Java

1. What is Java?
    - OOP programming language
    - Common programming language
    - Large developer community and support
    - Compiled programming language where the source code needs to be compiled into bytecode before it can run

2. Why would you choose Java?
    - Large developer community
    - Huge amount of documentation
    - Great exception handling
    - Good for enterprise level applications
    - A lot of environments support java
    - Forwards compatible

3. Is Java 100% Object Oriented Programming? Why or Why Not?
    - No because of primitives

4. JDK vs JRE vs JVM
    - Java Development Kit contains compiler and debugger tools to compile the Java source code, also includes Java Runtime Environment and Java Virtual Machine
    - The JRE provides an environment and in that environment the JVM runs the compiled byte code. It contains both the library and the JVM.
    - The JVM creates a layer of abstraction from the Java Code and the computer environment. It is used to convert the java bytecode into machine readable code.

5. State vs Behavior
    - States are properties of an object
    - Behaviors are actions that an object can take

6. Class vs Object
    - An object is an instantiation of a class
    - The class is the blueprint that provides the states and behaviors its instantiated objects will have

7. What are the class members?
    - Instance variables
    - Methods
    - Constructors
    - Inner class
    - static/instance blocks

8. What are the types of variables?
- Instance variable
- Local variable
- Static variable

9. What are the scopes of the variables?
- Class - inside class
- Method - inside method
- Block - inside for or if statements

10. What are the Access Modifiers?
- Public - globally accessible
- Protected - accessible within the package and its subclasses
- Default - accessible only within the package
- Private - accessible only within that class

11. What are the Non-Access Modifiers?
- Static - variable or function is shared between all instances of that class as it belongs to the type, not the actual objects themselves
- Final - define an entity that can only be assigned once. Once a **final** variable has been assigned, it always contains the same value.
- Abstract - purpose of extending
- Synchronized - used for threads; accessed by only one thread at a time
- Volatile - used for threads; lets JVM know that a thread accessing a variable must merge its own private copy with the master copy in memory
- Transient - used in serialization; field that is transient won't be sent at all and will appear as null on the other end

12. What is a Constructor?
- A method that creates an instance of a class based on the parameters based into the constructor

13. What are the types of Constructors?
- Default (created by the JVM for you if no constructor is specified)
- No args
- Other

14. What are the 4 pillars of Object Oriented Programming?
- Abstraction
- Encapsulation
- Inheritance
- Polymorphism

15. What is Abstraction?

- Abstraction is focused on the concept of reusability which is achieved using generalization or specialization. In abstraction, the programmer's focus is what happens, rather than the implementation. An example of abstraction is using abstract classes. To reiterate the point of generalization and specialization, an abstract class provides general states and behaviors that can be overridden by classes that extend it, thus creating a unique implementation of that abstract class.

16. How do you achieve Abstraction?
    - Abstract class
    - Using a framework, such as Spring

17. Abstract Class vs Interface
    - Typically, a programmer would use an abstract class to denote a kind of hierarchy between the abstract class and the classes that extend it. An interface is just implemented, so the interface can be used across a variety of classes that may not be related at all.
    - Can only extend one abstract class
    - Can implement as many interfaces as you would like
    - If a programer wants to have method definitions or contain variables that aren't static and final, they would use an abstract class. The methods in an abstract class can also be overridden, while in an interface, all the methods must be implemented by the class that implements the interface

18. What is Polymorphism?
    - Polymorphism is the concept that in Java, objects can have the same states and behaviors, so they are interchangeable. The JVM will determine the program's intentions at runtime or compile time. This refers to runtime polymorphism and compile time polymorphism. Runtime polymorphism is also known as dynamic polymorphism, and an example of this is method overriding. Method overriding is runtime polymorphism because at runtime, the JVM will go look for the method from parent to child class based on the override annotation. If the method signature matches, then it will use that, otherwise it will go look for the parent's method. Method overloading is compile time polymorphism. At compile time, assuming there is more than one method with the same method name, the JVM will find the appropriate method based on the return type and parameter list.

19. How do you achieve Polymorphism?
    - Method overriding
    - Method overloading

20. What are the types of Polymorphism?
    - Compile time polymorphism (static)
    - Runtime polymorphism (dynamic)

21. What is Inheritance?
    - Inheritance is the concept that classes can share states and behaviors through extension. This relationship between classes is made between the parent or super class and the child or subclass.

22. How do you achieve Inheritance?
    - Extend class

23. What is Encapsulation?
    - Encapsulation is a technique to control an object's accessibility to its states. You can achieve this control using access modifiers. The access modifiers are public, protected, default, and private. Public refers to global accessibility. Protected refers to accessibility within the package and its subclasses. Default refers to accessibility within the package and not its subclasses. Private refers to accessibility within that class only. These access modifiers are used for info hiding or data hiding.

24. How do you achieve Encapsulation?
    - Access modifiers

25. BEAN vs POJO
    - A POJO is a plain old java object
    - A bean is a POJO but with more restrictions
        - Must have a no args constructor
        - Must have private instance variables
        - Must have getters and setters

26. What is a hashcode?
    - A hashcode is an identifier for an object that has been converted to an integer. Although a hashcode should be unique, hash collisions may happen. A hash collision is when two different objects share a hashcode. Two same objects will have the same hashcode, but two same hashcode won't necessarily refer to the same object.

27. STACK vs HEAP
    - A stack refers to Java Stack Memory, which is LIFO. It's a stack of memory. Whenever a method is invoked, a new block is created in the stack memory for the method to hold local primitive values and reference to other objects in the method. As soon as method ends, the block becomes unused and become available for next method. Stack memory size is much less compared to Heap memory.
        - StackOverflow
    - A heap refers to the Java Heap Space, which is used by Java runtime to allocate memory to Objects and JRE classes. Whenever we create any object, it's always created in the Heap space. Garbage Collection runs on the heap memory to free the memory used by objects that don't hold any more reference. Any object created in the heap space has global access and can be referenced from anywhere of the application.

    - OutOfMemoryError

28. Final vs Finally vs Finalize
    - Final is a non-access modifier that marks your variable as the value that it is initialized at forever, so a constant. Can also be applied to a class to make it so that no one can extend that class or a method to prevent subclasses from overriding that method.
    - Finally refers to the finally clause in a try, catch, finally block. Finally will always execute, even if there is an exception. The only time that finally wouldn't happen is if you call system.exit or if there's a power failure or if you exit the application or anything that would be abnormal in the try/catch block.
    - Finalize - a method that the garbage collector invokes when the JVM figures out that a particular instance should be garbage collected. The main purpose of using finalize is to release resources used by objects before they're removed from the memory. Programmers are strongly recommended to not override the finalize method because the garbage collector will panic and continue calling finalize on the object.

29. What does the keyword "Static" mean?
    - Static is a non-access modifier that means you can use that method or variable from a class without instantiating that class. Static also means that it belongs to the class itself and not an instance of a class. For static variables there's 1 shared copy of the variable among all instances of the class and static methods can only use static variables.

30. What is an Exception? What class does it extend?
    - Exception extends Throwable, which has Exception and Errors as subclasses and RuntimeException, which is a subclass of Exception..
    - Exceptions are events in execution which deviate or interrupt from the normal flow of execution.
    - You can throw and catch exceptions.

31. What is a Try-Catch-Finally Block?
    - A try-catch-finally block is used to handle checked exceptions so that the JVM will compile the code. In the try block, the programmer enters the code that may need to be caught. In the catch block, the programmer handles the potential exception or exceptions. The finally block will execute after the catch block always, unless the system exit method is called or the power fails or any such other exceptional states.

32. Throw vs Throws
    - Throw - method body to invoke an exception; act of throwing an exception explicitly; throw is followed by an instance variable
    - Throws - method signature; used to declare an exception; throws is followed by exception class names

33. What is the Garbage Collector?

- The garbage collector is a long running thread (daemon thread) which gets rid of objects that no longer hold a reference in the heap to free up space.

34. Checked vs Unchecked Exceptions
    - Checked Exceptions - fall under Exceptions; must be thrown or surrounded in a try/catch block otherwise the JVM won't compile the code. Outside resource failing.
        - Exception, IOException, FileNotFoundException, SQLException
    - Unchecked Exceptions - fall under RuntimeExceptions, which is a subclass of Exception. Unchecked Exceptions can be compiled and will throw an exception at runtime. Unchecked exceptions can be caught or handled with logic.
        - RuntimeException, ArrayIndexOutOfBoundsException, ArithmeticException

35. RunTime vs CompileTime
    - Compile time is when the JVM compiles your java source code into bytecode.
    - Runtime is when your compiled Java code is run when a user uses your application.

36. What are some types of exceptions?
    - Errors: StackOverflow, VirtualMachineError, OutOfMemoryError
    - Exceptions: FileNotFoundException, IOException, SQLException
    - RuntimeException: ArrayIndexOutOfBounds, NullPointerException, RuntimeException

37. String vs StringBuilder vs StringBuffer
    - String - Immutable (cannot be changed or extended); stored in string pool; thread safe
    - StringBuilder - Mutable; for strings changing rapidly; do not extend string because string is final
    - StringBuffer - Mutable; synchronized, much slower; do not extend string because string is final; thread safe, useful for threads/strings being manipulated by multiple threads

38. == vs .equals()
    - == operator - Strict comparison by value and identity; used for both primitive and objects comparison
    - equals() Method - Comparison by value only; method can be overridden ; used for objects comparison

39. What is File I/O?
    - File Input Output
    - Java uses the concept of a stream to speed I/O. streams can be used to read/write from and to the console, files, data, and more.

