| 1. **Which java version have you used in your project?** 2. **Where did you use stream api?** 3. **What is the advantage of lambda expression?** 4. **What is the need for a Spring framework?** 5. **What is the difference between spring and spring boot?** 6. **What is the advantage of web services?** 7. **Where did you use lambda expressions in your project?** 8. **What is the difference between 1.7 and 1.8?** 9. **difference between == and equals() w.r.t String class?** 10. **Which collections have you used in the project? Explain internal mechanics of Hashmap?** 11. **Difference between HashTable and HashMap** 12. **How to make a hashmap thread safe?** 13. **What is the difference between Synchronized hashmap and concurrent hashmap?** 14. **how to make arrayList thread safe?** 15. **Does java support multiple inheritance?** 16. **if there are three interfaces with similar abstract methods in all of them, what happens? Will I get ambiguity?** 17. **Can there be similar hashcodes? What happens during that time with a hashmap?** 18. **Important methods in Object class and String class?** 19. **How did you implement your own hashCode() method, what logic you wrote inside that?** 20. **How does Set internally detect duplicate elements?** 21. **Best practices for using Collection framework.** 22. **What is the Concurrent modification exception?** 23. **What design pattern is used for implementing String Class?** 24. **Deep cloning vs shallow cloning** 25. **Difference between Error And Exception** 26. **Difference between Unchecked exception and Error** 27. **How do you define your own exception?** 28. **What is static import?** 29. **What is Reflection?** 30. **Class vs object** 31. **how to remove duplicates from arraylist?** 32. **diff between interface and abstract class?** 33. **diff between interface and abstract class w.r.t Java 8?** 34. **difference between checked and unchecked exceptions?** 35. **diff between overloading and overriding** 36. **Why is stringbuilder used over string for concatenation of strings?** 37. **Why is the key in hashmap immutable ?** 38. **What are immutable classes? How can you make them immutable?** 39. **If an arraylist is final in an immutable class and you have initialized it; can you modify it later somewhere in the same class?** 40. **What are the runtime expectations that you faced in your project?** 41. **What is a list?** 42. **String concatenation program?** 43. **Why is string immutable?** 44. **How to compare objects in 2 array List?** 45. **How to create a singleton class in java?** 46. **What is Object-oriented programming?** 47. **What is abstraction?** 48. **What are the object-oriented principles?** 49. **How do you create a thread?** 50. **Difference between Runnable and Callable?** 51. **Explain Serialization in java.** 52. **What is a Transient keyword?** 53. **Explain Life Cycle of thread.** 54. **What is the use of synchronization?** 55. **What is the use of synchronization?** 56. **Difference between abstract class and interface?** 57. **When do you use collections?** 58. **What is the advantage of Collection?** 59. **59. Can we override the main method?** 60. **Difference between final, finally and finalize?** 61. **What is a runtime exception?** 62. **Difference between string buffer and string builder?** 63. **Difference between string buffer and string?** 64. **Difference between Hashmap and Hashtable?** 65. **How do you add values and keys in a hashmap?** 66. **What are the best practices for memory management?** 67. **What is a garbage collector?** 68. **What is a synchronized keyword?** 69. **What are cursors?** 70. **What is aggregation?** 71. **What happens when we add a custom object to the hashmap as a key? Will it add a duplicate object?** 72. **Why do we need to override equals and hashcode methods when we are trying to add custom objects into the hashmap or hashset?** 73. **What is the Serialization? Why do we need it?** 74. **How can we stop instance variables from serialization?** 75. **will gc method destroy an unreferenced object?. Can you ensure that it will destroy the unused object** 76. **Two objects referring to each other, not any other references. will garbage collector free the memory out of memory exception. What should you be carefull** 77. **How multithreading is useful in a single core cpu.** 78. **If there is a return statement in the catch block, will finally block execution?** 79. **If 3 threads are there T1, T2 and T3, we want T2 to execute after T1 and T3 to execute after T2, how to achieve this?** 80. **Can we use $\_ as a variable name?** 81. **syso('j'+'a'+'v'+'a'), what will be the output of this statement?** 82. **How to increase bucket size of hashmap** 83. **What is volatile** 84. **What is static** 85. **Difference between volatile and static** 86. **Object level lock vs class level lock** 87. **What is bucket level lock and what are it’s advantages** 88. **What is the Join method** 89. **What is Functional interface** 90. **What is string manipulation** 91. **write a program to find duplicate in 2 strings** 92. **If we have a long method that you want to run. how would you run it using Runnable? In the instance you want to stop the thread at any given moment and get the results how would you implement your thread?** 93. **What implementation TreeSet uses for sorting? (Uses Merge sort internally and self balances using Red-Black Tree?)** 94. **Is there any way to block/stop the execution of the finally block?** 95. **What do you understand about parallel aggregate operation in stream api?** 96. **Explain Consumer, Supplier and Predicate interface?** 97. **How the interface is better over abstract class. I.e interface over implementation** 98. **Advantages of HAS-A or composition over inheritance or IS-A** 99. **Difference between StringBuffer and StringBuilder?** 100. **What are Wrapper Classes?** 101. **add 2 integer using lambda** 102. **wait vs join** 103. **Find non repeated words in 2 sentences.** 104. **singleton design pattern implement** 105. **sleep() vs wait()** 106. **Cohesion vs Coupling?** 107. **Write a program to sort a Collection of Objects: Where the object is a Person with firstName and LastName.Sort the collection according to lastName in it’s natural order and print to console.** 108. **Write a program to display a Collection that only includes the word “java” ignoring cases separated by comma. Use lambda to solve it. 5 to 7** 109. Java 8 flatmap vs map 110. Explain Function interface 111. What is the lambda expression? 112. How are lambda expressions and functional interfaces related? 113. How to calculate hashcode? 114. Can we add objects as keys in a hashmap? 115. What are the default methods? 116. What is volatile? 117. What is the difference between Synchronized hashmap and concurrent hashmap? |
| --- |

| **Which java version have you used in your project?**  I have used java 8 features such as lambda expression,functional interfaces and some what i used stream api. |
| --- |
| **Where did you use stream api?**  Basically stream api gives us a very compact way of coding. I used a stream api to manipulate the collection. At one point, I had a list of employee objects from the database but we needed such objects of the employee which has enabled status. So I used stream api. Mostly I used stream api to manipulate collection. |
| **What is the advantage of lambda expression?**  Lambda expression allows us to write compact code. We can reduce code length and readability will be increased. |
| **What is the need for a Spring framework?**  Basically any framework helps us do faster coding. For example we have a list of values and if we use a normal array, to sort the array we need to write sorting logic. But if we use any collection, then we have a sort method. So this way we could do faster development. Spring framework gave us dependency injection,AOP… we help us to write clean and faster code. |
| **What is the difference between spring and spring boot?**  Basically boot is not developed from scratch. It is developed on top of the spring framework, that is why a lot of things are the same. Only difference is that using  microservices and cloud made it easy to boot. |
| **What is the advantage of web services?**  Web service is all about data communication. It allows us to expose our code to a network which can be accessed by http requests.  Most important thing is Web Service can be located on the same computer within the same network or different computer with different network. Web service is protocol independent, language independent and platform independent. Web services support standard protocol and data format like HTTP, XML, and SOAP. |
| **Where did you use lambda expressions in your project?**  Basically I used predicate,Function,Supplier and Consumer interfaces, there I used lambda expressions. Also while using stream api, I used lambda expressions. |
| **what is the difference between 1.7 and 1.8:**  Java 7:   1. Try with resources The try-with-resources statement is a try statement that declares one or more resources. A *resource* is an object that must be closed after the program is finished with it. 2. Multi-catch block 3. Allows us to use underscore(\_) in numeric value 4. Type Inference for Generic Instance Creation i.e before java 7, we write List<String> list = new ArrayList<String>(); but in java 7, we can write List<String> list = new ArrayList<>(); we can skip this.   Java 8:   1. Functional interface 2. Lambda expression 3. Static method and default method inside interface 4. Stream api 5. Joda date and time api |
| **Difference between == and equals() w.r.t String class?c**  Basically == is an operator which compares references of two reference variables and equals method is present in object class which is meant for reference comparison only. But we can override it to compare data, just like String class. So if we use the equals method to compare two String objects then it compares data. |
