any models in Java, inheritance is most commonly used to simplify and simplify the relationship. But, there is a catch. Relational databases do not support inheritance. They have a flat structure.

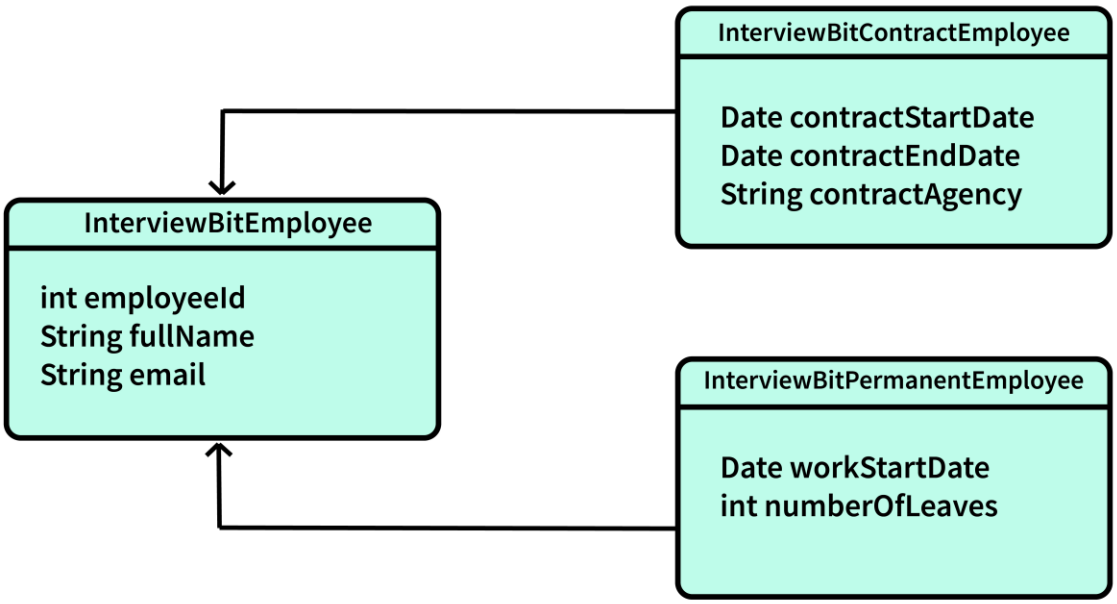
Hibernate's Inheritance Mapping strategies deal with solving how to hibernate being an ORM tries to map this problem between the inheritance of Java and flat structure of Databases.

Consider the example where we have to divide InterviewBitEmployee into Contract and Permanent Employees represented by IBContractEmployee and IBPermanentEmployee classes respectively. Now the task of hibernate is to represent these 2 employee types by considering the below restrictions:

The general employee details are defined in the parent InterviewBitEmployee class.

Contract and Permanent employee-specific details are stored in IBContractEmployee and IBPermanentEmployee classes respectively

The class diagram of this system is as shown below:



InterviewBit

Hibernate's Inheritance Mapping

There are different inheritance mapping strategies available:

- Single Table Strategy
- Table Per Class Strategy
- Mapped Super Class Strategy
- Joined Table Strategy

12. Is hibernate prone to SQL injection attack?

SQL injection attack is a serious vulnerability in terms of web security wherein an attacker can interfere with the queries made by an application/website to its database thereby allowing the attacker to view sensitive data which are generally irretrievable. It can also give the attacker to modify/ remove the data resulting in damages to the application behavior.

Hibernate does not provide immunity to SQL Injection. However, following good practices avoids SQL injection attacks. It is always advisable to follow any of the below options:

- Incorporate Prepared Statements that use Parameterized Queries.
- Use Stored Procedures.
- Ensure data sanity by doing input validation.

Intermediate Interview Questions

13. Explain hibernate mapping file

Hibernate mapping file is an XML file that is used for defining the entity bean fields and corresponding database column mappings.
These files are useful when the project uses third-party classes where JPA annotations provided by hibernating cannot be used.

In the previous example, we have defined the mapping resource as "InterviewBitEmployee.hbm.xml" in the config file. Let us see what that sample hbm.xml file looks like:

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
   <!-- What class is mapped to what database table-->
  <class name = "InterviewBitEmployee" table = "InterviewBitEmployee">

     <meta attribute = "class-description">
        This class contains the details of employees of InterviewBit.
     </meta>

     <id name = "id" type = "int" column = "employee_id">
        <generator class="native"/>
     </id>

     <property name = "fullName" column = "full_name" type = "string"/>
     <property name = "email" column = "email" type = "string"/>

  </class>
</hibernate-mapping>
```

14. What are the most commonly used annotations available to support hibernate mapping?

Hibernate framework provides support to JPA annotations and other useful annotations in the org.hibernate.annotations package. Some of them are as follows:

- javax.persistence.Entity: This annotation is used on the model classes by using "@Entity" and tells that the classes are entity beans.
- javax.persistence.Table: This annotation is used on the model classes by using "@Table" and tells that the class maps to the table name in the database.
- javax.persistence.Access: This is used as "@Access" and is used for defining the access type of either field or property. When nothing is specified, the default value taken is "field".
- javax.persistence.Id: This is used as "@Id" and is used on the attribute in a class to indicate that attribute is the primary key in the bean entity.
- javax.persistence.EmbeddedId: Used as "@EmbeddedId" upon the attribute and indicates it is a composite primary key of the bean entity.
- javax.persistence.Column: "@Column" is used for defining the column name in the database table.
- javax.persistence.GeneratedValue: "@GeneratedValue" is used for defining the strategy used for primary key generation. This annotation is used along with javax.persistence.GenerationType enum.
- javax.persistence.OneToOne: "@OneToOne" is used for defining the one-to-one mapping between two bean entities. Similarly, hibernate provides OneToMany, ManyToOne and ManyToMany annotations for defining different mapping types.
  org.hibernate.annotations.Cascade: "@Cascade" annotation is used for defining the cascading action between two bean entities. It is used with org.hibernate.annotations.CascadeType enum to define the type of cascading.
  Following is a sample class where we have used the above listed annotations:

```
package com.dev.interviewbit.model;
import javax.persistence.Access;
import javax.persistence.AccessType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToOne;
import javax.persistence.Table;

import org.hibernate.annotations.Cascade;

@Entity
@Table(name = "InterviewBitEmployee")
@Access(value=AccessType.FIELD)
public class InterviewBitEmployee {

   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   @Column(name = "employee_id")
   private long id;
```

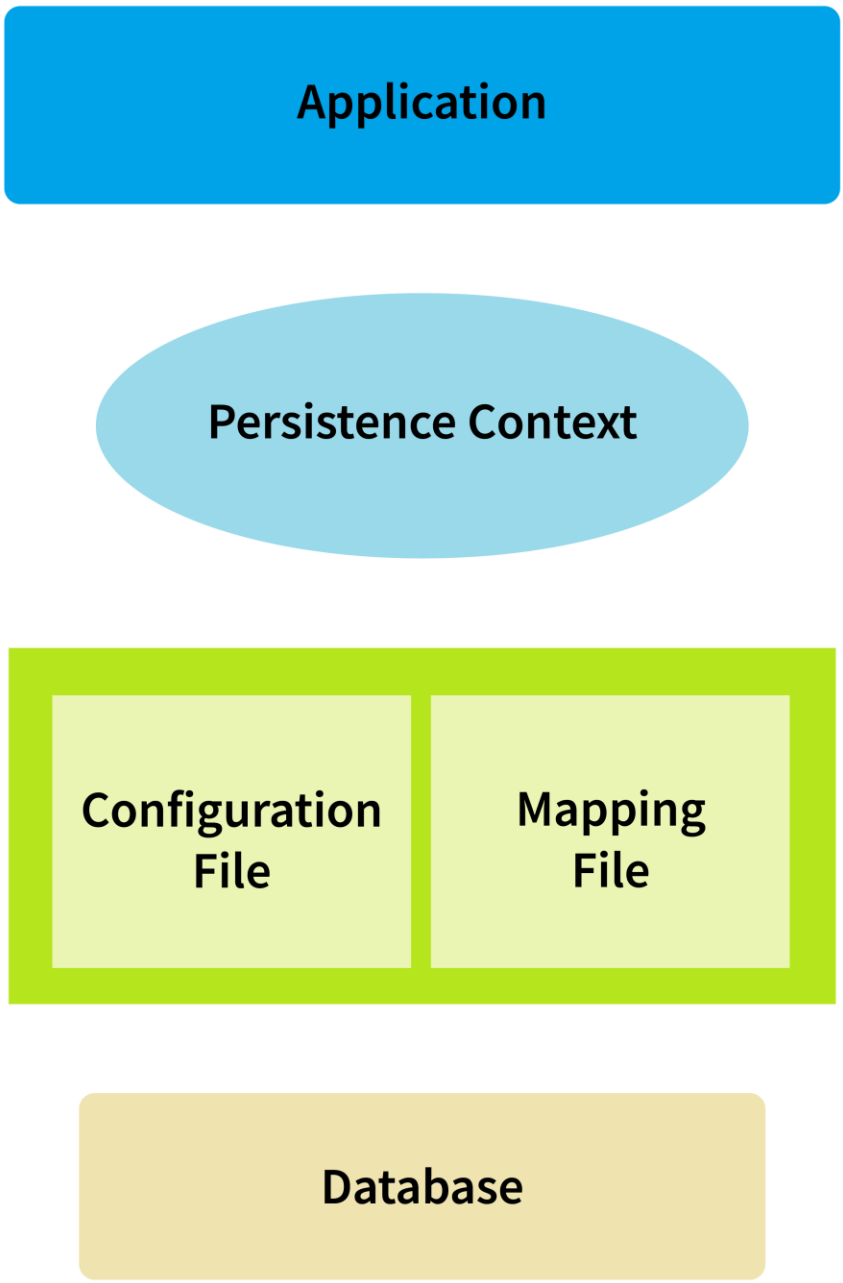
```
    @Column(name = "full_name")
    private String fullName;

    @Column(name = "email")
    private String email;

    @OneToOne(mappedBy = "employee")
    @Cascade(value = org.hibernate.annotations.CascadeType.ALL)
    private Address address;