40. What is Serialization?
    - Serialization is the process where java objects are converted to byte streams
    - Serialization is for transferring objects from Java to Java applications only.
    - Serialization (object to stream) and deserialization (stream to object)
    - Must use the serializable interface to be marked for serialization

- Transaction Isolation Level
- Invalid class exception
- Not serializable exception
- Class not found exception

41. What are the Data Structures in Java?
    - Various structures of holding data
    - Array - one block of memory; constant access time
    - ArrayList - dynamically resizable; object
    - Vector - dynamically resizable; object
    - LinkedList - uses nodes
    - Set - unique values; sortable
    - Queue - FIFO
    - Tree - binary search tree
    - Map - key value; not sortable
    - Deque - accessible from both ends
    - Stack - LIFO

42. What is a Generic?
    - A Generic is a template.which enables programmers to specify Generic methods and classes with a single method declarations, a set of related methods, or with a single class declaration
    - An example of a Generic is using the diamond syntax, and an example of using the diamond syntax is when you create a reference to to a List of some object type and initialize an ArrayList.

43. What is a Wrapper Class?
    - A Wrapper Class is meant to encapsulate primitive data types to turn them into objects using an operation of the JVM called autoboxing. You can also unbox an object to a primitive.
    - Can use object methods
    - Can be used with Generics

44. What is a Collection?
    - Iterable -> Collection -> Set, List, Queue
    - Collection - interface
    - Collections - utility class that has static methods for doing operations on objects of classes which implement the Collection interface

45. What is an Array?
    - A data structure containing like-typed variables that is initialized with a certain block of memory that use indexes which allow for consistent random access

46. What is a Set?

- An interface which extends collections. A set is an unordered collection of objects which can only contain unique values.
- Set has various methods to add, remove, size and more to enhance the usage of this interface

47. What are the types of sets?
    - HashSet
    - LinkedSet
    - TreeSet

48. What is a HashSet?
    - Implements the Set interface; also implements Serializable and Cloneable
    - Uses the hashtable as an underlying data structure
    - Duplicate values are not allowed
    - Objects that you insert in a HashSet aren't guaranteed to be inserted in the same order. Objects are inserted based on their hash code
    - Hash refers to how you go and find the value.
    - HashSet - only values, set of hashed values
    - HashMap - key/values, map where the keys are hashed
        - Collisions - update the older one

49. What is a SortedSet?
    - Not ordered
    - Implements the Comparable interface
    - All elements must be mutually comparable - compared elements accept each other as the argument to their compareTo method
    - Can be sorted in a natural order with Comparable interface or using a Comparator

50. What is a TreeSet?
    - An implementation for the SortedSet interface
    - No duplicate values
    - Doesn't preserve the insertion order of elements but elements are sorted by keys
    - An implementation of a self-balancing binary search tree

51. What is a List?
    - An interface that implements Collection
    - An ordered collection of objects in which duplicate values can be stored
    - By ordered, it means that it preserves the insertion order to allow positional access and insertion of elements

52. What are the types of lists?
    - ArrayList
    - LinkedList
    - Vector

- Stack

53. What is an ArrayList?
    - Inherits AbstractList class and implements List interface
    - Increases by half its size
    - Constant time access
    - Dynamically sized arrays
    - Not thread safe
    - Can only contain objects; not used for primitive types, but you can use a wrapper class for such cases

54. What is a LinkedList?
    - A linear data structure.
    - Not contiguous location.
    - Nodes are used to link elements by storing a reference to the next linked element.

55. STACK vs VECTOR?
    - STACK is a data structure which extends Vector, and it stores objects in a last-in-first-out matter. The last object to be added will be the first one to be removed.
    - VECTOR is a resizable data structure. It allows duplicate elements to be inserted and preserves insertion order. VECTOR is synchronized, so it is thread safe. Vector implements RandomAccess interface, so its retrieval methods are very fast. A vector increases in size by double. The Vector includes five methods that allows it to be treated like a stack. Those five methods are push, pop, peek, empty, and search.

56. What is a Queue?
    - A Queue is an interface that implements Collection. It is for data structures that use the first-in-first-out principle, meaning that the first object put into a queue will be the first one to be removed. Queues are also dynamically sized.
    - Implementations of Queue involve the LinkedList, PriorityQueue, and Deque.
    - LinkedLists - is a linear data structure where memory isn't stored in a contiguous location, and each element contains a reference to the next element, which is known as a node. LinkedLists also implement List.
    - PriorityQueue - is a data structure that is ordered according to natural ordering or using a Comparator. A priority queue that uses natural ordering won't allow elements that can't be compared.
    - Deque - (Double Ended Queue pronounced as deck) is a data structure that can be accessed from both ends, meaning that it can be used as a FIFO or LIFO data structure

57. What is Multithreading? How do you do it?
    - Multithreading is when your threads are running concurrently and handling different tasks. With one processor, multithreading isn't possible, but with multiple processors, it is. The OS divides processing time among different applications and also among each thread within an application.

58. What is a Process?
- A program in execution is often referred to as a process. A process can consist of multiple threads, and a thread is the smallest part of the process that can execute concurrently with other threads of the process.

59. What is a Thread?
- A thread is a part of a process that runs a path of the program in execution. It runs asynchronous to the main method thread. When your program starts up, at least one thread is created by the JVM, which is the main method thread. Another example of a thread is the garbage collector.

60. What is Singleton?
- A Singleton is a class that can only have one instance of that class at a time. Its purpose is to limit the number of objects to only one. An example of a Singleton would be a Factory. After the first time, if we try to create another instance, it will point to the first instance created. To design a Singleton class, you make the constructor private and write a static accessor method that gets your Singleton.

61. What are Reflections?
- Reflections refers to the accessing and manipulating a class at runtime. When the JVM compiles code, it will leave the names of methods and data and leave executable machine code, so that another method can take an object and read its method names and data names. Reflection is generally used in frameworks to operate on objects without having those objects be of any particular class.

62. What is varargs?
- VarArgs are short for variable arguments, which consist of a number of variable arguments. VarArgs are used when an indeterminate amount of arguments could be included. VarArgs are array-like. They can only be one type and one per method, and they must be used as the last parameter.
- Javascript equivalent - rest operator, spread operator

63. ArrayList vs Vector?
- ArrayList and Vectors can both be dynamically sized, but a Vector increases by double its size while an ArrayList increases by half of its size. Both Vectors and ArrayLists implement List, and they both allow RandomAccess. A Vector, however, is thread safe and ArrayLists are not. Meaning a change in a vector used across different threads would be reflected in the instance of that vector across those threads.

64. TreeSet vs HashSet?
- Both TreeSets and HashSets implement the Set interface. TreeSets also implement the SortedSet interface. The TreeSet is an implementation of a binary self-balancing tree, whereas HashSets are sets but with their values hashed.

65. Can constructors be private?
  - Yes. A private constructor would create a Singleton.

66. What is the difference between 'Collection' vs 'Collections'
  - Collection is the interface that implements Iterable. It marks a class as a Collection and therefore also Iterable.
  - Collections is the java util package that provides static methods available to Collection classes.

67. What is the main purpose of Version Control Software?
  - Version Control Software is software that is responsible for the management of changes to your files, documents, programs, etc. It records changes so that you can recall specific versions later. An example of a VCS is Git, which we use to store individual and collaborative projects. Git is used for versioning static files, text files, but it can’t version binary files.

68. What are the data types in Java? (8)
  - Boolean - true or false
    - Default false
    - 1 bit
  - Char - character type whose values are 16-bit Unicode characters
    - Default \u0000
    - 16 bits
  - Byte
    - Default 0
    - 8 bits
  - Short
    - Default 0
    - 16 bits
  - Int
    - Default 0
    - 32 bits
  - Long
    - Default 0
    - 64 bits
  - Float
    - Default 0.0
    - 32 bits
  - Double
    - Default 0.0
    - 64 bits

69. How do you invoke a static method?

- A static method can be invoked within the class that it is declared in without needing to create an instance of that class to invoke the method. Outside the defining class, a static method is called by prefixing it with its class name. It can also be qualified with an object, which will be ignored, but the class of the object will be used.

70. What is casting?
    - Casting is the process of taking an object of a particular type and turning it into another type, which gives you access to that type's methods. Casting is used when going from a more general to specific type. For example, Integer inherits from Number, so if you want to store an Integer as a Number, that's okay, since all Integers are Numbers. But if you want to go the other way around, you need a cast, since not all Numbers are Integers.

71. How many objects on the heap if I cast?
    - Because casting an object won't create another object in the heap, there will only be that one object in heap plus whatever else was already existing in the heap.

72. What is an Interface?
    - An interface is a class that other classes can implement. It is a contract which includes method signatures (but no method definitions) and sometimes static final variables which must all be implemented by classes that use that interface. An interface is helpful for providing semantic meaning across a number of classes without implying them to be related via inheritance extension.

73. Can I instantiate an Abstract class? Constructor?
    - Abstract classes can't be instantiated, but they can be subclassed. Abstract classes do have constructors, and those constructors are invoked when a concrete subclass is instantiated.

74. Why would you use an Abstract class over an Interface?
    - An abstract class, unlike an interface, doesn't work like a contract, so all of the inherited methods don't have to be implemented, and those that you want to use can be used using the parent's definition or by overriding. Abstract classes also allow you to inherit class variables, as long as they aren't static. Abstract classes are used when the relationship between the abstract class and the class extending it is hierarchical, when you want to inherit states, and when you want to define a method definition. (Interfaces can actually also provide default methods without affecting implementing classes, as it includes an implementation. An implementing class can override the default implementation provided by the interface.)