| **comparator and comparable?**  A comparable object is capable of comparing itself with another object. The class itself must implement the java.lang.Comparable interface to compare its instances.  Unlike Comparable, Comparator is external to the element type we are comparing. It’s a separate class. We create multiple separate classes (that implement Comparator) to compare by different members.   * Comparable is meant for objects with natural ordering which means the object itself must know how it is to be ordered. For example, the number of students. Whereas, Comparator interface sorting is done through a separate class. * Logically, Comparable interface compares “this” reference with the object specified and Comparator in Java compares two different class objects provided. * If any class implements Comparable interface in Java then collection of that object either List or Array can be sorted automatically by using Collections.sort() or Arrays.sort() method and objects will be sorted based on their natural order defined by CompareTo method.   For More clearity:<https://www.geeksforgeeks.org/comparable-vs-comparator-in-java/> |
| **Which collections have you used in the project? Explain internal mechanics of Hashmap?**  Ya I used several Collection classes in my project like ArrayList,HashSet. Ya of course i have used Hashmap too.  So basically we use hashmap when we want to store data in the form of key value format. When we create an object of a hashmap, it allocates a hashtable with 16 buckets for us.  When we try to add key-value pairs, the first thing both will add as an object. It will call the hashcode method on the key object which will give us a hashcode for that key. Then returned hashcode % 16 that is the number of buckets will give us an index where key value pairs will get stored. But in that bucket there may be a chance of duplicate key, to check duplicate key it will call equals method. If the equals method returns true, it will understand this is a duplicate key so it will not add. If equals returns false then it will add that key value pair in that bucket.  In the case of a hash collision, internally it will allocate a linkedlist. |
| **Difference between HashTable and HashMap**  1. HashMap is non synchronized. It is not-thread safe and can’t be shared between many threads without proper synchronization code whereas Hashtable is synchronized. It is thread-safe and can be shared with many threads.  2. HashMap allows one null key and multiple null values whereas Hashtable doesn’t allow any null key or value.  3. HashMap is generally preferred over HashTable if thread synchronization is not needed |
| **How to make a hashmap thread safe?**  We have a Collections class which has a synchronized map method which gives us a synchronized version of map.  HashMap map;  Collections.synchronizedMap(map); |
| **What is the difference between Synchronized hashmap and concurrent hashmap?**  In the case of Synchronized hashmap, the whole object will get locked whereas a concurrent hashmap has a bucket level lock concept.   1. You should use ConcurrentHashMap when you need very high concurrency in your project. 2. It is thread safe without synchronizing the whole map. 3. Reads can happen very fast while writing is done with a lock. 4. There is no locking at the object level. 5. The locking is at a much finer granularity at a hashmap bucket level. 6. ConcurrentHashMap uses multitude of locks   **how to make arrayList thread safe?**  We have a Collections class which has a synchronized list method which gives us a synchronized version of the list.  ArrayList list;  Collections.synchronizedList(list); |
| **Does java support multiple inheritance?**  NO. because of ambiguity. If two parent classes are having the same method or variable then the child class object will get confused from which parent he should access. But we can implement more than one interface. |
| **If there are three interfaces with similar abstract methods in all of them, what happens? Will I get ambiguity?**  No. Because while calling that method, it will always get calls from child class or it will always call child class implemented methods. |
| **Can there be similar hashcodes? What happens during that time with a hashmap?**  Yes. If two string objects are having the same text, in that case it will give us the same hashcode. Or if we have overridden hashcode methods  in your custom classes then there may be chances of duplicate hash codes. In this case hashmap will use a linked list internally. |
| **Important methods in Object class and String class?**  Basically there are 12 methods in object class such as: toString, hashcode, equals, clone,finalize, three wait and two notify methods. And the last one is registerNative which is JVM use. |
| **How did you implement your own hashCode() method, what logic you wrote inside that?**  I overridden the hashcode method when I tried to add a custom object in the hashmap as key. There I override the hashcode method in such a way, if two objects are having the same data then it should generate the same hashcode. |
| **How does Set internally detect duplicate elements?**  Basically the hashset does not allow duplication. Basically , a Set achieves uniqueness internally through HashMap. For more clarity use following link:  <https://www.geeksforgeeks.org/internal-working-of-sethashset-in-java/> |
| **Best practices for using Collection framework.**   1. Choosing the right collection map.  Always using interface type when declaring a collection  1. Use generic type and diamond operator. 2. Prefer isEmpty() over size() 3. Return empty collections or arrays, not null. 4. Do not use the classic for loop. 5. Favor using forEach() with Lambda expressions. |
| **What is the Concurrent modification exception?**  java.util.ConcurrentModificationException is a very common exception when working with Java collection classes. Java Collection classes are fail-fast, which means if the Collection will be changed while some thread is traversing over it using iterator, the iterator.next() will throw it. |
| **What design pattern is used for implementing String Class?**  Flyweight Design Pattern.  For more info:  <https://www.journaldev.com/1562/flyweight-design-pattern-java> |
| **Deep cloning vs shallow cloning wdw a**  Shallow cloning: it creates a clone of only the outer object that is in case of Employee and Address, it will clone only the Employee object not the Address object.  Deep Cloning: It creates a clone of both Outer as well as Inner Object. That is the Employee and Address object.  Object class clone method is meant for performing shallow cloning. If we want deep cloning then override the clone waaaaawda method and write our own logic. |
| **Difference between Error And Exception**  Exception:  Exceptions are those which can be handled at the run time whereas errors cannot be handled.  An exception is an Object of a type deriving from the System.Exception class. SystemException is thrown by the CLR (Common Language Runtime) when errors occur that are nonfatal and recoverable by user programs. It is meant to give you an opportunity to do something with a throw statement to transfer control to a catch clause in a try block.  Error  An Error is something that most of the time you cannot handle. Errors are unchecked exceptions and the developer is not required to do anything with these. Errors normally tend to signal the end of your program, it typically cannot be recovered from and should cause you exit from current program. It should not be caught or handled.  All the Errors are Exceptions but the reverse is not true. In general Errors are which nobody can control or guess when it happened, on the other hand Exception can be guessed and can be handled. |
| **Difference between Unchecked exception and Error:**  Error  An Error is a subclass of Throwable that indicates serious problems that a reasonable application should not try to catch. Most such errors are abnormal conditions. The ThreadDeath error, though a "normal" condition, is also a subclass of Error because most applications should not try to catch it.  Unchecked exception  The class Exception and its subclasses are a form of Throwable that indicates conditions that a reasonable application might want to catch. |
| **How do you define your own exception?**  We can define custom exceptions by extending Throwable class. If we want a checked exception then extend Exception and if we want to define unchecked exception then extend RuntimeException class. |
| **What is static import?**  In Java, the static import concept is introduced in the 1.5 version. With the help of static import, we can access the static members of a class directly without class name or any object.  import static java.lang.System.\*; //Using Static Import  public class StaticImportDemo {  public static void main(String args[]) {  //System.out is not used as it is imported using the keyword static.  out.println("Welcome to Tutorials Point");  }  } |
| **What is Reflection?**  Reflection is an API which is used to examine or modify the behavior of methods, classes, interfaces at runtime.  The required classes for reflection are provided under java.lang.reflect package.  Reflection gives us information about the class to which an object belongs and also the methods of that class which can be executed by using the object.  Through reflection we can invoke methods at runtime irrespective of the access specifiers used with them. |