    //getters and setters methods
}
```

15. Explain Hibernate architecture

The Hibernate architecture consists of many objects such as a persistent object, session factory, session, query, transaction, etc. Applications developed using Hibernate is mainly categorized into 4 parts:

- Java Application
- Hibernate framework - Configuration and Mapping Files
- Internal API -
    - JDBC (Java Database Connectivity)
    - JTA (Java Transaction API)
    - JNDI (Java Naming Directory Interface).
- Database - MySQL, PostGreSQL, Oracle, etc

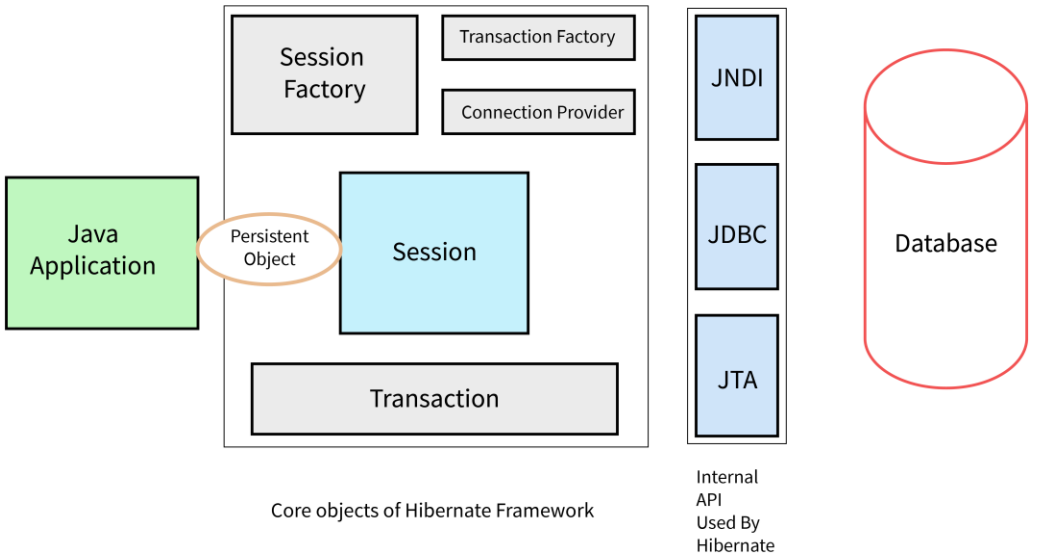
High Level Hibernate-Based Application Architecture



Architecture

The main elements of Hibernate framework are:

- SessionFactory: This provides a factory method to get session objects and clients of ConnectionProvider. It holds a second-level cache (optional) of data.
- Session: This is a short-lived object that acts as an interface between the java application objects and database data.
    - The session can be used to generate transaction, query, and criteria objects.
    - It also has a mandatory first-level cache of data.
- Transaction: This object specifies the atomic unit of work and has methods useful for transaction management. This is optional.
- ConnectionProvider: This is a factory of JDBC connection objects and it provides an abstraction to the application from the DriverManager. This is optional.
- TransactionFactory: This is a factory of Transaction objects. It is optional.



Core objects of Hibernate Framework

Hibernate Objects

16. Can you tell the difference between getCurrentSession and openSession methods?

Both the methods are provided by the Session Factory. The main differences are given below:

getCurrentSession()	openSession()
This method returns the session bound to the context.	This method always opens a new session.
This session object scope belongs to the hibernate context and to make this work hibernate configuration file has to be modified by adding **<property name = "hibernate.current_session_context_class"> thread </property>**. If not added, then using the method would throw an HibernateException.	A new session object has to be created for each request in a multi-threaded environment. Hence, you need not configure any property to call this method.
This session object gets closed once the session factory is closed.	It's the developer's responsibility to close this object once all the database operations are done.
In a single-threaded environment, this method is faster than openSession().	In single threaded environment, it is slower than getCurrentSession()single-threadeda

Apart from these two methods, there is another method openStatelessSession() and this method returns a stateless session object.

17. Differentiate between save() and saveOrUpdate() methods in hibernate session.

Both the methods save records to the table in the database in case there are no records with the primary key in the table. However, the main differences between these two are listed below:

save()	saveOrUpdate()
save() generates a new identifier and INSERT record into a database	Session.saveOrUpdate() can either INSERT or UPDATE based upon existence of a record.

save()	saveOrUpdate()
The insertion fails if the primary key already exists in the table.	In case the primary key already exists, then the record is updated.
The return type is Serializable which is the newly generated identifier id value as a Serializable object.	The return type of the saveOrUpdate() method is void.
This method is used to bring only a transient object to a persistent state.	This method can bring both transient (new) and detached (existing) objects into a persistent state. It is often used to re-attach a detached object into a Session

Clearly, saveOrUpdate() is more flexible in terms of use but it involves extra processing to find out whether a record already exists in the table or not.

18. Differentiate between get() and load() in Hibernate session

These are the methods to get data from the database. The primary differences between get and load in Hibernate are given below:

get()	load()
This method gets the data from the database as soon as it is called.	This method returns a proxy object and loads the data only when it is required.
The database is hit every time the method is called.	The database is hit only when it is really needed and this is called Lazy Loading which makes the method better.
The method returns null if the object is not found.	The method throws ObjectNotFoundException if the object is not found.
This method should be used if we are unsure about the existence of data in the database.	This method is to be used when we know for sure that the data is present in the database.

19. What is criteria API in hibernate?

Criteria API in Hibernate helps developers to build **dynamic criteria queries** on the persistence database. Criteria API is a more powerful and flexible alternative to HQL (Hibernate Query Language) queries for creating dynamic queries.

This API allows to programmatically development criteria query objects. The org.hibernate.Criteria interface is used for these purposes. The Session interface of hibernate framework has createCriteria() method that takes the persistent object's class or its entity name as the parameters and returns persistence object instance the criteria query is executed.

It also makes it very easy to incorporate restrictions to selectively retrieve data from the database. It can be achieved by using the add() method which accepts the org.hibernate.criterion.Criterion object representing individual restriction.

Usage examples:

To return all the data of InterviewBitEmployee entity class.

```
Criteria criteria = session.createCriteria(InterviewBitEmployee.class);
List<InterviewBitEmployee> results = criteria.list();
```

To retrive objects whose property has value equal to the restriction, we use Restrictions.eq() method. For example, to fetch all records with name 'Hibernate':

```
Criteria criteria= session.createCriteria(InterviewBitEmployee.class);
criteria.add(Restrictions.eq("fullName","Hibernate"));
List<InterviewBitEmployee> results = criteria.list();
```

To get objects whose property has the value "not equal to" the restriction, we use Restrictions.ne() method. For example, to fetch all the records whose employee's name is not Hibernate:

```
Criteria criteria= session.createCriteria(InterviewBitEmployee.class);
criteria.add(Restrictions.ne("fullName","Hibernate"));
List<Employee> results = criteria.list()
```

To retrieve all objects whose property matches a given pattern, we use Restrictions.like() (for case sensitivenes) and Restrictions.ilike()(for case insensitiveness)

```
Criteria criteria= session.createCriteria(InterviewBitEmployee.class);
criteria.add(Restrictions.like("fullName","Hib%",MatchMode.ANYWHERE));
List<InterviewBitEmployee> results = criteria.list();
```

Similarly, it also has other methods like isNull(), isNotNull(), gt(), ge(), lt(), le() etc for adding more varieties of restrictions. It has to be noted that for Hibernate 5 onwards, the functions returning an object of typeCriteria are deprecated. Hibernate 5 version has provided interfaces like CriteriaBuilder and CriteriaQuery to serve the purpose:

```
javax.persistence.criteria.CriteriaBuilder
javax.persistence.criteria.CriteriaQuery

// Create CriteriaBuilder
CriteriaBuilder builder = session.getCriteriaBuilder();

// Create CriteriaQuery
CriteriaQuery<YourClass> criteria = builder.createQuery(YourClass.class);
```

For introducing restrictions in CriteriaQuery, we can use the CriteriaQuery.where method which is analogous to using the WHERE clause in a JPQL query.

20. What is HQL?

Hibernate Query Language (HQL) is used as an extension of **SQL**. It is very simple, efficient, and very flexible for performing complex operations on relational databases without writing complicated queries. HQL is the object-oriented representation of query language, i.e instead of using table name, we make use of the class name which makes this language independent of any database.

This makes use of the Query interface provided by Hibernate. The Query object is obtained by calling the createQuery() method of the hibernate Session interface.

Following are the most commonly used methods of query interface:

- public int executeUpdate() : This method is used to run the update/delete query.
- public List list(): This method returns the result as a list.
- public Query setFirstResult(int rowNumber): This method accepts the row number as the parameter using which the record of that row number would be retrieved.
- public Query setMaxResult(int rowsCount): This method returns a maximum up to the specified rowCount while retrieving from the database.
- public Query setParameter(int position, Object value): This method sets the value to the attribute/column at a particular position. This method follows the JDBC style of the query parameter.
- public Query setParameter(String name, Object value): This method sets the value to a named query parameter.

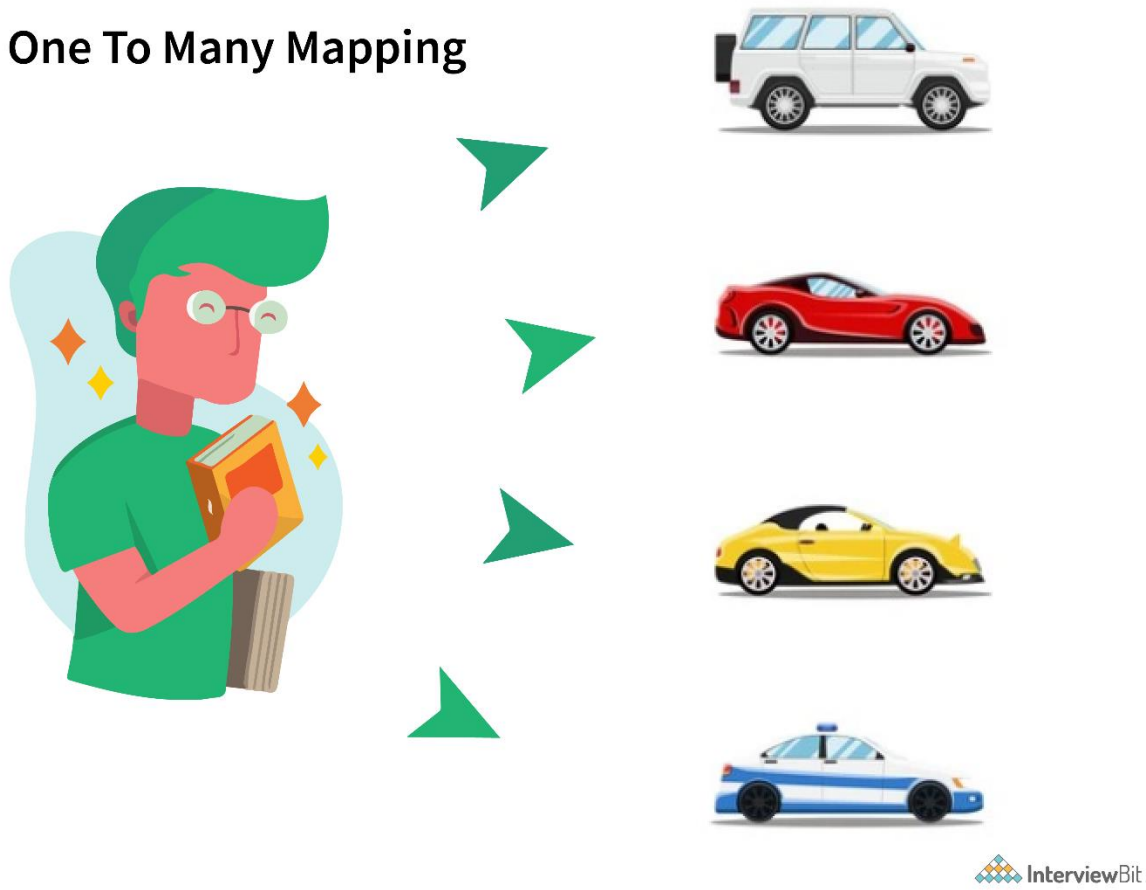
Example: To get a list of all records from InterviewBitEmployee Table:

```
Query query=session.createQuery("from InterviewBitEmployee");
List<InterviewBitEmployee> list=query.list();
System.out.println(list.get(0));
```

21. Can you tell something about one to many associations and how can we use them in Hibernate?

The **one-to-many association** is the most commonly used which indicates that one object is linked/associated with multiple objects.

For example, one person can own multiple cars.



Hibernate One To Many Mapping

In Hibernate, we can achieve this by using @OnetoMany of JPA annotations in the model classes. Consider the above example of a person having multiple cars as shown below:

```
@Entity
@Table(name="Person")
public class Person {

   //...
```

```
    @OneToMany(mappedBy="owner")
    private Set<Car> cars;