75. Can I force garbage collection?
    - No, you can't force the garbage collector to scan the heap. The garbage collector starts when the first user thread is started by the JVM, and it is terminated when the last user thread terminates. You can invoke the method (System.gc) to run the garbage collector,

but it does NOT guarantee that the garbage collector will run. The JVM is in charge of when a thread will run, and the garbage collector is a thread.

76. Which method does the garbage collector call?
    - The garbage collector calls finalize() before destroying the object which is eligible for garbage collection. This method shouldn't be overridden. You can trick the garbage collector by letting it think the object was already collected. User threads will be frozen. The finalize method was also deprecated in Java 9.

77. What is the finally block?
    - The finally block is the block that executes after a try-catch block. A finally block will always execute, even if an exception is thrown. A finally block won't execute when the system exit method is invoked, if the power goes out, or any other extreme circumstances occur.

78. Is a catch block needed?
    - A catch block is needed if you have a checked exception otherwise the JVM won't compile the code, but if you throw the exception in the method signature, you can also simply have a try-finally block.

79. What are Generics for?
    - Generics enable types (classes and interfaces) to be parameters when defining classes, interfaces, and methods. They are used for type-checking at compile time, so that code will be more type-safe. Using Generics also eliminates the use of casts. An example of using Generics is in specifying the type of a Collection using diamond syntax.

80. Comparable vs Comparator
    - Comparable is an interface that compares an object based on natural ordering (numbers and letters). It uses a convention of positive number for greater than the compared element, negative number for less than the compared element, and 0 for equal.
    - Comparator is an interface that implements Comparable and implements its compareTo method for custom ordering. Comparator allows you to implement your own rules.

81. Hashtable vs Hashmap
    - Hashtable - synchronized; doesn't allow null values for keys
        - Legacy class that has been re-engineered to implement the Map interface
        - Extends from Dictionary, an abstract class
    - Hashmap - not synchronized (better for non-threaded applications; allows one null key and any number of null values
        - Subclass of the AbstractMap class and a member of the Java Collection Framework right from the beginning of its introduction
        - Doesn't guarantee the order of the map will remain constant over time

82. How do I start a thread implementing Runnable?

- Invoke the run method to start a thread implementing Runnable.
- The Runnable object needs to be passed through the thread constructor and have invoke start.

83. Thread methods. (run, start, sleep, wait, notify, notifyAll, join).
    - Run - Executes starts the thread of execution.
    - Start - Creates a new thread and calls run().
    - Sleep - static method of a thread, doesn't release resource locks.
    - Wait - method of an object, releases resource locks.
    - Notify - wakes up only a single thread from a wait state to use a resource.
    - NotifyAll - wakes up all threads waiting for a resource.
    - Join - causes the current thread to wait for another thread to finish.

84. Difference between the run and start method?
    - Run is a method of an object calling runnable
    - Start is a method of an object inheriting from the Thread class
    - When you call run it executes the run method.
    - When you call start it instantiates a new thread and calls run.

85. What is a deadlock?
    - Deadlock is another component of the producer consumer problem.
    - Deadlock is when two or more threads are waiting for resources that the other(s) have.

86. Which is the fastest File IO available and why?
    - File Input Reader - reads your file line by line

87. Which are the scopes of a variable?
    - Instance
        - Only available to objects of the implementing class.
    - Local
        - Only available within the same block.
    - Static
        - Only available to the implementing class or block.

88. Can you override static methods?
    - No
    - Static methods are not associated with class instances so overriding is not applicable.
    - Static methods of a class cannot be accessed by its objects only the implementing class.

89. What is shadowing?
    - Shadowing is when two variables or objects with the same name and type have overlapping scopes.
    - When this happens the field with the larger scope is hidden.

90. What are Wrapper classes?
- Wrapper classes in java are classes bult to add functionality to primitives.
- Primitives can be turned into a wrapper object that has its own methods that can be used.
    - Integer
        - Has methods to return a base change value as a string
        - Change the current number to a string
        - Calculate the integer value of a number (“1” = 1)
    - Wrappers
        - Byte
        - Short
        - Integer
        - Long
        - Float
        - Double
        - Character
        - Boolean

- Also handy for generics since generics don’t work on primitive types.

91. What is Varargs used for?
- Short for variable length arguments
- When a method can have any number of parameters and overloading the method would cause too much ambiguity.
- So you can pass ellipses into the parameter of your method after the first argument and then access the varargs array.
- Varargs is an array of arguments that is an array of data passed in when a method is called, args is the array containing all the parameters passed into the function.

92. What is the difference between protected and default?
- Protected locks access of resources to class and subclass instances of the class in question.
- Default locks the access of resources to the package.

93. What is the final keyword used for?
- Final is a non-access modifier
- Final used on objects and variables makes them immutable
- Final used on a class makes it uninheritable.
- Final used on methods make them non overridable.

94. What is the difference between StringBuilder and Buffer?
- String builder is unsynchronized; faster
- String buffer is synchronized; slower

95. What is synchronization?
- Synchronization is the process of making resources thread-safe
- Done using the ‘synchronize’ keyword
    - Makes object only accessible by one thread

96. How do you go about starting a thread?
- If implementing Runnable, invoke the method run
- If extending Thread, invoke the method start, which will invoke the method run

97. What is the difference between a List of a Set?
- Set values need to be unique

98. Some concrete implementations of Set.
- Java has three general purpose ones.
    - HashSet
        - Methods:
            - Add - adds an element to the set if it’s not already present
            - Clear - removes all elements from the set.
            - Clone - returns a shallow copy of the hashset (elements themselves are not cloned)
            - Contains - returns true if set holds the specified element
            - isEmpty - returns true if the set is empty.
            - Iterator - returns an iterator over the elements of this set.
            - Remove - removes the specified object from the set if present.
            - Size - returns the cardinality of the set.
    - TreeSet - Combination of Tree and Set. (more sortable)
        - Methods:
            - Add - adds specified element if it doesn’t exist.
            - addAll - adds the contents of a collection to the set.
            - Ceiling - returns the least element of the set that is greater than or equal to the specified element, or null if no such element can be found.
            - Clear - removes all elements from this set.
            - Clone - returns a shallow copy of the set.
            - Comparator - returns the comparator used to order the elements of this set or null if using natural ordering.
            - Contains - returns true if this set contains the specified element.
            - descendingIterator - returns an iterator over the elements in descending order.
            - descendingSet - returns a reverse order view of the elements in the set.
            - First - returns the lowest element in the set.
            - Floor - returns the greatest element in the set that is less than or equal to the element, or null if it doesn’t exist.

    - headSet - returns a view of the portion of this set whose elements are less than (or equal to) to the specified element.
    - Higher - returns the lease element in this set that is strictly greater than the element or null if DNE.
    - isEmpty - returns true if set is empty
    - Iterator - returns an iterator over the elements in ascending order.
    - Last - returns the highest element currently in the set.
    - Lower - returns the greatest element in the set strictly less than the given element, or null if DNE.
    - pollFirst - retrieves and removes the lowest element, or returns null if empty.
    - pollLast - retrieves and removes the highest element, or returns null if empty.
    - Remove - removes the specified element if present.
    - Size - returns the cardinality of the TreeSet.
    - Subset - returns a portion of the set whose elements range from the lowest specified to the highest specified.
    - tailSet - returns a view or portion of the set greater than or equal to if inclusive is true to the specified element.

  - LinkedHashSet
    - Inherits from HashSet.

99. LinkedList vs ArrayList
  - ArrayList internally uses an array to store data.
  - LinkedList uses doubly linked lists to store data.
  - ArrayList is slower with manipulation because it might require bit-shifts in memory (if an object is deleted for example).
  - LinkedList is faster for manipulation because it uses doubly linked lists to store data.
  - ArrayList can act only as a list because it only implements the list interface alone.
  - LinkedList can act as both a list and a queue because it implements both List and Deque.
  - ArrayList is better for storing and accessing data.
  - LinkedList is better for manipulating data.

100. How do you insert elements in a Map?
  - Invoke the maps put method.

101. Different between Exception and Error?
  - Exceptions and errors are both catchable but unlike exceptions catching an error can’t save your application.
  - Errors are references to larger issues that require termination of the application, there is nothing you can do to keep your application running once an error is thrown while exceptions can be caught and handled to keep your application running.

102. What is a Singleton?
- A singleton is an object that is very large or very complex.
- Because of this it is inefficient to have multiple instances of these objects in an application
- So to solve this you create one instance and pass it around your application to the methods that need it.

103. What is the IS-A rule?
- IS-A refers to an inheritance association, Objects that inherit from other objects have an is-a relationship with those objects (a lab IS-A dog).

104. Where are variable references stored?
- Depends what type of variables they are, and their scope, local variables or variables with block scope are stored in the stack.
- Objects and their instance variables are stored in the heap.

105. When is an object ready for garbage collection?
- An object is ready for garbage collection when there's no longer any reference to it in memory.
- This typically happens in when the object goes out of scope.

106. What makes the String class special?
- Strings can be created with literals
- Strings created with literals are stored in the string pool
- Strings created with new are stored in the heap

107. What is the difference between an Exception and a RuntimeException?
- Exceptions are the super class of runtime exceptions.
- Exceptions that aren't runtime exceptions are checked exceptions meaning that they must be handled at compile time (try-catch).
- Runtime exceptions themselves are unchecked exceptions you don't have to implement any logic to deal with them at compile time and they are discovered and handled at runtime.