| **Diff between object and class**  Class is a blueprint of a real time entity. Basically class is a keyword which is used to create user defined data type.  Where as an object is a physical existence of a class. When we create objects, memory gets allocated to instance variables of that class. |
| **how to remove duplicates from arraylist?**  We can do this with different ways like   1. With the help of for each loop  | public class Demo {  public static void main(String[] args) throws Exception {  ArrayList<Integer> al = new ArrayList<Integer>();  al.add(10);  al.add(8);  al.add(9);  al.add(10);  al.add(9);  al.add(7);  removeDup(al);  }  private static void removeDup(ArrayList<Integer> al) {  ArrayList<Integer> uniqueList = new ArrayList<Integer>();  for (Integer i : al) {  if(!uniqueList.contains(i)){  uniqueList.add(i);  }  }  System.out.println(uniqueList);  }  } | | --- |  1. With the help of LinkedHashSet: A better way (both time complexity and ease of implementation wise) is to remove duplicates from an ArrayList is to convert it into a Set that does not allow duplicates. Hence LinkedHashSet is the best option available as this do not allow duplicates as well it preserves the insertion order.  | public static void main(String[] args) throws Exception {  ArrayList<Integer> al = new ArrayList<Integer>();  al.add(10);  al.add(8);  al.add(9);  al.add(10);  al.add(9);  al.add(7);  Set<Integer> set = new LinkedHashSet<Integer>(al);  al.clear();  al.addAll(set);  System.out.println(al);  } | | --- |  1. Using java 8 stream api:  | public static void main(String[] args) throws Exception {  ArrayList<Integer> al = new ArrayList<Integer>();  al.add(10);  al.add(8);  al.add(9);  al.add(10);  al.add(9);  al.add(7);  List<Integer> uniqueList = al.stream().distinct().collect(Collectors.toList());  System.out.println(uniqueList);  } | | --- | |
| **Diff between interface and abstract class?**  Abstract class:   1. Abstract class can have abstract and non-abstract methods. 2. Abstract class doesn't support multiple inheritance. 3. Abstract class can have final, non-final, static and non-static variables. 4. Abstract class can provide an implementation of interface. 5. The abstract keyword is used to declare abstract class. 6. An abstract class can extend another Java class and implement multiple Java interfaces. 7. An abstract class can be extended using the keyword "extends". 8. A Java abstract class can have class members like private, protected, etc.   Interface:   1. Interface can have only abstract methods. Since Java 8, it can have default and static methods also. 2. Interface supports multiple inheritance. 3. Interface has only static and final variables. 4. Interface can't provide the implementation of abstract class. 5. The interface keyword is used to declare the interface. 6. An interface can extend another Java interface only. 7. An interface can be implemented using keyword "implements". 8. All members are public by default. |
| **Diff between interface and abstract class w.r.t java 8?**  According to Java 8,   1. Abstract class can not have a default method where as interface can have. 2. Abstract classes have constructor. Interface not. 3. Can not extend more than one abstract class but interface can. 4. Abstract class can have non static variables where as interface can not. |
| **Difference between checked and unchecked exception?**  Checked: are the exceptions that are checked at compile time. If some code within a method throws a checked exception, then the method must either handle the exception or it must specify the exception using *throws* keyword.  Unchecked: are the exceptions that are not checked at compile time. In C++, all exceptions are unchecked, so it is not forced by the compiler to either handle or specify the exception. It is up to the programmer to be civilized, and specify or catch the exceptions. |
| **Diff between overloading and overriding**   | Overloading | Overriding | | --- | --- | | Used to increase readability of code | Used to extend existing functionality of parent class | | Method signature must be different | Method signature must be same | | It is called as compile time polymorphism or static polymorphism | It is called as runtime polymorphism or dynamic polymorphism | | Don’t bother about return type | Return type should be the same or co variant. | | Access specifiers should be same of different | Access specifiers should be the same or in increasing order. | |
| **Why is stringbuilder used over string for concatenation of strings?**  Using stringBuilder.append("") takes less than a second for the same query with s = s+ “”. So the difference is huge. Inside a loop StringBuilder is much faster. As the string length increases, so does the concatenation time.  Another thing is whenever you try to modify string objects, it creates a new object with modification. So when you try to concatenate string it will create a new object which is not required.  So whenever we want to perform more modification operations on string, we should use stringBuilder over string. |
| **Why is the key in hashmap immutable ?**  If immutable, the object's hashcode wont change and it allows caching the hashcode of different keys which makes the overall retrieval process very fast.  Also for mutable objects ,the hashCode() might be dependent on fields that could change, if this happens you won’t be able to find the key (and its value) in the HashMap since hashCode() returns different values. |
| **What are immutable classes? How can you make them immutable?**  Immutable class means that once an object is created, we cannot change its content. In Java, all the wrapper classes (like String, Boolean, Byte, Short) and String class are immutable.  We can create our own immutable class as well.  Following are the requirements:   1. Class must be declared as final (So that child classes can’t be created) 2. Data members in the class must be declared as final (So that we can’t change the value of it after object creation) 3. A parameterized constructor 4. Getter method for all the variables in it 5. No setters(To not have option to change the value of the instance variable) |
| **If an arraylist is final in an immutable class and you have initialized it; can you modify it later somewhere in the same class?**  I still can add new elements to ArrayList , remove elements and update it. It just means that you can't re-assign its reference. |
| **What are the runtime expectations that you faced in your project?**  Mostly, I faced NUllPointerException. When we expect an object to get but sometimes we don’t. In that case we got a NullPointerException. |
| **What is a list?**  List is Collection interface which allows duplicate data and preserves the insertion order. ArrayList,LinkedList,Vector are the implementation classes of List. |
| **String concatenation program?**   | public class JavaConcat {  public static void main(String[] args){  String a = "w3schools";  String b = ".in";  String c = a.concat(b);  System.out.println(c);  }  } | | --- | |
| **Why is string immutable?**   1. 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 "Test" to "TEST", all other clients will also see that value. 2. Since String is immutable, it is safe for multithreading. A single String instance can be shared across different threads. This avoids the use of synchronization for thread safety. Strings are implicitly thread-safe. 3. Since String is immutable, its hashcode is cached at the time of creation and it doesn’t need to be calculated again. This makes it a great candidate for the key in a Map and its processing is faster than other HashMap key objects. This is why String is the most widely used HashMap keys. |
| **How to compare objects in 2 array List?**  ArrayList manipulation: please follow given link  <https://howtodoinjava.com/java/collections/arraylist/compare-two-arraylists/> |
| **How to create a 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 private, so that you can restrict the creation of the object. Provide a static method to get instance of the object, wherein you can handle the object creation inside the class only. |
| **What is Object-oriented programming**  Object-oriented programming (OOP) is a way of writing computer programs using the idea of "objects" to represent data and methods. ... Also, because of the way object-oriented programming is designed, it helps the developer by allowing for code to be easily reused by other parts of the program or even by other people. |
| **What is abstraction?**  Abstraction is hiding internal implementation and provides only services to users. For example when we call the write method of FileWriter class, it writes given data into the respective file. We don’t care how it is written internally on disc and all. |
| **What are the object-oriented principles?**   1. Abstraction is the process of exposing the essential details of an entity, while ignoring the irrelevant details, to reduce the complexity for the users. 2. Encapsulation is the process of bundling data and operations on the data together in an entity. 3. Inheritance is used to derive a new type from an existing type, thereby establishing a parent-child relationship. 4. Polymorphism lets an entity take on different meanings in different contexts. |
| **How do you create a thread?**  We can create a thread by using three ways:   1. by extending Thread class 2. By implementing Runnable interface 3. By implementing Callable interface   Best way is to use either Runnable or Callable. |