    // getters and setters
}
```

In the Person class, we have defined the car's property to have @OneToMany association. The Car class would have owned property that is used by the mappedBy variable in the Person class. The Car class is as shown below:

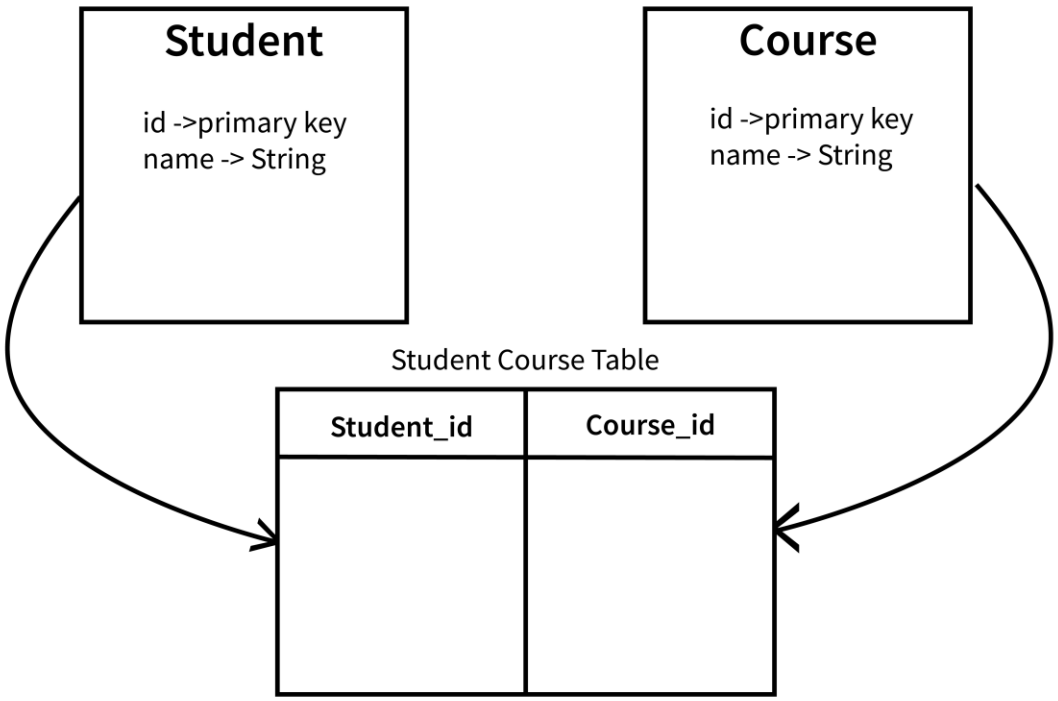
```
@Entity
@Table(name="Car")
public class Car {

    // Other Properties

    @ManyToOne
    @JoinColumn(name="person_id", nullable=false)
    private Person owner;

    public Car() {}

    // getters and setters
}
```

@ManyToOne annotation indicates that many instances of an entity are mapped to one instance of another entity – many cars of one person.

22. What are Many to Many associations?

Many-to-many association indicates that there are multiple relations between the instances of two entities. We could take the example of multiple students taking part in multiple courses and vice versa.

Since both the student and course entities refer to each other by means of foreign keys, we represent this relationship technically by creating a separate table to hold these foreign keys.



InterviewBit

Many To Many Associations

Here, Student-Course Table is called the Join Table where the student_id and course_id would form the composite primary key.

23. What does session.lock() method in hibernate do?

session.lock() method is used to reattach a detached object to the session. **session.lock()** method does not check for any data synchronization between the database and the object in the persistence context and hence this reattachment might lead to loss of data synchronization.

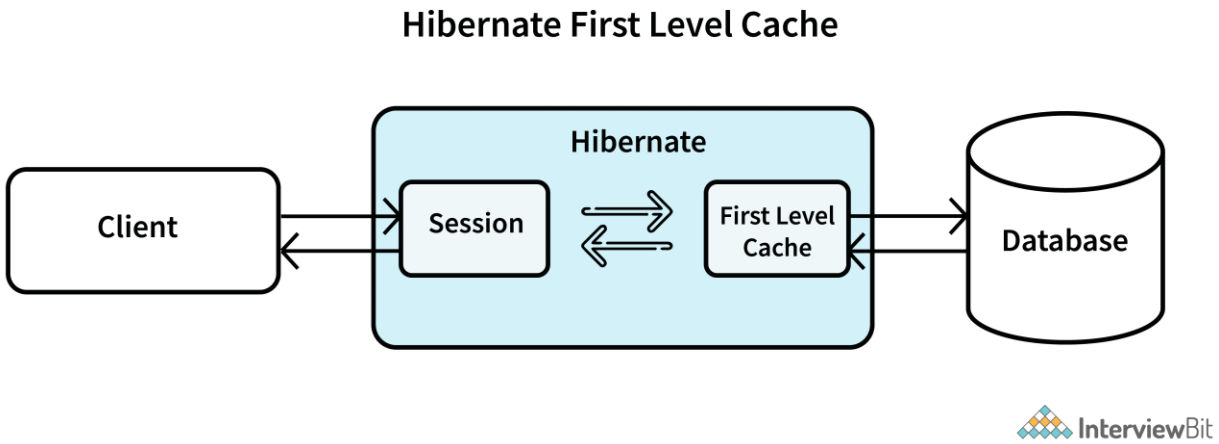
24. What is hibernate caching?

Hibernate caching is the strategy for improving the application performance by pooling objects in the cache so that the queries are executed faster. Hibernate caching is particularly useful when fetching the same data that is executed multiple times. Rather than hitting the database, we can just access the data from the cache. This results in reduced throughput time of the application.

Types of Hibernate Caching

First Level Cache:

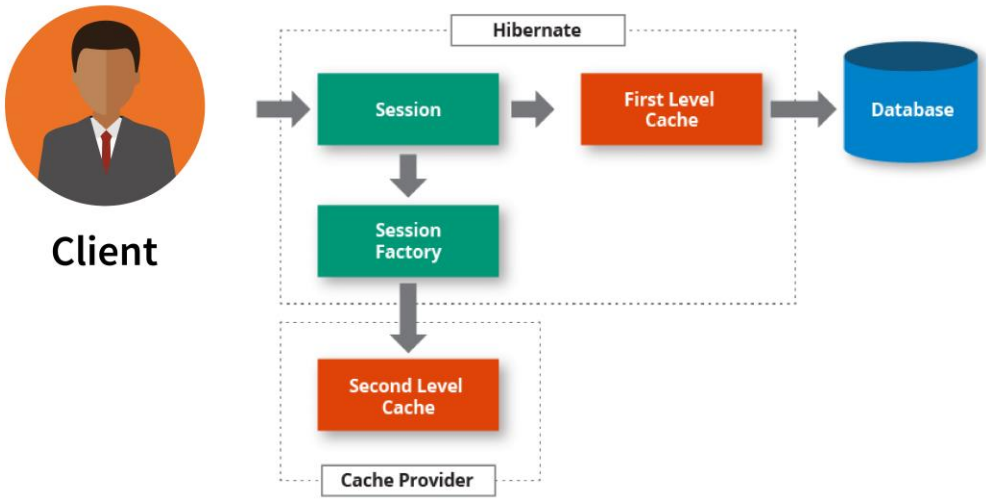
- This level is enabled by default.
- The first level cache resides in the hibernate session object.
- Since it belongs to the session object, the scope of the data stored here will not be available to the entire application as an application can make use of multiple session objects.



First Level Caching

Second Level Cache:

- Second level cache resides in the SessionFactory object and due to this, the data is accessible by the entire application.
- This is not available by default. It has to be enabled explicitly.
- EH (Easy Hibernate) Cache, Swarm Cache, OS Cache, JBoss Cache are some example cache providers.



InterviewBit

Second Level Caching

25. When is merge() method of the hibernate session useful?

Merge() method can be used for updating existing values. The specialty of this method is, once the existing values are updated, the method creates a copy from the entity object and returns it. This result object goes into the persistent context and is then tracked for any changes. The object that was initially used is not tracked.

26. Collection mapping can be done using One-to-One and Many-to-One Associations. What do you think?

False, collection mapping is possible only with **One-to-Many** and **Many-to-Many associations.**

27. Can you tell the difference between setMaxResults() and setFetchSize() of Query?

setMaxResults() the function works similar to LIMIT in SQL. Here, we set the maximum number of rows that we want to be returned. This method is implemented by all database drivers.

setFetchSize() works for optimizing how Hibernate sends the result to the caller for example: are the results buffered, are they sent in different size chunks, etc. This method is not implemented by all the database drivers.

28. Does Hibernate support Native SQL Queries?

Yes, it does. Hibernate provides the createSQLQuery() method to let a developer call the native SQL statement directly and returns a Query object.

Consider the example where you want to get employee data with the full name “Hibernate”. We don’t want to use HQL-based features, instead, we want to write our own SQL queries. In this case, the code would be:

```
Query query = session.createSQLQuery( "select * from interviewbit_employee i
be where ibe.fullName = :fullName")
                    .addEntity(InterviewBitEmployee.class)
                    .setParameter("fullName", "Hibernate"); //named parameter
s
List result = query.list();
```

Alternatively, native queries can also be supported when using NamedQueries.

Hibernate Interview Questions For Experienced

29. What happens when the no-args constructor is absent in the Entity bean?

Hibernate framework internally uses Reflection API for creating entity bean instances when get() or load() methods are called. The method Class.newInstance() is used which requires a no-args constructor to be present. When we don't have this constructor in the entity beans, then hibernate fails to instantiate the bean and hence it throws HibernateException.

30. Can we declare the Entity class final?

No, we should not define the entity class final because hibernate uses proxy classes and objects for lazy loading of data and hits the database only when it is absolutely needed. This is achieved by extending the entity bean. If the entity class (or bean) is made final, then it cant be extended and hence lazy loading can not be supported.

31. What are the states of a persistent entity?

A persistent entity can exist in any of the following states:

Transient:

- This state is the initial state of any entity object.
- Once the instance of the entity class is created, then the object is said to have entered a transient state. These objects exist in heap memory.
- In this state, the object is not linked to any session. Hence, it is not related to any database due to which any changes in the data object don't affect the data in the database.

```
InterviewBitEmployee employee=new InterviewBitEmployee(); //The object is in
the transient state.
   employee.setId(101);
   employee.setFullName("Hibernate");
   employee.setEmail("hibernate@interviewbit.com");
```

Persistent:

- This state is entered whenever the object is linked or associated with the session.
- An object is said to be in a persistence state whenever we save or persist an object in the database. Each object corresponds to the row in the database table. Any modifications to the data in this state cause changes in the record in the database.

Following methods can be used upon the persistence object:

```
session.save(record);
session.persist(record);
session.update(record);
session.saveOrUpdate(record);
session.lock(record);
session.merge(record);
```

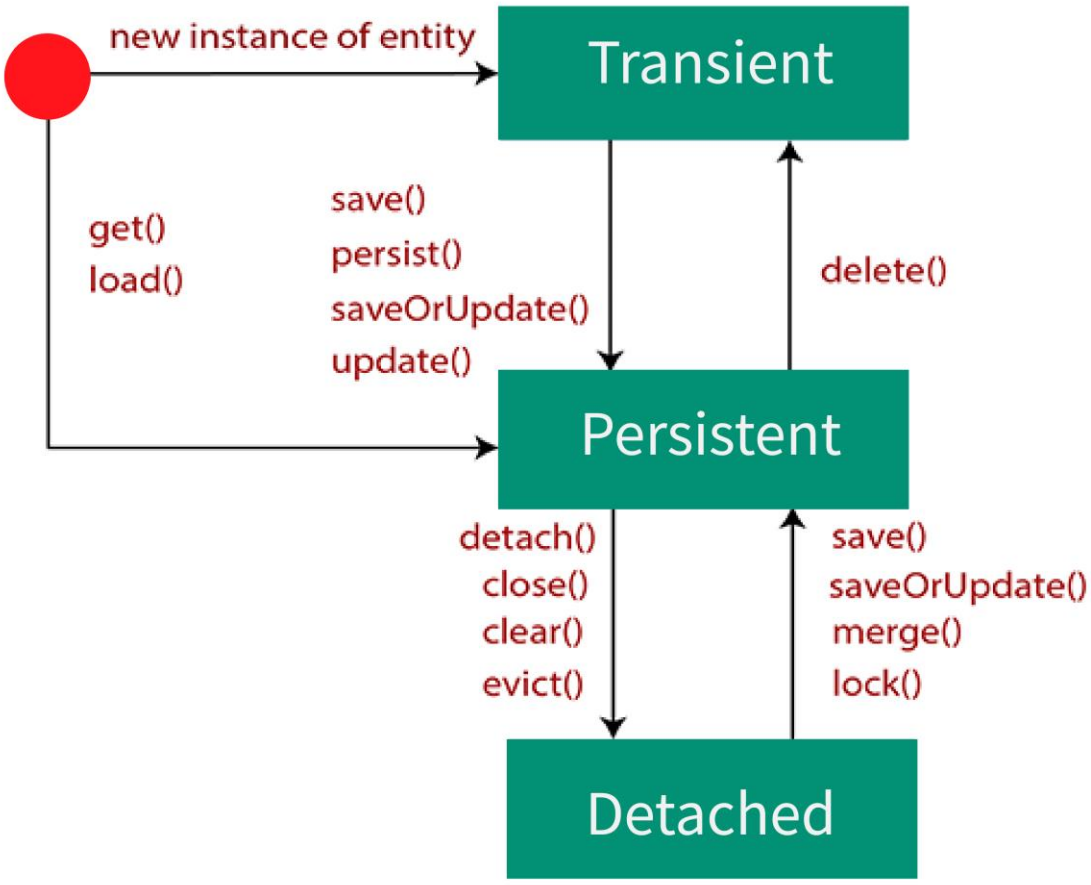
Detached:

- The object enters this state whenever the session is closed or the cache is cleared.
- Due to the object being no longer part of the session, any changes in the object will not reflect in the corresponding row of the database. However, it would still have its representation in the database.
- In case the developer wants to persist changes of this object, it has to be reattached to the hibernate session.

- In order to achieve the reattachment, we can use the methods load(), merge(), refresh(), update(), or save() methods on a new session by using the reference of the detached object.

The object enters this state whenever any of the following methods are called:

```
session.close();
session.clear();
session.detach(record);
session.evict(record);
```



InterviewBit

Persistent Entity

32. Explain Query Cache

Hibernate framework provides an optional feature called cache region for the queries' resultset. Additional configurations have to be done in code in order to enable this.

The query cache is useful for those queries which are most frequently called with the same parameters. This increases the speed of the data retrieval and greatly improves performance for commonly repetitive queries.

This does not cache the state of actual entities in the result set but it only stores the identifier values and results of the value type. Hence, query cache should be always used in association with second-level cache.

Configuration:

In the hibernate configuration XML file, set the use_query_cache property to true as shown below:

```
<property name="hibernate.cache.use_query_cache">true</property>

In the code, we need to do the below changes for the query object:
Query query = session.createQuery("from InterviewBitEmployee");
query.setCacheable(true);
query.setCacheRegion("IB_EMP");
```

33. Can you tell something about the N+1 SELECT problem in Hibernate?

N+1 SELECT problem is due to the result of using lazy loading and on-demand fetching strategy. Let's take an example. If you have an N items list and each item from the list has a dependency on a collection of another object, say bid. In order to find the highest bid for each item while using the lazy loading strategy, hibernate has to first fire 1 query to load all items and then subsequently fire N queries to load big of each item. Hence, hibernate actually ends up executing N+1 queries.

34. How to solve N+1 SELECT problem in Hibernate?

Some of the strategies followed for solving the N+1 SELECT problem are:

- Pre-fetch the records in batches which helps us to reduce the problem of N+1 to (N/K) + 1 where K refers to the size of the batch.
- Subselect the fetching strategy
- As last resort, try to avoid or disable lazy loading altogether.

35. What are the concurrency strategies available in hibernate?

Concurrency strategies are the mediators responsible for storing and retrieving items from the cache. While enabling second-level cache, it is the responsibility of the

developer to provide what strategy is to be implemented to decide for each persistent class and collection.

Following are the concurrency strategies that are used:

- **Transactional:** This is used in cases of updating data that most likely causes stale data and this prevention is most critical to the application.
- **Read-Only:** This is used when we don't want the data to be modified and can be used for reference data only.
- **Read-Write:** Here, data is mostly read and is used when the prevention of stale data is of critical importance.
- **Non-strict-Read-Write:** Using this strategy will ensure that there wouldn't be any consistency between the database and cache. This strategy can be used when the data can be modified and stale data is not of critical concern.

36. What is Single Table Strategy?

Single Table Strategy is a hibernate's strategy for performing inheritance mapping. This strategy is considered to be the best among all the other existing ones. Here, the inheritance data hierarchy is stored in the single table by making use of a discriminator column which determines to what class the record belongs.

For the example defined in the Hibernate Inheritance Mapping question above, if we follow this single table strategy, then all the permanent and contract employees' details are stored in only one table called InterviewBitEmployee in the database and the employees would be differentiated by making use of discriminator column named employee_type.

Hibernate provides @Inheritance annotation which takes strategy as the parameter. This is used for defining what strategy we would be using. By giving them value, InheritanceType.SINGLE_TABLE signifies that we are using a single table strategy for mapping.

- @DiscriminatorColumn is used for specifying what is the discriminator column of the table in the database corresponding to the entity.
- @DiscriminatorValue is used for specifying what value differentiates the records of two types.

The code snippet would be like this:

InterviewBitEmployee class:

```
@Entity
@Table(name = "InterviewBitEmployee")
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name = "employee_type")
@NoArgsConstructor
@AllArgsConstructor
public class InterviewBitEmployee {
   @Id
   @Column(name = "employee_id")
   private String employeeId;
   private String fullName;
   private String email;
}
```

InterviewBitContractEmployee class:

```
@Entity
@DiscriminatorValue("contract")
@NoArgsConstructor
@AllArgsConstructor
public class InterviewBitContractEmployee extends InterviewBitEmployee {
   private LocalDate contractStartDate;
   private LocalDate contractEndDate;
   private String agencyName;
}
```

InterviewBitPermanentEmployee class:

```
@Entity
@DiscriminatorValue("permanent")
@NoArgsConstructor
@AllArgsConstructor
public class InterviewBitPermanentEmployee extends InterviewBitEmployee {
   private LocalDate workStartDate;
   private int numberOfLeaves;
}
```

37. Can you tell something about Table Per Class Strategy.

Table Per Class Strategy is another type of inheritance mapping strategy where each class in the hierarchy has a corresponding mapping database table. For example, the InterviewBitContractEmployee class details are stored in the interviewbit_contract_employee table and InterviewBitPermanentEmployee class details are stored in interviewbit_permanent_employee tables respectively. As the data is stored in different tables, there will be no need for a discriminator column as done in a single table strategy.

Hibernate provides @Inheritance annotation which takes strategy as the parameter. This is used for defining what strategy we would be using. By giving them value,

InheritanceType.TABLE_PER_CLASS, it signifies that we are using a table per class strategy for mapping.

The code snippet will be as shown below:

InterviewBitEmployee class:

```
@Entity(name = "interviewbit_employee")
@Inheritance(strategy = InheritanceType.TABLE_PER_CLASS)
@NoArgsConstructor
@AllArgsConstructor
public class InterviewBitEmployee {
   @Id
   @Column(name = "employee_id")
   private String employeeId;
   private String fullName;
   private String email;
}
```

InterviewBitContractEmployee class:

```
@Entity(name = "interviewbit_contract_employee")
@Table(name = "interviewbit_contract_employee")
@NoArgsConstructor
@AllArgsConstructor
public class InterviewBitContractEmployee extends InterviewBitEmployee {
   private LocalDate contractStartDate;
   private LocalDate contractEndDate;
   private String agencyName;
}
```

InterviewBitPermanentEmployee class:

```
@Entity(name = "interviewbit_permanent_employee")
@Table(name = "interviewbit_permanent_employee")
@NoArgsConstructor
@AllArgsConstructor
public class InterviewBitPermanentEmployee extends InterviewBitEmployee {
   private LocalDate workStartDate;
   private int numberOfLeaves;
}
```

Disadvantages:

- This type of strategy offers less performance due to the need for additional joins to get the data.
- This strategy is not supported by all JPA providers.
- Ordering is tricky in some cases since it is done based on a class and later by the ordering criteria.

38. Can you tell something about Named SQL Query

A named SQL query is an expression represented in the form of a table. Here, SQL expressions to select/retrieve rows and columns from one or more tables in one or more databases can be specified. This is like using aliases to the queries.

In hibernate, we can make use of @NameQueries and @NameQuery annotations.

- @NameQueries annotation is used for defining multiple named queries.
- @NameQuery annotation is used for defining a single named query.

Code Snippet: We can define Named Query as shown below

```
@NamedQueries(
   {
        @NamedQuery(
        name = "findIBEmployeeByFullName",
        query = "from InterviewBitEmployee e where e.fullName = :fullName"
        )
   }
)
```

:fullName refers to the parameter that is programmer defined and can be set using the query.setParameter method while using the named query.

Usage:

```
TypedQuery query = session.getNamedQuery("findIBEmployeeByFullName");
query.setParameter("fullName","Hibernate");
List<InterviewBitEmployee> ibEmployees = query.getResultList();
```

The getNamedQuery method takes the name of the named query and returns the query instance.

39. What are the benefits of NamedQuery?

In order to understand the benefits of NamedQuery, let's first understand the disadvantage of HQL and SQL. The main disadvantage of having HQL and SQL scattered across data access objects is that it makes the code unreadable. Hence, as good practice, it is recommended to group all HQL and SQL codes in one place and use only their reference in the actual data access code. In order to achieve this, Hibernate gives us named queries.

A named query is a statically defined query with a predefined unchangeable query string. They are validated when the session factory is created, thus making the application fail fast in case of an error.

Thymeleaf Interview Question

What is Thymeleaf?

Thymeleaf can bring natural templates to our development workflow, it is a java based library that is used in creating web application.It is a server side mordern template engine that is used for development of web and standalone environments.
Thymeleaf provides support for serving XHTML/HTML5 in web applications.It converts our files into well formed XML files.It contains 6 types of templates given below:
XML
Valid XML
XHTML
Valid XHTML
HTML5
Legacy HTML5

Name some companies using Thymeleaf?

Companies that uses Thymeleaf are:
Enerko Informatik
Enonic
Lagerwey
PPI AG
Broadleaf Commerce
Auchan
Apereo CAS
Connect Group
Trabe

What is Thymeleaf Template?

Thymeleaf Template are locates in the custom directory as it is a Maven build file.It can directly open browsers and can display correct web pages.
Templates processed by the thymeleaf are HTML, XML, TEXT, etc.

How to check if list is empty using thymeleaf?

We can check if the list is empty by using the following code:

```
<div th:if="false">
 --content--
</div>
```

What is Thymeleaf fragment?

Thymeleaf fragment helps us in importing fragments of a template into another template.It is a small piece of code that can be included in another template.
It is used to create reusable, small components like the header, footer, navigation menu and other parts of a website that are repeated used on multiple pages.

What is a dialect in Thymeleaf?

By using dialect we can build layouts and resuable templates to improve the code reise.Thymeleaf dialect is good in Processing logic, Preprocessing & Postprocessing logic, and Expression objects.
There are 2 types of Thymeleaf Dialect are:
Custom dialects - By using this dialect we can build layouts.
Layout Dialects - It uses decorator pattern for working the layout files.

How to loop through Map in Thymeleaf?

We can loop through Map in Thymeleaf, by using the folowing code:

```
<tr th:each="instance : ">
    <td th:text="${instance.key}">keyvalue</td>
    <td th:text="${instance.value.numOfData}">num</td>
</tr>
```

What is Template Resolver in Thymeleaf?

A template resolver helps in resolving templates into TemplateResolution objects which contains information like template mode, caching, prefix and suffix of templates.It is reponsible for loading templates from a specific location.