108. What does the Iterable interface do?
- Iterable is a marker interface meaning that it marks the implementing class as iterable allowing it to be iterated through.
- Allows the class to use a for-each loop
- iterator() - returns an iterator for the object.

109. What is a Map?
- Map is an object that maps keys to values.
- Maps cannot have duplicate keys

- Each key can only map to at most one value.
    - `Put - places an object into the map`
    - `Get - returns an object from the map at a specific key`
    - `Remove - removes the mapping from the key if present.`
    - `containsKey - returns a boolean for whether or not a map holds a specific key value.`
    - `containsValue - returns true if the map contains a value else false`
    - `Size - returns the number of key-value mappings in the map.`
    - `Empty - returns true if the map is empty`
    - `Clear - removes all key value pairs from the map`

110. Can I sort a Map? (AHHHH)
    - Kinda
    - There's typically no need to sort a map because values are found by unique keys
    - It's possible to do if the keys or the values within them are sortable.

111. How do I start a Thread extending the Thread class
    - Invoke the start method

112. What is Starvation?
    - Part of the producer consumer problem
    - Occurs when two threads are sharing resources producer is producing resources and the consumer is consuming them
    - When the producer produces so fast that the consumer can't keep up it reaches a point where it can no longer produce causing it to starve on runtime
    - The same thing happens when a consumer is consuming faster than a producer can produce, it reaches a point where it isn't efficiently receiving data also causing starvation.

113. InputStream vs Reader
    - InputStream is a byte stream (used for serialized data).
    - Reader is a character stream (used for reading character data (files)).
    - An InputStream can be changed to a reader using an InputStreamReader class

114. When is a class fully synchronized?
    - When the parts of the program that modify the fields of the class are inside a synchronized block.
    - synchronized(MyClass.class) allows only one thread in at a time.
    - synchronized(this) allows one thread per instance.

115. Can an Interface have variables? What are they?
    - Yes but they are implicitly static and final

116. What is an Abstract class?

- A class that cannot be instantiated.

117. What are default methods in an interface?
    - A method which is declared as default in an interface can have an implementation in the interface.

118. What's the first line in a constructor?
    - super()

119. What is constructor chaining?
    - The process of calling one constructor from within another constructor

120. What is a short circuit operator?
    - In java there are two (&& and ||)
    - && will evaluate to false if the first half of the expression evaluates to false
    - || will evaluate to true if the first half of the expression evaluates to true

121. Where are Strings stored in memory?
    - Depends on how they're declared, the JVM however does create a special section of memory in the heap for strings called the string pool.
    - Using new creates a string object that's stored in the heap with other objects
    - When a string is declared using string literals it is added to the string pool.

122. What is hashCode() for?
    - `Objects.hash(Object... values)` should be used in cases when you want a hash of a sequence of objects, e.g. when defining your own `hashCode` method and want a simply-coded hash for multiple values that make up the identity of your object.
    - `Objects.hashCode(Object o)` should be used when you want the hash of a single object, without throwing if the object is null.
    - `Object::hashCode()` should be used when you want the hash of a single object, and will throw an exception if the object is null.

123. What's the parent of all exceptions?
    - Throwable
        - Exception
            - RuntimeException
        - Error

124. Can I catch an error? Does it make sense?
    - Errors do inherit from the throwable class so yes they can be caught
    - However it doesn't make sense to catch them because there's not typically anything you can do to handle an error.

125. Is there any case the finally block won't execute?
- Typically no but there are some extraordinary times where it is possible.
- If for some reason the exit method is invoked.
- An error is thrown by the application causing the program to terminate.
- Power failure.

126. Array vs ArrayList
- ArrayList - dynamically sized, takes objects, Collection

127. How do I insert elements in a map?
- put

128. What is a Factory?
- A large class that's used to instantiate objects from a particular class
- Typically a singleton because of its complexity
- Used to instantiate classes with numerous and complex constructors.

129. How do I make an object Serializable?
- Implement the Serializable Interface
- Serializable is a marker interface (has no methods)

JUnit

130. What is your experience with JUnit?
- Test Driven Development
- JUnit 4-5
- Mocking with Mockito

131. Can you test your entire application with JUnit?
- (You would have to test all units of it [methods], but we normally separate this testing in different test cases (specific functionality) with multiple tests).

132. What does a Unit refer to?
- The smallest testable subset of an application. In most languages that's functions or methods.

133. Annotations.
- Test - Denotes a method as a test.
- Before - Runs before every test.
- BeforeClass - Runs once before the class is run.
- After - Runs after each test.
- AfterClass - Runs once after the tests are done.
- Ignore - ignore a test temporarily

134. Difference between @BeforeClass and @Before?
- Before runs before every test

- Before class runs once before the class is run.

135. Why do we test software?
- Several reasons, Testing is very important in all phases of development.
    - Unit Testing
        - Make sure methods/functions are working properly.
        - Not meant to catch bugs
    - Regression Testing
        - New features don't break old ones
        - BDD - behavior driven
    - Integration Testing
        - Make sure new code is compatible with the existing code base
        - Complete system testing
    - Stress Testing
        - Make sure you application can perform well with large loads of traffic or large calculations.
    - Acceptance Testing
        - Make sure end users like the new state of the application.
        - Easy to understand
        - Are user needs fulfilled?
    - Compliance
        - Laws
    - Etc.
        - There's a lot of tests.

136. When does a test fail?
- When a test assertion evaluates to false (meaning the method didn't execute properly and there is a bug).
- When the test throws an error or exception.
- Or if the fail method is invoked resulting in aborting the test as a failure.

137. Different types of assert methods.
- assertTrue
- assertEquals
- assertFalse
- assertSame
- assertNotSame
- assertNull
- assertNotNull
- assertArrayEquals
- assertThat

138. Which version of JUnit have you used?
- JUnit 4-5

139. How do you test your application automatically?

- Depends what environment you're developing your application in, If you're using an IDE a lot of them have settings to run tests automatically, either when you run the application or after it completes a build process. Certain processes and applications can use hooks to run tests when a commit or push is made as well (Jenkins, husky).

SQL

140. What is an RDBMS?

- It stands for Relational Database Management system which is a program that allows you to create, update, and save data in a relational database.
- Some examples are Oracle, PostgreSQL, MySQL, SQLite.

141. What is an ERD?

- An ERD is a Entity Relationship Diagram, which is a diagram that maps out your database tables and the multiplicity and their relationships to each other. Creating an ERD during your design phase helps you visualize what you need to code when setting up the architecture of your database.

142. What is SQL?

- SQL stands for Structured Query Language and it's a programming language used to communicate with the data stored in relational database management system.
- SQL has sublanguages that are used specifically to either create/alter/drop tables, add/update/delete row entries in those tables, control what data you're retrieving from the tables, control user permissions, and handle transactions.

143. What are subtypes/sublanguages of SQL?

- Sublanguages are a group of commands in SQL used to work with specific parts of a database table.
- DDL, DML, DQL, DCL TCL

144. What is DDL? What are the keywords?

- It stands for Data Definition Language and it is used to define tables and set their properties. With it, you can create a table, alter the properties of the table and drop the table.
- The keywords are CREATE, ALTER, DROP, TRUNCATE

145. What is DML? What are the keywords?

- DML stands for Data Manipulation Language, The commands of SQL that are used to insert data into the database, modify the data of the database and to delete data from the database are collectively called as DML.
- You can insert rows into the table, update rows and delete rows
- The keywords are INSERT, UPDATE, DELETE

146. What is DCL? What are the keywords?
- DCL stands for data control language, and its the sub language that's used to grant and revoke users of their rights and permissions to a database
- The keywords are GRANT and REVOKE

147. What is DQL? What are the keywords?
- DQL stands for Data Query Language. The commands of SQL that are used to retrieve data from the database are collectively called as DQL. So all Select statements comes under DQL.
- The keyword with DQL is SELECT
- DQL is sometimes included in DML

148. What is TCL? What are the keywords?
- TCL stands for Transaction control language. Transaction Control Language commands are used to manage transactions in database. These are used to manage the changes made by DML-statements.
- The keywords are COMMIT, ROLLBACK, SAVEPOINT

149. What tools do you need to run an SQL Statement?
- Schema to write your query in??

150. What is a Database?
- It is a structured collection of data that is stored in a computer or server and can be accessed and manipulated in various ways.

151. What are Objects in SQL?
- Objects in SQL are any defined object that can store or reference data.
- Some objects may encompass other SQL objects
- The database itself is a SQL object itself that encompasses all other data objects
- The database contains Schema objects that contain table objects and views
- Columns are the smallest data object

152. What are Tables?
- Tables are data objects under a schema object and it holds columns that contain your data entries.

153. What are Views?
- Views are virtual tables based on the stored result-set of a SQL statement.
- A views fields are fields from one or more real tables in the database.
- You can store views in variables and perform operations to update or alter the view.

154. What are Triggers? When can it execute?
- A trigger is a database object that executes one or more SQL statements when a specified operation is done.

- You can define when a trigger fires when you define the trigger

155. What is a Schema?
- A schema is a collection of database objects associated with one particular database username.
- The username is called the schema owner, or the owner of the related group of objects.
- You can have multiple schemas in a database.
- Schema types
    - Snowflake - normalized; different relationships between tables
    - Star - not normalized; all tables point back to one fact table

156. What is an Index?
- An index is used to speed up the performance of queries.
- It does so by reducing the number of database data pages that have to be scanned/visited.
- In SQL Server, a clustered index determines the physical order of data in a table.
- There can only be one clustered index per table (the clustered index is the table).