| **Difference between Runnable and Callable?**   1. Runnable is available in Java right from the beginning i.e. JDK 1.0 while Callable was later added to Java 5, so you cannot use Callable before Java 5. 2. Runnable can not return value. Callable can return value. 3. Callable call() method can throw [Checked exceptions](http://www.java67.com/2012/12/difference-between-runtimeexception-and-checked-exception.html) while Runnable's run() method cannot throw checked exceptions. 4. In order to use Callable, you need to override the call() method while in order to use Runnable interface you need to override the run() method in Java. 5. There is one limitation while using Callable interface in Java that you cannot pass it to Thread as you pass the Runnable instance. There is no constructor defined in the Thread class which accepts a Callable interface. So in order to execute a Callable instance you need to use the ExecutorService interface of Java 5 Executor framework |
| **Explain Serialization in java.**  Serialization is a mechanism of converting the state of an object into a byte stream. Deserialization is the reverse process where the byte stream is used to recreate the actual Java object in memory.  The byte stream created is platform independent. So, the object serialized on one platform can be deserialized on a different platform.  The ObjectOutputStream class contains the writeObject() method for serializing an Object.  The ObjectInputStream class contains the readObject() method for deserializing an object.  Points to remember  1. If a parent class has implemented Serializable interface then child class doesn’t need to implement it but vice-versa is not true.  2. Only non-static data members are saved via the Serialization process.  3. Static data members and transient data members are not saved via Serialization process.So, if you don’t want to save value of a non-static data member then make it transient.  4. Constructor of an object is never called when an object is deserialized.  5. Associated objects must be implementing Serializable interface. |
| **What is a Transient keyword?**  Transient is a Java keyword which marks a member variable not to be serialized when it is persisted to streams of bytes. |
| **Explain Life Cycle of thread.**     1. New Thread: When a new thread is created, it is in the new state. The thread has not yet started to run when thread is in this state. When a thread lies in the new state, it’s code is yet to be run and hasn’t started to execute. 2. Runnable State: A thread that is ready to run is moved to runnable state. In this state, a thread might actually be running or it might be ready to run at any instant of time. 3. Blocked/Waiting state:When a thread is temporarily inactive, then it’s in one of the following states: 4. Blocked 5. Waiting   For example, when a thread is waiting for I/O to complete, it lies in the blocked state. It’s the responsibility of the thread scheduler to reactivate and schedule a blocked/waiting thread. A thread in this state cannot continue its execution any further until it is moved to a runnable state. Any thread in these states does not consume any CPU cycle.  4. Timed Waiting: A thread lies in a timed waiting state when it calls a method with a time out parameter. A thread lies in this state until the timeout is completed or until a notification is received. For example, when a thread calls sleep or a conditional wait, it is moved to a timed waiting state.  5.Terminated State: A thread terminates because of either of the following reasons:  Because it exists normally. This happens when the code of thread has entirely executed by the program.  Because there occurred some unusual erroneous event, like segmentation fault or an unhandled exception.  A thread that lies in a terminated state no longer consumes any cycles of CPU. |
| **How does the garbage collector work?**  All objects are allocated on the heap area managed by the JVM. ... As long as an object is being referenced, the JVM considers it alive. Once an object is no longer referenced and therefore is not reachable by the application code, the garbage collector removes it and reclaims the unused memory.  Although destroying an object is not a programmer's job. Still as good programming, we should call the garbage collector explicitly as System.gc().  Garbage collector always call the finalize method before destroying an unreferenced object. |
| **When do you use multithreading?**  Multi threading is most useful when a process consists of mutually independent tasks , either of which can be performed when the other is waiting for some resources to be freed up and allocated by the CPU.  For example , suppose a heavy process in which one of the many tasks is to read a file and store it in the local cache and other is to use the printer to print out the content. In this situation , if the printer is not available at the moment then in a single thread environment , the process will halt until the printer is freed up. However, if multi threading is being used, while one thread can wait for the printer, others can perform a read task on the file. This way the CPU cycles are used more efficiently.  More Example:   1. We can send mail using thread. 2. We can use thread to perform read/write file system which takes long time |
| **What is the use of synchronization?**  When multiple threads are working on the same object or shared object. In that case there may be a chance of data inconsistency problem if multiple threads are trying to modifying that object. |
| **When do you use collections?**  Sometimes you need to group multiple items in a single unit. Collections are used in situations where data is dynamic. Collections allow adding an element, deleting an element and host of other operations. |
| **What is the advantage of Collection?**  You can focus on your business logic development rather than implementing our own collection classes. So, it reduces your development efforts.  Collection classes are already developed, so you get proper quality classes, rather than creating your own.  Reduced effort for code maintenance by using collection classes shipped with JDK in util package. |
| **Can we override the main method?**  No. if we redefine the main method in child class, it will be method hiding not overriding. |
| **Difference between final, finally and finalize?**  **Final:**  Final is the keyword if we use it with variable, we can not modify it. If we use it with a method,we can not override it. If we use it with a class, we can not extend that class.  **Finally:**  It is a block,which we use with try catch blocks. Whatever resources we have created in try block, we will close in finally block. Finally block contains cleanup logic.  **Finalize:**  It is a method. Garbage collector will call this method before destroying unreferenced objects to perform cleanup operations on those objects. |
| **What is runtime exception?**  Exceptions that are unchecked and will raise at runtime because of logical mistake. |
| **Difference between string buffer and string builder?**  **StringBuffer is synchronized i.e. thread safe.** **It means two threads can't call the methods of StringBuffer simultaneously.** **StringBuilder is non-synchronized i.e. not thread safe**. It means two threads can call the methods of StringBuilder simultaneously. |
| **Difference between string buffer and string?** |
| **Difference between Hashmap and Hashtable?**   | HashMap | HashTable | | --- | --- | | Non legacy class | Legacy class came in 1.0 version | | Not synchronized | synchronized | | Can add null key and value | Can not add null key and value. | | Not thread safe | Thread safe | | High performance | Low performance | | Less secure in multithreaded env | More secure in multithreaded env | |
| **How do you add values and keys in a hashmap?**  We add key and values in a hashmap as an object. We use the put method to add. In the case where we are adding a custom object as a hashmap key, we have to override hashcode and equals method. |
|  |
| **What is a garbage collector?**  **Garbage collector** is a daemon thread which continuously runs in the background. It destroys unreferenced objects.  It is used to free up the heap memory by destroying unreferenced objects. Garbage collector calls finalize method before destroying an unreferenced object to perform cleanup operations. |
| **What is a synchronized keyword?**  Synchronized is a keyword which we use with method or block. When we declare a method synchronized then at a time only one thread can access that method.  It increases security but decreases performance because waiting time for threads increases parallely. |
| **What are cursors?**  To iterate over collection we use cursors. There are three cursors like Enumeration,Iterator,ListIterator.  Using cursors, we can read,remove,update collections. |
| **What is aggregation?**  Aggregation in Java is a relationship between two classes that is best described as a "has-a" and "whole/part" relationship.Example: Employee HAS-A Address. Using aggregation, we can reuse the code like suppose we have Employee,Vendor,HR classes. But every one has Address. So we can create a single Address class which we can reuse in Employee,Vendor,HR. |
| **What happens when we add a custom object to the hashmap as a key? Will it add a duplicate object?**  When we are trying to add custom objects in the hashmap as a key then it will add duplicate objects as a key. To avoid this problem we have to override equals and hashcode methods in that class. For example when we are adding an Employee object as a hashmap key then we have to override equals and hashcode methods in the Employee class. |
| **Why do we need to override equals and hashcode methods when we are trying to add custom objects into the hashmap or hashset?**  Generally, when we are trying to add an object into a hashmap, it will call hashcode method on key to find bucket to store key value and then it will call equals method on new key and already present key to check duplicate key. If the equals method returns true then it will understand it is a duplicate key and it will override the old value, otherwise it will add a key value.  If we don’t override equals method in custom classes then that equals method will compare references, so that even two objects with the same data will add it i.e duplicate key will be la  added. So to compare data we have to override equals method.  And two objects with the same data should come to the same bucket, so that it will call the equals method on it, we need to override the hashcode method such that two objects with the same data will generate the same hashcode. |