157. What is a Cursor?
- A cursor is a temporary work area created in the system memory when a SQL statement is executed.
- A cursor contains information on a select statement and the rows accessed by it.
- This temporary work area is used to store the data retrieved from the database, and manipulate this data.
- Different from view because cursors are defined and used within the scope of a stored procedure.

158. What is a Sequence?
- A sequence is a user-defined, schema-bound object that generates a sequence of numeric values according to the specification with which the sequence was created.
- The sequence of numeric values is generated in an ascending order or descending order at a defined interval and can be configured to restart when exhausted.
- You can use a sequence to define a row by a unique value.

159. What is a Constraint? Give a few examples.
- Constraints are rules used to limit the type of data that can go into a table.
- Used to maintain the accuracy and integrity of the data inside the table.
- Can be divided into two types, column level and table level.
- Column level constraints limit only column data
- Table level constraints limit the whole table.
- Most used constraints include:
    - NOT NULL
    - UNIQUE

    - PRIMARY KEY
    - FOREIGN KEY
    - CHECK
    - DEFAULT

160. What are the different relationships in SQL?
- One-to-one - both tables can have only one record on either side of the relationship.
- One-to-many - one table can only have one record on its side of the relationship while others can have many records of the relationship
- Many-to-many - both sides can have many records on either side of the relationship.

161. What is a Primary Key?
- A primary key is a field in a table which uniquely identifies each row/record in a database table.
- Primary key columns cannot be null.
- A table can only have one primary key, which may consist of single or multiple fields.

162. What is a Unique Key?
- Also called a unique constraint.
- A unique constraint can be used to insure that there is no duplicate data in the rows of the table.

163. What is a Foreign Key?
- A foreign key is a column (or columns) that references a column that is the primary key of another table.
- The purpose of the foreign key is to insure referential integrity of the data.

164. What are the different types of Joins?
- Inner Join - Returns data that occurs in both tables.
- Left Join - Returns the data that occurs in the left table and the values that matched records in the right table.
- Right Join - Returns the data from the right table and the matching data from the left table
- Full Outer Join - returns all records when there is a match in either table.

165. What are the set operators?
- Set operators combine the results of two component queries into a single result.
- Queries containing set operators are called compound queries.
- Include:
    - UNION
    - UNION ALL
    - INTERSECT
    - MINUS

166. What keyword does Oracle use? What is it in Microsoft?

- ????????????????

167. What are the Transaction Properties?

- ACID.
- Atomicity.
- Consistency.
- Isolation.
- Durability.

168. What is Atomicity?

- Atomicity - each transaction is all or nothing, if any part of the transaction fails the whole thing fails, and then the database is rolled back to its last consistent state.

169. What is Consistency?

- Consistency - ensures that any transaction will bring the database from one valid state to another. (does not protect for correctness of transaction)

170. What is Isolation?

- Isolation - ensures that the concurrent execution of transactions results in a system state that would be obtained if transactions were used sequentially. One transaction has nothing to do with any of the others.

171. What is Durability?

- Durability - ensures that once a transaction has been committed, it will remain committed even in the event of power loss, crashes, or errors. Once a sequence of statements execute the results need to be stored permanently in non-volatile memory to defend against these faults.

172. What are the different Isolation Levels? What anomalies does each allow?

- Read Uncommitted - allows all anomalies.
- Read Committed - prevents dirty reads, suffers non-repeatable and phantom reads.
- Repeatable Read - prevents non-repeatable reads, suffers phantom reads.
- Serialized - prevents all anomalies including phantom reads

173. What is a Dirty Read?

- A transaction anomaly that occurs when a transaction is allowed to read data from a row that has been modified by another running transaction that has not been committed yet.

174. What is a Non-Repeatable Read?

- A transaction anomaly that occurs when data is read twice in the same transaction and returns different values because another transaction has modified the data between the reads. (more about update problems)

175. What is a Phantom Read?
    - Data getting changed in a transaction by other transactions in which rows are added or removed resulting in different result sets. (more about insert/delete problems)

176. What is Referential Integrity?
    - A concept that states that when data references another source of data, that that data exists.
    - In SQL data sources are tables, and foreign keys are the external references.
    - In SQL you can't delete a value from a table if it's referenced from other tables, this would cause the loss of referential integrity otherwise.

177. What is Normalization? What is it used for?
    - Normalization is the process of organizing the database into different forms each with their own sets of rules.
    - Makes the database more consistent and safe.
    - There are several normal forms.
    - Minimize data redundancy, reduce duplicate data

178. What is 1NF?
    - First Normal Form
    - The lowest level of normalization.
    - To be in 1NF a relation of the database is in first normal form if the domain of each attribute contains atomic values.
    - This means that fields in the rows of the database can't be broken into smaller more logical parts and contains only one of these parts.
    - Example could be street addresses :
        - Use Street, city, state.
        - Instead of address.
179. What is 2NF?
    - Second Normal Form, one level of normalization higher than 1NF.
    - To be in 2NF a relation of the database must first be in 1NF.
    - To be in 2NF a relation must have all of it's attributes dependent on the relations primary key.

180. What is 3NF?
    - Third Normal Form, the next step form 2NF.
    - To be in 3NF the relation must first be in 2NF.
    - To be in 3NF the relation's attributes must not be determined by non-prime attributes
    - "All columns must depend on the whole key if they aren't part of the key."

181. What is Transitive Dependency?
    - A transitive dependency in a database is an indirect relationship between values in the same table where one value is determined by another value.

182. What is an Aggregate Function?

- An aggregate function is a function where the values of multiple rows are grouped together to form a single value of more significant meaning or measurement.
- An aggregate function takes data and returns an interpolation from that dataset.
- Examples:
-
    - average()count()

183. What is a Scalar Function?

- A scalar function is a function that takes data and modifies it, it doesn't do any calculations with it, just altering the look or returning metadata of the query.
- Good examples are:
    - Upper case
    - Lower case
    - Mid
    - Len
    - Round
    - Now
    - format

184. WHERE vs HAVING

- A where clause is used to filter records form a result, the filter happens before any query is done.
- Having clause is used to filter an already existing group, having filters results from a completed query.

185. GROUP BY vs ORDER BY

- Order by alters the order in which items are returned, essentially just sorting the result set.
- Group by will aggregate records by the specified columns which allows you to perform aggregation functions on non-grouped columns.

186. What is a Subquery?

- Also called an inner or nested query
- A query within another SQL query and embedded within the WHERE clause.
- Used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

187. Functions vs Stored Procedures

- Functions are defined and must return some value.
- Functions can only have input parameters with predefined output parameters.
- Stored Procedures are just a series of steps that are taken at a given point.
- Procedures can have input and output parameters.

188. What is OLAP?
- Stands for Online Analytical Processing.
- A data analytics tool.

189. What is OLTP?
- Stands for Online Transaction Processing.
- A transactional tool.

190. What is JDBC?
- JDBC - Java Database Connectivity.
- An API specification for connecting java programs to popular databases
- Done using a driver for the particular dialect.

191. What do you need to get a JDBC Connection?
- First you must have a configured driver.
- DriverManager loads the specified driver and gets a connection to the database.
- DataSource is an interface that's prefered to DriverManager because it allows some transparency for some parts of the datasource.

192. What are the steps to setup a JDBC Connection?
- Import JDBC packages
- Load and register the driver
- Try with resources to get a connection from the driver manager or a class abstracting it.
- Create a statement object to perform a query.
- Execute the statement and retrieve the result set.

193. What is DAO?
- Stands for Data Access Object
- The interface for your database
- Establishes connections and executes queries sent to it.
- Passes result set.

194. What are the different types of Views?
- Standard views?
    - Combines data from one or more tables through a query.
- Indexed views?
    - A view that has been computed and stored.
    - A view is indexed by creating a unique clustered index on it.
    - Dramatically increases performance on some types of queries.
- Partitioned Views?
    - Joins horizontally partitioned data from a set of member tables across one or more servers.

195. What are the different types of Cursors?

- Static
    - Populates the result set at the time of cursor creation and query results is cached for the lifetime of the cursor.
    - Can move forward and backward.
    - Changes made with UPDATE, INSERT, or DELETE are not reflected in a static cursor

- Dynamic
    - Allows you to see updation, deletion, and insertion in the data source while the cursor is open. Hence a dynamic cursor is sensitive to changes to the data source.

- Forward only
    - The fastest among all cursors but it doesn't support backward scrolling.
    - You can update, delete data using Forward only cursor.
    - It is sensitive to changes in the original data source.
    - There are different types of forward only cursors.
        - Forward only keyset.
        - Forward only static.
        - Forward only fast forward.

- Keyset driven
    - A keyset driven cursor is controlled by a set of unique identifiers as the keys in the keyset.
    - The keyset depends on all the rows that qualified the select statement at the time the cursor was opened.
    - A keyset driven cursor is sensitive to any changes to the data source and supports update and delete operations.
    - They are also scrollable.

196. What are the different types of Indexes?
- Unique - Guarantees unique values for the column(s) included in the index.

- Covering - includes all of the columns that are used in a particular query(set of queries), allowing the database to use only the index and not actually have to look at table data to retrieve the results.

- Clustered - this is a way in which the actual data is ordered on the disk, which means if a query uses the clustered index for looking up the values, it does not have to take the additional step of looking up the actual table row for any data not included in the index.

197. Inner Join vs Intersect
- Intersect is used to retrieve the common records from both the left and the right query of the intersect operator.

- Intersect does all columns where as inner join does only the specified columns.
- Despite this the intersect operator returns almost the same result as an inner join a lot of the time.

198. What are the C.R.U.D operations?
- Stands for Create, Read, Update, and Delete.
- They are the four basic functions of persistent storage.

199. What is a nested query?
- See subquery.

200. Truncate vs Delete.
- Truncate is not transactional because it can't be rolled back.
- Truncate can modify the object storage attributes so it's not ordinary DML statement.
- Different dialects might classify it's sublanguage differently
- Delete removes the contents of a column.
- Truncate marks the columns for deallocation.

201. DATE vs TIMESTAMP.
- A few main differences
    - Timestamp converts from current time to UTC and back, while Datetime doesn't do any time conversion
    - Timestamp differs by time zone while datetime will remain constant.
    - Timestamp can also be indexed while datetime cannot.
    - Queries with datetime cannot be cached while queries with timestamp will be cached.

202. Can I do a subquery in an INSERT statement?
- Yes
- `insert into prices (group, id, price) select 7, articleId, 1.50 from article where name like 'ABC%';`

203. What is a Cursor? What can I use it for?
- A cursor is a temporary work area created in the system memory when a SQL statement is executed.
- A cursor contains information on a select statement and the rows of data accessed by it.
- This temporary work area is used to store data retrieved from the database, and manipulation of this data.

204. Which are the main interfaces of JDBC?
- Connection
- Statement
- Prepared statement
- Callable statement
- Result Set

205. ALTER vs UPDATE.
- Alter is a table scope command (DDL)
- Update is row scoped (DML)
- Alter table is altering the number or attributes of the columns and the functions in the table.
- Update is altering the data in the rows of the table.

206. CLOB vs BLOB.
- BLOB: Variable-length binary large object string that can be up to 2GB long
    - Primarily intended to hold non-traditional data such as voice or mixed media.
    - BLOB strings are not associated with a character set.

- CLOB: Variable-length character large object string that can be up to 2GB long.
    - Can store single-byte character strings or multibyte, character-based data.
    - Considered a character string.

207. Aggregate Function.
- DUPLICATE SEE 182.

208. FULL JOIN vs UNION.
- Union combines the results of two or more queries into a single result set that includes the rows that belong to all queries in the union.
- Full Join - joins everything! Combines the results of both left and right outer joins. The joined table will contain all records from both tables and fill in NULLs for missing matches on either side. Needs at least one match to join.

209. Can I do a subquery in the FROM clause of a SELECT statement? Why?
- Yes
- The result of the subquery in the from clause is a view, that view becomes the table from which the query using the from will select out of as if it were a real database table.

210. What is the Read Committed Isolation level?
- The second isolation level.
- Protects from dirty reads.

211. When can it execute?
- QUE??

212. CREATE vs INSERT.
- Create is a DDL command that is used to make a table, procedures, and functions.
- INSERT is a DML command that is used to add a row to a table.

213. What does a DATE subtraction return?

- If you’re using DATEDIFF() then it depends what you specify
    - Can get the difference in years, quarters, months, days, weeks, hours, minutes, seconds, milliseconds.

214. LEFT JOIN vs MINUS.
- MINUS is a keyword in some sql dialects and it works like A-B in sets.
- MINUS would return everything in set A that are not found in BOTH A and B.
- LEFT JOIN = (A MINUS B) + (A INNER JOIN B).

215. Can I do a subquery in an UPDATE statement?
- Yes you can.
- Could go in a WHERE clause.
-
```
UPDATE Production.Product
SET ListPrice = ListPrice * 2
WHERE ProductID IN
   (SELECT ProductID
    FROM Purchasing.ProductVendor
    WHERE BusinessEntityID = 1540);
```

216. What is the Serializable Isolation level?
- The highest of the transaction isolation levels.
- Protects against all of the read anomalies
    - Dirty read
    - Non-repeatable read
    - Phantom read

217. Stored Procedure vs UDF.
- UDF = User Defined Functions
- Functions can only have input parameters and must return a value
- Procedures can have input and output parameters and don’t have to return anything.

218. Statement vs PreparedStatement.
- Statement is vulnerable to SQL injection in which an attacker could finish a query logically and end it with a semicolon and start another query that could potentially harm the database (Bobby Tables).
- Prepared statements are not vulnerable to sql injection they preload the database with a query and then send in expected values after which are cast to strings and placed into the query.

219. DROP vs DELETE.
- DROP is a DDL command that removes it’s arguments from the table or removes the whole table from the database.
- DELETE is a DML command that removes the values from the columns matching it’s arguments.

220. VARCHAR vs VARCHAR2.

- VARCHAR - Is reserved in some dialects (Oracle) to support the differentiation between null and empty string.
- VARCHAR2 - does not distinguish between a null and an empty string.

221. What is a composite Primary Key?

- Composite Key or Composite Primary key, refers to cases where more than one column is used to specify the primary key of a table.
- In such cases, all foreign keys will also need to include all the columns in the composite key.
- Note that the columns that make up a composite key can be of different data types.

222. What is a SELF JOIN?

- A self-join is a query in which a table is joined (compared) to itself. Self joins are used to compare values in a column with other values in the same column in the same table.
- A practical use for self joins is obtaining running counts and running totals in a SQL query.

223. UNION vs UNION ALL.

- Union all command is equal to the union command, except that union all selects all values.
- The difference between them is that union all will not eliminate duplicate rows, instead it pulls all rows from all tables fitting the query specifics and combines them into a table.

224. Can I do a subquery in a DELETE statement?

- Yes
- Can also go in a where clause.
- Or a from clause.

225. Connection vs DriverManager.

- Connection is the actual connection to the database where you can load statements
- DriverManager loads the driver needed to connect to the database (if configured correctly) and generates a connection object on the java side.

226. INNER JOIN vs OUTER JOIN.

- Inner join returns the matching data from two tables.
- Outer join returns the inner join along with the rows in the table which couldn’t be matched.

227. What is a theta join?

- Theta joins allow for arbitrary comparison relationships (>, <, >=, <=, =).
- Theta joins using an equality operator are called equijoin.
- A natural join is a equijoin on attributes that have the same name in each relationship.

228. What is a natural join?

- A join operation that creates an implicit join clause for you based on the common columns in the two tables being joined.
- Common columns are columns that have the same name in both tables.
- Can be an inner join, left outer join, or a right outer join.

229. All types of joins. You can get asked differences between any of them.

- Inner
- Outer
- Left
- Right
- Natural
- Theta

230. What is an index?

- An on-disk structure associated with a table to speed up row retrieval (table of contents).

231. Clustered vs Unclustered index.

- Clustered indexes sort and store the data rows in the table or view based on their key values.
- Nonclustered indexes have a structure separate from the data rows which contains the nonclustered index key values and each key value has a pointer to the data row.

232. What is a view?

- DUPLICATE

233. Regular view vs Materialized view.

- Views are virtual only and run the query definition each time they are accessed.
- Materialized views are disk based and are updated periodically based on query definition.

234. Unary, Binary and Ternary relations.

235. All sections of the SELECT statement.

- SELECT (contents) FROM (table) WHERE (constraints);

=== SPECIALTY QUESTIONS ===

236. How would you get the minimum salary from every department? (All depts are in ONE table)

-
  ```
  SELECT MIN(salary) AS "Lowest salary"
  FROM employees
  GROUP BY department;
  ```

237. - How would you get the top 5 salaries from every department? (All departments are in ONE table)

- SELECT * FROM employees
  ORDER BY salary DESC
  LIMIT 5

HTML, CSS, and Bootstrap

238. What is HTML?

- Hypertext Markup Language
- Defines how a web page is laid out.
- Uses tags
- Know the format of tags.

239. What is the extension to an HTML file?

- .html

240. What are some common tags?

- Html
- Head
- Body
- P
- Div
- Span
- h1 - h6
- A
- Style
- Script
- Form
- Ul
- Ol
- Dl
- Li
- Dt
- Dd
- img
- etc.

241. What is the first tag in EVERY html file?

- Html

242. What is CSS?

- Cascading style sheet

- Defines how the elements on the page will look.

243. HTML vs CSS
    - Layout vs Look

244. What are the different forms of CSS?
    - If this question is referring to the different ways in which css can be applied to an html document then:
        - External - In a separate file that's linked.
        - Internal - At the top of the document in style tags.
        - Inline - as an attribute on the elements themselves.

245. What is iFrame?
    - An HTML document embedded inside another HTML document on a website.
    - The IFrame element is used to insert content from another source, like an ad on a web page.

246. What is DOM?
    - Document Object Model
    - Represents the html page structure as nodes and objects. That way programming languages can connect to the page.

247. What is Bootstrap?
    - A front-end, open source toolkit used to speed up development with HTML, CSS, and JavaScript.
    - Built for mobile first web-development
    - Using a grid system.

248. What are Meta Tags?
    - Provide metadata about the HTML document.
    - Metadata will not be displayed on the page, but will be machine parsable.
    - Meta elements are typically used to specify page description, keywords, author of the document, last modified, and other metadata.

249. What is ASCII?
    - American standard code information interchange
    - A character encoding standard for electric communication.

250. What is ANSI?
    - American National Standards Institute.
    - Creates standards for the computer industry.
    - Sets standards throughout a wide range of technical areas from electrical specifications to communication protocols.

251. What is UTF-8?
- Unicode Transformation Format
- 8 means it uses 8 bit blocks to represent a character
- A character encoding that can contain any unicode characters.

252. What tag would you use to create tables?
- The table tag
- Tr is table row
- Th is table head
- Td is table data (column data)

253. What tag would you use to create list?
- Three different lists
    - Ordered list
        - Ol
        - li
    - Unordered list
        - Ul
        - li
    - Description list
        - Dl
        - Dt
        - Dd

254. What tag would you use to create forms?
- Form tag creates a form
- Input specifies the different form inputs
- Button or input type=”button” can be used to submit.
- If there’s only one button on the form it will default to the submit button.