| **What is the Serialization? Why do we need it?**  Serialization refers to the translation of java object state into bytes to send it over the network or store it in hard disk. ... We need serialization because the hard disk or network infrastructure are hardware component and we cannot send java objects because it understands just bytes and not java objects |
| **How can we stop instance variables from serialization?**  So if you want to prevent any instance variable from being serialized , do use transient. when you will de-serialize the object it will be initialized with a default value. |
| **will gc method destroy an unreferenced object?. Can you ensure that it will destroy the unused object?**  Yes but when can not say. We can gc explicitly using two ways:   1. Using System.gc() method : System class contains static method gc() for requesting JVM to run Garbage Collector. 2. Using Runtime.getRuntime().gc() method : Runtime class allows the application to interface with the JVM in which the application is running. Hence by using its gc() method, we can request JVM to run Garbage Collector.   There is no guarantee that any one of above two methods will definitely run Garbage Collector. |
| **Two objects referring to each other, not any other references. will garbage collector free the memory.** |
| **out of memory exception. What should you be carefull?**  Generally when our heap area gets full, we get out of memory error. To avoid this, we can try to avoid unnecessary object creation. Try to un-reference the objects after use. Try to create objects locally. |
| **How multithreading is useful in a single core cpu?**  Yes you can do multithreading on a single processor system.  In a multiprocessor system , multiple threads execute , simultaneously on different cores. Eg- If there are two threads and two cores , then each thread would run on an individual core.  In a single-processor system, multiple threads execute , one after the other or wait until one thread finishes or is by the OS , depending on the thread priority and the OS policy.But the running threads , gives an illusion thpreemptedat they run simultaneous , relative to the required application response time of the User space application. |
| **If there is a return statement in the catch block, will finally block execute?**  Yes. it will execute finally block. |
|  |
| **Can we use $\_ as a variable name?**  **Yes** |
| **syso('j'+'a'+'v'+'a'), what will be the output of this statement?**  It will give an addition of its ASCII-VALUE. |
| **How to increase bucket size of hashmap**  We can increase bucket size using load factor.  HashMap<Key,Value> map = new HashMap<Key,Value>(initial capacity,load factor);  HashMap<Key,Value> map = new HashMap<Key,Value>(20,0.90f);  **Default load factor is 0.75f.** |
| **What is volatile**  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. |
| **What is static?**  It is a keyword in java. If we use with a variable, then it will get memory after class loading. And use with the , thmethoden we can call it without instance.  If we use with a block then it will get execute after class loading |
| **Difference between volatile and static**  volatile variable value access will be direct from main memory and not use any cached values. It should be used only in multi-threading environments. static variable will be loaded one time. If it's used in a single thread environment, even if the copy of the variable will be updated and there will be no harm accessing it as there is only one thread.  Now if a static variable is used in a multithreading environment then there will be issues if one expects the desired result from it. As each thread has their own copy then any increment or decrement on a static variable from one thread may not reflect in another thread. |
| **Object level lock vs class level lock**  When we want to call synchronized non static methods then we need object level lock.  When we want to call synchronized static methods, then we need class level lock of that class. |
| **What is bucket level lock and what are its advantages**  When we use a concurrent hashmap, then it has a bucket level lock concept. In this, instead of locking all the map, it will lock only a particular bucket. So multiple threads can access the same map simultaneously. |
| **What is the Join method**  Join method in Java allows one thread to wait until another thread completes its execution. In simpler words, it means it waits for the other thread to die. |
| **What is Functional interface**  A functional interface is an interface that contains only one abstract method. They can have only one functionality to exhibit. From Java 8 onwards, lambda expressions can be used to represent the instance of a functional interface. Runnable, ActionListener, Comparable are some of the examples of functional interfaces |
| **What is string manipulation**  String manipulation (or string handling) is the process of changing, parsing, splicing, pasting, or analyzing strings.  For string manipulation, java has given different methods like slice,contains etc |
| **write a program to find duplicate in 2 strings**   1. **public class DuplicateCharacters {** 2. **public static void main(String[] args) {** 3. **String string1 = "Great responsibility";** 4. **int count;** 6. **//Converts given string into character array** 7. **char string[] = string1.toCharArray();** 9. **System.out.println("Duplicate characters in a given string: ");** 10. **//Counts each character present in the string** 11. **for(int i = 0; i <string.length; i++) {** 12. **count = 1;** 13. **for(int j = i+1; j <string.length; j++) {** 14. **if(string[i] == string[j] && string[i] != ' ') {** 15. **count++;** 16. **//Set string[j] to 0 to avoid printing visited character** 17. **string[j] = '0';** 18. **}** 19. **}** 20. **//A character is considered as duplicate if count is greater than 1** 21. **if(count > 1 && string[i] != '0')** 22. **System.out.println(string[i]);** 23. **}** 24. **}** 25. **}**   **Output:**  **Duplicate characters in a given string:**  **r**  **e**  **t**  **s** |
| **If we have a long method that you want to run. how would you run it using Runnable? In the instance you want to stop the thread at any given moment and get the results how would you implement your thread?**  Using Thread.interrupt() method  // Java program to illustrate  // stopping a thread  // using the interrupt() method  class MyThread implements Runnable {  Thread t;  MyThread()  {  t = new Thread(this);  System.out.println("New thread: " + t);  t.start(); // Starting the thread  }  // execution of thread starts from run() method  public void run()  {  while (!Thread.interrupted()) {  System.out.println("Thread is running");  }  System.out.println("Thread has stopped.");  }  }  // Main class  public class Main {  public static void main(String args[])  {  // creating objects t1 of MyThread  MyThread t1 = new MyThread();  try {  Thread.sleep(1);  // t1 is an object of MyThread  // which has an object t  // which is of type Thread  t1.t.interrupt();  Thread.sleep(5);  }  catch (InterruptedException e) {  System.out.println("Caught:" + e);  }  System.out.println("Exiting the main Thread");  }  } |
| **What implementation TreeSet uses for sorting?**  TreeSet uses a TreeMap internally, which is based on a "Red-Black tree" algorithm. |
| **Is there any way to block/stop the execution of finally block?**  1. An infinite loop above it will stop its execution 2. System.exit() above it will stop it |
| **What do you understand by parallel aggregate operation in stream api?** |
| **Explain Consumer, Supplier and Predicate interface?**  These are the functional interfaces that came in 1.8.  Consumer take one value as a input and process it  Supplier does not take any argument only returns some value  Predicate takes one argument and returns boolean |
| **How the interface is better over abstract class. I.e interface over implementation**  If we have a choice to have either interface or abstract class then we should go for interface. It allows you to use multiple inheritance.  And one more thing, interfaces are always lighter than any class. Because it does not contains constructor,instance creation etc |
| **Advantages of HAS-A or composition over inheritance or IS-A**  **HAS-A means** using properties of another class by creating an object of that class that is using that object we will access properties of another class.  **IS-A means** extending another class when we need it’s members.  When we need to access a few members of another class, then it’s good to go with HAS-A because unnecessarily we will import all other members which we don’t need if we extend it. |
| **Difference between StringBuffer and StringBuilder?**  StringBuffer is synchronized i.e. thread safe. It means two threads can't call the methods of StringBuffer simultaneously. StringBuilder is non-synchronized i.e. not thread safe. |
| **What are Wrapper Classes?**  We can wrap primitive data into objects using wrapper classes. In Java we have 8 wrapper classes like Byte,Short,Integer,Long,Float,Double,Character,Boolean. Wrapper classes convert primitive data into objects. |
| **Write a program to add 2 integers using lambda.**  public static void main(String[] args) {  BiConsumer<Integer, Integer> biConsumer = (n1,n2)->{  System.out.println(n1+n2);  };  biConsumer.accept(10, 20);  } |