255. What does attribute "required" do?
- Used on input to specify that it must be filled out for the form to submit.

256. What does attribute 'target="_blank"' do?
- Anchor links may have the target attribute which controls what happens when the anchor is clicked.
- If the target is _blank this tells the browser to open a new window (or tab depending on user settings) when the link is clicked.

257. What do elements and attributes need to be to achieve XHTML?
- Elements
    - XHTML elements must be **properly nested**
    - XHTML elements must always be **closed**
    - XHTML elements must be in **lowercase**

        - XHTML documents must have **one root element**
    - Attributes
        - Attribute names must be in **lower case**
        - Attribute values must be **quoted**
        - Attribute minimization is **forbidden**

258. What tag do you use to create an image?
    - Img
    - Requires a src attribute which is where the image is from
    - Also requires an alt attribute = specifies the alternate text for the image if it couldn't be loaded.

259. JavaScript and AJAX
    - JavaScript is an object oriented interpreted scripting language
        - Used on web pages for DOM manipulation to make the pages more interactive.
    - AJAX
        - Stands for Asynchronous JavaScript and XML.
        - A new technique for creating better, faster, and more interactive web applications using XML, HTML, CSS, and JavaScript.

260. What is JavaScript?
    - An object oriented computer programming language commonly used to create interactive effects within web browsers.

261. What is OBP?
    - Represents Object Based Programming.
    - Basically a superset of OOP.

262. What is DOM Manipulation?
    - The altering of the DOM while the application is running.
    - You can use this to add remove and change the way elements are laid out and how they look on the page.

263. Is JavaScript loosely typed or strict typed?
    - JavaScript is loosely typed you don't declare the data types of variables explicitly.

264. What kind of compiler does JavaScript use/have?
    - Javascript is interpreted not compiled
    - The script is interpreted by other applications like web-browsers or npm.

265. Is JavaScript case-sensitive?
    - Yes javascript is case sensitive.

266. What is Prototypal Inheritance?
- Everything in javascript is an object
- Each object has a private property which holds a link to another object called its prototype.
- When you construct an object the prototype chain links all the way up to Object the parent of everything in javascript to build the object.

267. Is JavaScript Pass-By-Value or Pass-By-Reference?
- For primitives javascript is pass by value.
- Javascript also uses call by sharing for objects.
- Call by sharing is similar to pass by reference in that the function is able to modify the same mutable object but unlike pass by reference isn't able to assign directly over it.

268. What is Type Coercion?
- Type coercion is a property of javascript to attempt to compare values of different types because it doesn't have strict types
- When == is done it will try to change the type of the object for a temporary comparison.
    - A good example is (1 == '1' will be true, even though one is a string and one is a number).
- This can be avoided by using === which won't try to do any casting.

269. What are the variable scopes in JavaScript?
- Global and local scope

270. What are functions in JavaScript?
- Functions are objects to start
- A function object defines a series of tasks and/or calculates a value.
- Must be defined in the scope you wish to call it in.

271. What are the types of functions?
- Function declaration - create a function but don't assign it to a variable
- Function expression - create a function and assign it to a variable

272. What is an anonymous function?
- A function that is created at runtime.
- The function operator can be used anywhere that it's valid to use an expression.
- Called "anonymous" because the function was not "named"

273. What is a self-invoking function?
- A self-invoking function is started automatically without being called, this is done right after it's created.
- This is used for avoiding naming conflict as well as for achieving encapsulation.
- Variables or declared objects are not accessible outside this function.

274. What is Hoisting?
- Hoisting in javascript is when the interpreter moves all variable and function declarations to the top of the current scope.
- Only the declarations are hoisted, assignments are left where they are.

275. What are Closures?
- A closure is an inner function that has access to the outer functions variables.
- A closure has three scope chains: its own scope, outer functions, and globals.

276. JSON vs JS Object
- JSON is JavaScript Object Notation
- JSON is a data interchange format, a way to pass structured information between languages.
- JSON is formatted similarly to how a JavaScript Object is formatted which makes it very easy to parse and build objects from.

277. What does the keyword "STRICT" mean?
- New as of ES5.
- Is a literal expression indicates that the code should be executed in “strict mode”
- In strict mode you can’t do a lot of things that you can do outside of un-strict mode, for example using undeclared variables.

278. What does it mean to be "Falsy"?
- Javascript doesn’t use true and false when doing logical comparisons JavaScript uses type coercion, truthy is not just the value true, it’s values that could logically represent a true value.
- Truthy values could be true, empty object, empty array, empty function, positive numbers, and non empty strings
- Falsy values could be false, null, undefined, NaN, zero, and empty strings.

279. What are the data types in JavaScript? (6)
- String
- Number
- Boolean
- Undefined
- Arrays
- Objects
- Symbols
- Function

280. == vs ===
- == uses type coercion meaning that it will try to see if they hold the same value by casting them to the same type, i.e. “1” = 1

281. What is Bubbling?
- Event bubbling is a form of event propagation.
- Event bubbling propagates events from children to parents.
- An event gets fired and is handled by a child element, from there it propagates up to parents with handlers for the event.

282. What is Capturing?
- Capturing is another form of event propagation.
- Capturing works the opposite of bubbling:
    - When an event is fired and parent and children elements have handlers registered the parent’s handler fires first and then the event trickles down to the child.

283. Are JS objects mutable or immutable?
- In javascript, only objects and arrays are mutable.
- Primitives are immutable.

284. What are events?
- HTML events are actions generated with user interaction with an element or elements.
- Good examples are click events on buttons, page loading, resizing the window, clicking links, hovering, etc.
- When a user does this with the mouse or keyboard the element can throw events to the javascript to work with or call predefined functions.

285. What is a Listener?
- A listener is an object that waits for events or actions to occur
- In javascript the most used is the Event Listener, when an event occurs the element with an event listener attached will call or execute some type of function to make the web page more interactive.

286. What are the JSON conversion types?
- JSON to DataObject conversions:
    - String JSON can convert to string, long if long value , double if double value, boolean if true or false, date/datetime if is a date value.
    - Long to string, long, double, or date if value is a date.
    - Double to string, long if long value, double.
    - Boolean to string and boolean.

287. What is AJAX?
- DUPLICATE

288. What is HTTP?
- HyperText Transfer Protocol.

- The underlying protocol used by the world wide web.
- Defines how messages are formatted and transmitted, and what actions servers and browsers should take in response to various commands.

289. What are the Xml Http Request methods?
- New XMLHttpRequest() - creates a new XMLHttpRequest object.
- abort() - cancels the current request.
- getAllResponseHeaders() - returns header information.
- getResponseHeader() - returns specific header information.
- open(method, url, async, user, psw) - specifies a request.
- send()/send(string) - sends the request to the server.
- setRequestHeader() - adds a label/value pair to the header to be sent.

290. What are other commonly used XML HTTP methods?
- Look above…

291. What are the Ready States in HTTP?
- 0: request not initialized
- 1: server connection established
- 2: request received
- 3: processing request
- 4: request finished and response is ready

292. What are the HTTP Methods?
- GET - requests a representation of the specified resource. Get should only be used to get data.
- HEAD - asks for a response identical to that of a GET request, but without the response body.
- POST - used to submit an entity to the specified resource, often causing a change in state or side effects on the server.
- PUT - replaces all current representations of the target resource with the request payload.
- DELETE - deletes the specified resource.
- CONNECT - establishes a tunnel to the server identified by the target resource.
- OPTIONS - used to describe the communication options for the target resource.
- TRACE - performs a message loop-back test along the path to the target resource.
- PATCH - used to apply partial modifications to a resource.

293. What are the HTTP Statuses?
- 100 - info.
- 200 - success.
- 300 - redirect.
- 400 - bad request.
- 500 - server error.

294. What are the properties?
- WHAT?

295. What is a DOM?
- DUPLICATE

296. What is a Callback Function?
- A callback function is a function placed into a parameter of another function to make sure that certain steps are done when that function ends.

297. What is the AJAX workflow?
- What is this asking??

298. GET vs POST
- GET requests can be cached
- GET requests remain in the browser history
- GET requests can be bookmarked
- GET requests should never be used when dealing with sensitive data
- GET requests have length restrictions
- GET requests should be used only to retrieve data
- POST requests are never cached
- POST requests do not remain in the browser history
- POST requests cannot be bookmarked
- POST requests have no restrictions on data length
- `POST` is NOT idempotent. So if you retry the request N times, you will end up having N resources with N different URIs created on server.
- PUT method is idempotent. So if you send retry a request multiple times, that should be equivalent to single request modification
- Idempotent - method can be called many times without different outcome

Servlets

299. What is J2EE?
- J2EE stands for Java Platform Enterprise Edition. It is a collection of Java APIs used to write server-side applications.

300. What is TCP/IP?
- Stands for Transmission Control Protocol/Internet Protocol.
- It is a communication protocol used to interconnect network devices on the internet.
- It is a suite that encompasses a number of different protocols for different purpose and needs

301. What are the different content types?
- Content types is also known MIME (Multipurpose Internet Mail Extension) Type
- Commonly used content types:

        - text/html
        - text/plain
        - application/jar
        - application/pdf
        - images/jpeg
        - images/png
        - images/gif
        - audio/mp3
        - video/mp4

302. What is a servlet?
    - Servlet is a technology used to create web applications; it is used to respond to incoming requests.