| **wait vs join:**   * Most obvious difference, both are present different packages, the wait() method is declared in java.lang.Object class while join() is declared in java.lang.Thread class. * The wait() is used for inter-thread communication while the join() is used for adding sequencing between multiple threads, one thread starts execution after first thread execution finished. * We can start a waiting thread (went into this state by calling wait()) by using notify() and notifyAll() method but we can not break the waiting imposed by join without unless or interruption the thread on which join is called has execution finished. * One most important difference between wait() and join() that is wait() must be called from synchronized context i.e. synchronized block or method otherwise it will throw IllegalMonitorStateException but On the other hand, we can call join() method with and without synchronized context in Java. |
| **Find non repeated words in 2 sentences.** |
| **singleton design pattern implementation:**  We can implement singleton design pattern using different ways like   1. Using static keyword i.e Eager initialization  | // Eager Initialization  public class GFG  {  private static final GFG instance = new GFG();  private GFG()  {  // private constructor  }  public static GFG getInstance(){  return instance;  }  } | // Eager Initialization  public class GFG  {  private static final GFG instance =null;  Static{  instance = new GFG();  }  private GFG()  {  // private constructor  }  public static GFG getInstance(){  return instance;  }  } / | | --- | --- |  1. **Lazy initialization**   Here we create object only when user requests object and that too at first time   | public class GFG  {  private static GFG instance;  private GFG()  {  // private constructor  }  public static GFG getInstance()  {  if (instance == null)  {    instance = new GFG();  }  return instance;  }  } | | --- |  1. **Thread Safe Singleton**  | public class GFG  {  private static GFG instance;  private GFG()  {  // private constructor  }  synchronized public static GFG getInstance()  {  if (instance == null)  {  // if instance is null, initialize  instance = new GFG();  }  return instance;  }  } | Problem with this approach is,the method is synchronized, so every thread has to wait. That’s why we use **Lazy initialization with Double check locking so ,** only thread has to wait to access object | | --- | --- |  1. **Lazy initialization with Double check locking**  | public class GFG  {  private static GFG instance;  private GFG()  {  // private constructor  }  public static GFG getInstance()  {  if (instance == null)  {    synchronized (GFG.class)  {  if(instance==null)  {  instance = new GFG();  }    }  }  return instance;  }  } | | --- | |
| **sleep() vs wait()** |
| **Cohesion vs Coupling?**  Cohesion is the indication of the relationship within a module.  Coupling is the indication of the relationships between modules. |
| **Write a program to sort a Collection of Objects: Where the object is a Person with firstName and LastName.Sort the collection according to lastName in it’s natural order and print to console.**   | class Person{  private String firstName;  private String lastName;  public Person(String firstName, String lastName) {  super();  this.firstName = firstName;  this.lastName = lastName;  }  public String getFirstName() {  return firstName;  }  public void setFirstName(String firstName) {  this.firstName = firstName;  }  public String getLastName() {  return lastName;  }  public void setLastName(String lastName) {  this.lastName = lastName;  }      }  public class Demo{    public static void main(String[] args) {  Comparator<Person> c = (p1,p2)->{  return p1.getLastName().compareTo(p2.getLastName());  };  ArrayList<Person> al = new ArrayList<Person>();  al.add(new Person("Sam", "patil"));  al.add(new Person("ram", "sane"));  al.add(new Person("shyam", "anuse"));    Collections.sort(al,c);  for(Person s:al) {  System.out.println(s.getFirstName()+" "+s.getLastName());  }  }  } | | --- | |
|  |

Spring Questions

| 1. **Explain MVC architecture** 2. **What does controller annotation do?** 3. **What is the default scope of Spring bean? How can we change it?** 4. **explain JDBC connectivity** 5. **Advantages of Spring boot over spring?** 6. **Why is Spring Lightweight?** 7. **What is Dependency Injection?** 8. **What are the annotations that you used in java?** 9. **Difference between the path variable and request param?** 10. **What is autowiring?** 11. **What is @component?** 12. **How did you configure beans?** 13. **Who will intercept your request when you hit the UI?** 14. **What is Inversion of control?** 15. **How do you achieve inversion of control?** 16. **The advantage of IoC?** 17. **Difference between spring and spring-boot?** 18. **Difference between Spring REST and Spring MVC.** 19. **Difference between SOAP and REST?** 20. **When you are executing the query? When will you use executeQuery? executeUpdate? What will you use to select employees from the table?** 21. **Can you explain the callable interface?** 22. **Different between @Controller @Service @Repository** 23. **What is the default scope of bean? How can we change it?** 24. **What is the Http response code for success in REST** 25. **how to push a text file or pdf to REST Api** 26. **What is the IOC?** 27. **What is the design pattern of ioc?** 28. **what are the scopes in spring beans- all** 29. **What is the real advantage of web services other than language independence?** 30. **If the database is down how will you perform unit testing?** 31. **How to call a webservice from another web application.** 32. **How to resolve circular dependency in beans?** 33. **Which are the implicit objects in JSP?** 34. **Difference between save() and persist() methods.** 35. **How to perform server side validation and how to inform users the same?** 36. **How to rollback transactions in crudrepository?** 37. **How to deploy the web service?** 38. **How to secure endpoints of restful webservice?** 39. **If there are 2 Mvc classes or apps one who performs general functions like logging and transaction mgmt and another performs a specific job that you code then how would you make the connection between 2 ?** 40. **What are the servlets used in Spring** 41. **What are the configurations of Spring tell all in details** 42. **What is a Spring boot Actuator ?** 43. **Who creates Spring beans in Spring** 44. **Explain the spring app execution** 45. **How to launch Rest api with only JDK and browser** 46. **What is Idempotent ?** 47. **Explain different Response codes** 48. **Explain solid principles** 49. **How to exclude class from @ component scan in Spring Boot?** 50. **In Spring Boot how can we deploy applications on different servers than inbuilt tomcat?** 51. **In the Spring Boot Application if we have configuration classes for the development environment, How would you implement your test classes for a separate environment without modifying the development code using JUNIT.** 52. **How do you create separate objects for testing in Spring Boot if your development configuration has only Java Configuration, no XML.** 53. **What is MVC design pattern? What problem does it solve?** 54. **What is Service Oriented Architecture(SOA)? What is messaging, supply chain in terms of SOA?** 55. **What is the advantage of SOA?** 56. **What are different strategies in Hibernate?** 57. **Difference between lazy/eager loading in Hibernate?** 58. **What are the ways you can Inject Dependency in Spring?** 59. **What are the containers of Spring?** 60. **if we have url shortener microservice how will you implement microservices? How many you will implement.** 61. **How to limit spring beans in spring** 62. **The singleton bean thread safe?** |
| --- |

| **Explain MVC architecture.**  MVC is an architecture that separates business logic, presentation and data.  MVC is a systematic way to use the application where the flow starts from the view layer, where the request is raised and processed in the controller layer and sent to the model layer to insert data and get back the success or failure message.  Generally when we send a request from the browser, the request goes to the controller, controller communicates with database and business logic with the help of model and at the end response will display with the help of view. |
| --- |
| **What does controller annotation do?**  Controller annotation gives ability, so that class can accept http requests and after processing requests , it can send responses back to the client.  Also it tells the spring container that it is a spring bean, so the container will create an object of that class. |
| **What is the default scope of Spring bean? How can we change it?**  The default scope of spring bean is Singleton, which means for every bean it will return the same object. We can change it to prototype using scope attribute. |
| **Explain JDBC connectivity**  To perform jdbc communication we need to follow the following steps:   1. Register the driver 2. Create the connection 3. Create statement object 4. Add and send the query for execution 5. Gather the result 6. Close the connection |
| **Advantages of Spring boot over spring?**  Spring boot provides following features:   1. Create stand-alone Spring applications. 2. Embed Tomcat, Jetty or Undertow directly (no need to deploy WAR files) 3. Provide opinionated 'starter' POMs to simplify your Maven configuration 4. Automatically configure Spring whenever possible 5. Provide production-ready features such as metrics, health checks and externalized configuration 6. Absolutely no code generation and no requirement for XML configuration |
| **Why is Spring Lightweight?**   1. Spring provides you different modules and allows you to use any one based on your requirement, Ideally the spring jar is just 2-3 MB. 2. If you compare Spring with EJB, then you have to write very less code and configurations too. The beauty of Spring is that you can actually focus on business logic whereas in EJB you have to write a lot of code along with business logic which makes the code bulky and tightly coupled. 3. Through Spring you are playing with POJO which do not depend on Framework |
| **What is Dependency Injection?**  Dependency Injection (DI) is a design pattern that removes the dependency from the programming code so that it can be easy to manage and test the application.  Dependency Injection makes our programming code loosely coupled.  DI is a process of injecting dependency into dependent objects.  Spring has two types of DI:   1. Setter injection 2. Constructor injection |
| **What are the annotations that you used in java?**  Controller,RequestMapping,PathVariable,RequestParam,Getmapping,Postmapping,Query,Service,Component,Repository,RestController,Required,Value,PropertySources,Configuration,Bean,ComponentScan |
| **Difference between the path variable and request param?**  The key difference between @RequestParam and @PathVariable is that @RequestParam used for accessing the values of the query parameters where as @PathVariable used for accessing the values from the URI template.  **URI Templates** provide a structured way for defining how the URLs for your API should be constructed.  **Uniform Resource Identifiers** |
| **What is autowiring?**  Automatically injecting dependency objects into dependent objects is called autowired. We do this with @Autowired annotation |
| **What is @component?**  Because of this annotation Spring will automatically import the beans into the container and container will create objects and will manage it’s lifecycle.  It tells spring that it is bean class. |
| **How did you configure beans?**  We can configure spring using three ways:   1. Using xml configuration 2. Using annotation 3. Using java configuration |
| **Who will intercept your request when you hit from UI?**  Basically in my application, we have API gateway.so every request from outside will hit api gateway first. In API gateway we validate request using interceptor and then we forward to our helper controllers. |
| **What is Inversion of control?**  As the name implies Inversion of control means now we have inverted the control of creating the object from our own using new operator to container or framework. Now it’s the responsibility of the container to create objects as required. |
| **How do you achieve inversion of control?**  We can achieve IOC with the help of dependency injection. |
| **The advantage of IoC?**   1. decoupling the execution of a task from its implementation making it easier to switch between different implementations. 2. Greater modularity of a program. 3. Greater ease in testing a program by isolating a component or mocking its dependencies and allowing components to communicate through contracts |
| **Difference between spring and spring-boot?**  Boot is built on existing Spring f/w. To eliminate xml, Autoconfig, faster development. Cloud base and microservice. |
| **Difference between Spring REST and Spring MVC.**  REST is **an architectural style for communication based on strict use of HTTP request types**. MVC is an object oriented programming architecture based on separation of the functions of Data Model, Data Presentation and Logical program flow.  **MVC is about how the inner side of your app works.** **REST is about how your app "talks" with other apps**. You can combine them.  **While the traditional MVC controller relies on the View technology, the RESTful web service controller simply returns the object and the object data is written directly to the HTTP response as JSON/XML** |
| **Difference between SOAP and REST?Representational State Transfer)**   1. REST allows a greater variety of data formats, whereas SOAP only allows XML. 2. Coupled with JSON (which typically works better with data and offers faster parsing), REST is generally considered easier to work with. 3. Thanks to JSON, REST offers better support for browser clients. 4. REST provides superior performance, particularly through caching for information that’s not altered and not dynamic. 5. It is the protocol used most often for major services such as Yahoo, Ebay, Amazon, and even Google. 6. REST is generally faster and uses less bandwidth. It’s also easier to integrate with existing websites with no need to refactor site infrastructure. 7. This enables developers to work faster rather than spend time rewriting a site from scratch. Instead, they can simply add additional functionality. |

| **Advantages of SOAP over Rest:**  Because you can achieve most outcomes using either protocol, it’s sometimes a matter of personal preference. However, there are some use cases that SOAP tends to be better-suited for. For instance, if you need more robust security, SOAP’s support for WS-Security can come in handy. It offers some additional assurances for data privacy and integrity. It also provides support for identity verification through intermediaries rather than just point-to-point, as provided by SSL (which is supported by both SOAP and REST).  Another advantage of SOAP is that it offers built-in retry logic to compensate for failed communications. REST, on the other hand, doesn’t have a built-in messaging system. If a communication fails, the client has to deal with it by retrying. There’s also no standard set of rules for REST. This means that both parties (the service and the consumer) need to understand both content and context.  Other benefits of SOAP include:  (Simple Object Access Protocol  SOAP’s standard HTTP protocol makes it easier for it to operate across firewalls and proxies without modifications to the SOAP protocol itself. But, because it uses the complex XML format, it tends to be slower compared to middleware such as ICE and COBRA.  Additionally, while it’s rarely needed, some use cases require **greater transactional reliability** than what can be achieved with HTTP (which limits REST in this capacity). If you need ACID-compliant transactions, SOAP is the way to go.  In some cases, designing SOAP services can actually be less complex compared to REST. For web services that support complex operations, requiring content and context to be maintained, designing a SOAP service requires less coding in the application layer for transactions, security, trust, and other elements.  SOAP is highly extensible through other protocols and technologies. In addition to WS-Security, SOAP supports WS-Addressing, WS-Coordination, WS-ReliableMessaging, and a host of other web services standards |
| --- |
| **When you are executing the query? When will you use executeQuery? executeUpdate? What will you use to select employees from the table?**  To send and execute select query we use executeQuery method which returns ResultSet object containing records.  To send non select query we use executeUpdate method which returns integer number. |
| **Can you explain the callable interface?**  We use a callable interface to call PL/SQL procedures and functions. The Callable interface has a call method. |
| **Different between @Controller @Service @Repository**  Controller,Service,Repository are the child annotations of Component annotation.  All this annotations tells that the given class is spring bean. Along with that they have some extra ability given below:  **Controller:** It can handle http request and response and can return view page back to the client  **Service:** right now it is used only to inform the given class belongs to a service layer and it contains business logic.  **Repository:** it converts all the exceptions to the single exception DATAACCESSEXCEPTION. |
| **What is the default scope of bean? How can we change it?**  The default scope of spring bean is Singleton. But we can change it by using scope annotation or scope attribute in xml. |
| **What is the Http response code for success in REST**  **200 OK** |
| **how to push a text file or pdf to REST Api**  Send the file first in a multipart/form-data POST, and return an ID to the client. The client then sends the metadata with the ID, and the server re-associates the file and the metadata. |
| **What is the IOC?**  **Inversion of control: It**  is a principle where we transfer control of object creation or program to the container of framework. In spring object creation or dependency injection is totally handled by the spring container. Control is not with us. |
| **What is the design pattern of ioc?** |
| **what are the scopes in spring beans- all**  Default scope spring bean is singleton.  Other scopes are:   1. Singleton: It will single object which will share by all the components 2. Prototype: For every getbean request it will create new object 3. Session: for one session one object 4. Request: for one request one object |
| **What is the real advantage of web services other than language independence?**  Main adavantage of web service is cross platform.  Apart from it,   1. Web service is a unit of managed code which will be access using HTTP requests. 2. Web services allow you to expose the functionality of your existing code over the network. Once it is exposed on the network, other applications can use the functionality of your program. 3. Web services use standardized industry standard protocol for the communication.This standardization of protocol stack gives the business many advantages such as a wide range of choices, reduction in the cost due to competition, and increase in the quality. |
| **If the database is down how will you perform unit testing?**  First thing: Using a database for unit testing is not a good choice. Because for testing every functionality going to database is costly call. For testing using external resources is not good.  To test we can use hard code stuff or we can use inmemory database just like H2 database. |
| **How to call a webservice from another web application?**  From spring, we can call web service using RestTemplate class. |
| **How to resolve circular dependency in beans?**  Circular dependency is the problem which generally occurs when Bean A is depend is depend on Bean B and Bean B is depend on Bean A. and when we are using constructor injection we will get problem.  So,  We can use setter injection in this case or @Lazy annotation we can use. Better to use setter injection. |
| **Which are the implicit objects in JSP?**  There are nine implicit objects given in JSP  request,response,page,pageCOntext,session,exception,out,application,config |
| **Difference between save() and persist() methods.** |
| **How to perform server side validation and how to inform users the same?**  We can use JSR Bean Validation API for server side validation and it will return respective message to the client as well. |
| **How to rollback transactions in crudrepository?**  @Transactional(rollbackFor = CustomException.class)  And if we are satisfy with operation then then throw CustomException manually. |
| **How to deploy the web service?** |
| **How to secure endpoints of restful webservice?**  We can use JWT token or Spring Security. |
| **If there are 2 Mvc classes or apps one who performs general functions like logging and transaction mgmt and another performs a specific job that you code then how would you make the connection between 2 ?**  Using Spring AOP module. |
| **What are the servlets used in Spring?**  We use DispatcherServlet in spring framework which acts as a Front Controller. Every requests comes to front controller first then it dispatches to the respective helper controller. |
| **What are the configurations of Spring tell all in details**  We can configure spring bean using three ways:   1. Xml file 2. Annotations 3. Java configurations |
| **What is a Spring boot Actuator ?**  Spring Boot Actuator is a sub-project of the Spring Boot Framework. It includes a number of additional features that help us to monitor and manage the Spring Boot application. It contains the actuator endpoints.  For example: we can check wether database connection is up or not. How much disk space is used and how much remained. We can take dump file of heap area.  We can write custom endpoints as well to perform custom requirement. |
| **Who creates Spring beans in Spring?**  Beans will be created by IOC containers. There are two containers BeanFactory and ApplicationContext. In spring MVC we use ApplicationContext. |
| **Explain the spring app execution.**  When we deploy an application, ioc container will instantiates all the beans with singleton scope. Then when we send request, it will goto dispatcherServlet first and then it will dispatch request to respective controller. Once controller gets response from service layer, controller class will return view page name. View page name will return to again dispatcherServlet, which will call respective view page with the help of view resolver. |
| **How to launch Rest api with only JDK and browser** |
| **What is Idempotent ?**  An operation (or service call) to be idempotent, clients can make that same call repeatedly while producing the same result. In other words, making multiple identical requests has the same effect as making a single request. Note that while idempotent operations produce the same result on the server (no side effects), the response itself may not be the same (e.g. a resource's state may change between requests).  The PUT and DELETE methods are defined to be idempotent. However, there is a caveat on DELETE. The problem with DELETE, which if successful would normally return a 200 (OK) or 204 (No Content), will often return a 404 (Not Found) on subsequent calls, unless the service is configured to "mark" resources for deletion without actually deleting them. However, when the service actually deletes the resource, the next call will not find the resource to delete it and return a 404. However, the state on the server is the same after each DELETE call, but the response is different.  GET, HEAD, OPTIONS and TRACE methods are defined as safe, meaning they are only intended for retrieving data. This makes them idempotent as well since multiple, identical requests will behave the same.  POST is not idempotent. |
| **Explain different Response codes**  **Follow given link**  [**https://www.tutorialspoint.com/http/http\_status\_codes.htm**](https://www.tutorialspoint.com/http/http_status_codes.htm)  **The Status-Code element in a server response, is a 3-digit integer where the first digit of the Status-Code defines the class of response and the last two digits do not have any categorization role. There are 5 values for the first digit:**   | **S.N.** | **Code and Description** | | --- | --- | | **1** | **1xx: Informational**  **It means the request has been received and the process is continuing.** | | **2** | **2xx: Success**  **It means the action was successfully received, understood, and accepted.** | | **3** | **3xx: Redirection**  **It means further action must be taken in order to complete the request.** | | **4** | **4xx: Client Error**  **It means the request contains incorrect syntax or cannot be fulfilled.** | | **5** | **5xx: Server Error** | |
| **Explain solid principles** |
| **How to exclude class from @ component scan in Spring Boot?**  We can exclude class using excludeFilters.  ComponentScan#excludeFilters can be used to exclude component classes from scanning. For example  @ComponentScan(basePackages = "com.logicbig.example.client;com.logicbig.example.service",  excludeFilters = @ComponentScan.Filter(  type = FilterType.ASSIGNABLE\_TYPE,  classes = {WholeSaleOrderService.class, Wholesaler.class}) |
| **In Spring Boot how can we deploy applications on different servers than inbuilt tomcat?**  **Steps:**   1. Change packaging to war 2. Add following dependency for external tomcat or you can add respective dependency for other servers  | <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-tomcat</artifactId>  <scope>provided</scope>  </dependency> | | --- |  1. Do changes to main class as given below  | @SpringBootApplication  public class App extends SpringBootServletInitializer  {  public static void main( String[] args )  {  SpringApplication.run(App.class, args);  }  } | | --- | |
| **In the Spring Boot Application if we have configuration classes for the development environment, How would you implement your test classes for a separate environment without modifying the development code using JUNIT.** |
| **How do you create separate objects for testing in Spring Boot if your development configuration has only Java Configuration, no XML.** |
| **What is MVC design pattern? What problem does it solve?**  **MVC stands for Model View Controller.**  It helps us to separate data(model),UI(View) and logic(Controller) coding.  Well, just as how having separated functions solve the problems of readability, modularity, and coupling, so does MVC. Say if you wanted to change a piece of code, you can tackle it in a smaller subset that is more or less isolated from the larger piece of code. This allows you to add, modify or remove code more efficiently and logically. |
| **What is Service Oriented Architecture(SOA)? What is messaging, supply chain in terms of SOA?**  In service oriented architecture we divide our project into different services. Each service is called a microservice. |
| **What is the advantage of SOA?**   1. We divide projects into multiple services. 2. Each service is deployed on separate servers. So we can take advantage of multiple JVM. Instead of putting burden on a single JVM, we can divide the burden. 3. Parallel development possible. 4. Failing one service will not affect another one. 5. We can use different technology in different services of the same application. 6. We can divide the server load. |
| **What are different strategies in Hibernate?** |
| **Difference between lazy/eager loading in Hibernate?** |
| **What are the ways you can Inject Dependency in Spring?**  There are two ways to inject dependency:   1. Setter injection: which injects dependency through setter method. 2. Constructor injection: which injects dependency through constructor.   Setter is better in case of circular dependency. In circular dependency, constructor injection will create problems. |
| **What are the containers of Spring?**  We have two IOC containers in Spring:   1. Bean Factory: It is a legacy container. It is a lazy loading container. It creates beans only when we call it. 2. ApplicationContext: It is a JEE container. It is an eager loading container. It creates beans on loading only. |
| **if we have url shortener microservice how will you implement microservices? How many you will implement.** |
| **How to limit spring beans in spring?**  First up we should try to avoid unnecessary objects. If possible try to use a singleton pattern. We can split applicationContext into multiple files. |
| **The singleton bean is thread safe?**  Singleton Beans are thread safe or not depending on how the class whose scope is singleton is written. Each calling thread will have its own execution and does not interfere with another thread's execution unless there is some code in the singleton scoped class which is shared by all calling threads.e.g if a class has globally declared variables that is sass as accessed by its method and the values are modified then this may cause concurrency issue so it is better to have those variables at the method level and not at the class level. |