303. What is the Lifecycle of a Servlet?
    1) Servlet class is loaded
        a) Loaded when the first request for the servlet is received by the web container
        b) Load on startup
    2) Servlet instance is created
        a) Created after the servlet class is loaded.
        b) Created only once in the servlet life cycle.
    3) Init method is invoked
        a) Web container calls the init method once after creating the servlet instance.
        b) Used to initialize the servlet.
    4) Service method is invoked
        a) Called every time a request is received for the servlet
    5) Destroy method is invoked
        a) Called before removing the servlet instance from the service.
        b) Used to clean up any resources

304. App Server vs Web Server
    - A Web server exclusively handles HTTP requests, whereas an application server serves business logic to application programs through any number of protocols.
    - Web server is responsible for serving static content over HTTP while Application server is responsible for serving dynamic content, managing EJB, lookup over JNDI, and application security.

305. What is XML?
    - Stands for extensible markup language
    - It defines set of rules for encoding documents in a format that is both human-readable and machine readable

306. What is Well-Formed XML?
    - Refers to an XML document that satisfies certain rules specified by the W3C
        - documents with correct syntax
    - A Well-Formed XML requires that:
        - Content be defined
        - Content be delimited with beginning and end tag
        - Content be properly nested

    - There can be only one root tag

307. What are the rules for an element in XML?
    - Element names are case-sensitive
    - Element names must start with a letter or underscore
    - Element names cannot start with the letters xml (or XML, or Xml, etc)
    - Element names can contain letters, digits, hyphens, underscores, and periods
    - Element names cannot contain spaces
308. Element vs Attribute
    - Elements can be viewed as containers that stores text, elements, etc…
    - Attributes defines the property of an element
309. What is XMLNS?
    - Means XML namespaces
    - Is used to avoid element name conflicts
310. What is DTD?
    - DTD stands for Document Type Definition.
    - This defines the elements that may be included in the document, what attributes the elements have, and the ordering and nesting of the elements.
    - Documents validated against DTD is considered to be “Well-Formed” and “Valid”
311. What is an XML Schema?
    - Describe the structure of an XML document
    - Provides a more powerful alternative to DTD
312. What is XSL?
    - Stands for Extensible Stylesheet Language
    - Describes how XML elements should be displayed
    - Consist of three parts:
        - XSLT - language for transforming XML documents
        - XPath - language for navigating in XML documents
        - XSL-FO - Used for formatting XML documents
313. What are the types of XSL?
    - ?????????????????
314. What is XSLT?
    - Stands for Extensible Stylesheet Language Transformations
    - Used for transforming XML documents into other XML documents
        - Ex: HTML to XHTML
315. What is XPath?
    - Stands for XML Path Language
    - Used to navigate through elements and attributes in an XML document
    - Core component of XSLT
    - Defines part of the XML document
    - Provides powerful “path expressions”
        - Used to select node or list of nodes from an XML document
    - Provides library of standard functions for manipulating string, numbers, etc.
316. What is JAXB?
    - Stands for Java Architecture for XML Binding

- It is used to convert Java Object to XML and XML to Java Objects

317. What are the different types of XML parsing?
    - DOM parser
    - SAX parser
    - JDOM parser
    - StAX parser
    - XPath parser
    - DOM4J parser
318. What is Marshalling?
    - The process of transforming the memory representation of an object to a data format suitable for storage or transmission
319. What is Unmarshalling?
    - The process of transforming a representation of an object that was used for storage or transmission to a representation of the object that is executable.
320. What is a Front Controller?
    - A design pattern that represents a single controller that handles all incoming requests
321. What is a Dispatcher?
    - Dispatcher is part of the Front Controller Design Pattern
    - It is responsible for view management and navigation
322. What is MVC?
    - Represents Model, View, Controller
    - MVC is an architectural pattern that separates an application into three main components:
        - Model - all data related logic
        - View - all UI related logic
        - Controller - handles all business logic and incoming requests.
323. What is the deployment descriptor?
    - In J2EE, a deployment descriptor describes how a component should be deployed.
    - An example of a “deployment descriptor” is the web.xml.
324. What are the important tags you need to include in the web.xml?
    - web-app
    - servlet, servlet-name, servlet-class
    - servlet-mapping, servlet-name, url-pattern
    - init-param, param-name, param-value
    - context-param, param-name, param-value

325. What is the HttpServlet class? What methods does it provide?
    - It is a class that extends from the GenericServlet class and provides Http specific methods
    - Provided methods:
        - doGet
        - doPost
        - doPut
        - doDelete

        - service
        - etc...

326. What is the flow of client-to-server using a servlet?
    - The client sends requests through HTTP over TCP/IP to the HTTP server. The servlet will receive the HTTP requests and return a response back to the client.
327. What is a RequestDispatcher?
    - Is an interface that facilitates dispatching a requests to another resource.
    - The RequestDispatcher provides two methods:
        - Forward and Include
328. Forward() vs Include()
    - Forward() - forwards a request from a servlet to another resource
    - Include() - includes the contents of a resource into the response
329. Forward() vs SendRedirect()
    - Forward sends requests to other resources in the same server while SendRedirect sends requests to other resources in different servers
    - Forward is declared in the the RequestsDispatcher interface while SendRedirect is declared in HttpServletResponse
330. Context-Param vs Init-Param
    - Init-param is only accessible to the servlet that it is declared in while context-param is accessible to all servlets
331. What are the important tags you need to include within Context-Param and Init-Param?
    - <param-name> and <param-value>
332. Eager Loading vs Lazy Loading
    - Lazy loading - a servlet is not loaded when server is starting and load only when the servlet receives a request
    - Eager Loading - servlet loads while server is starting
        - A servlet can be set to load eagerly when you declare **<load-on-startup>**
333. Servlet Context vs Servlet Config
    - Servlet Config - are specified only for a particular servlet and is used for initialization purpose
    - Servlet Context - are specified for the entire application and is accessible by all servlets within that application
334. getServletConfig() vs getServletContext()
    - ???????????????
335. getParameter() vs getAttribute()
    - getParameter - returns the HTTP requests parameters passed from client to server
    - getAttribute - mostly for server-side usage only; filling a request with attributes that can be used within the same requests.
        - Ex: set an attributes within servlet and read it from a JSP
336. URL vs URI vs URN
    - URL (Uniform Resource Locator) - identifies the location of the resource and how to access it
    - URI (Uniform Resource Identifier) - identifies a resource by name, location, or both

- URN (Uniform Resource Name) - identifies a resource by name in a given namespace but does not specify how to obtain it

337. What are the method signatures of doGet() and doPost()?
    - HttpServletRequest req, HttpServletResponse res
338. Explain the communication between the Client and the Server in a JEE application.
    - The **client communicates** with the business tier running on the **Java EE server** either directly or, as in the case of a **client** running in a browser, by going through web pages or servlets running in the web tier
339. Servlet Hierarchy.
    - Interfaces
        - Servlet
        - ServletConfig
        - Serializations
    - GenericServlet
    - HttpServlet
    - UserDefinedServlet
340. Where do init, service and destroy come from in the hierarchy?
    - They come from the Servlet interface
341. Init, Service and Destroy, how many times do they execute?
    - The init and destroy method is called once
    - The service method is called for every requests the servlet receives
342. doGet vs doPost vs GET vs POST.
    - GET and POST are HTTP protocols while doGet and doPost are methods specific to Java Servlet
    - doGet is handles all GET requests while doPost handles all POST requests
    - GET HTTP requests supplies all required data in the URL while POST HTTP requests supplies all required data in the request/message body
343. Another name for web.xml?
    - Web application deployment descriptor
344. Sections and tags on the web.xml.
345. Servlet Context vs Servlet Config. How do you write it on the web.xml?
    - Servlet Context - <context-param>
    - Servlet Config - <init-param>
346. How do I declare a Servlet?
    - <servlet><servlet-name><servlet-class>
    - <servlet-mapping><servlet-name><url-pattern>
347. How does Error Handling work on JEE?
    - Handle the error on the server side with try-catch or throws
    - Return status codes to the front-end depending on errors or lack thereof.

348. How does the Servlet Container track sessions?
    - An HttpSession object can be created by invoking the requests getSession() method.
    - Sessions in servlets manage user sessions; used with JSPs and JSFs because they have views, meaning they serve as both the client and server; not with RESTful servlets

349. Where to forward from, Where to redirect from?

- ?????????

350. What is the service method signature on GenericServlet?

- Public abstract void service(ServletRequest req, ServletResponse res) throws ServletException, java.io.IOException.

351. Service method in HttpServlet.

- Protected void service(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException.

352. How many versions of init does HttpServlet have?

- 2 inits that are inherited from GenericServlet.
- init()
- init(ServletConfig config)

353. Method signature doGet.

- Protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException.

354. How to maintain user state?

- Cookies initialized when the user logs in.
- Send the session id from the cookie in HTTP requests.
- Access the HttpSession object from the request object using the getSession() method.

355. Front Controller.

- A design pattern for full-stack applications.
- Front-end sends a request to the back-end.
- Back-end has a single servlet that is the request dispatcher it receives all requests.
- When the dispatcher gets a request it consults the handler mapper which tells it by the request type url and other aspects what controller to send the request to.
- The dispatcher sends it to the appropriate controller who uses a service to call a DAO.
- The DAO sends a query to the database and receives the databases response which it sends back through the service.
- The service returns it to the controller who passes it to the dispatcher.
- If the request was for a page then the servlet would communicate with the view resolver to produce the correct view on the front-end.
- The dispatcher writes a response with the data in it to the response stream to the front-end producing the view or passing the data for manipulation or display.

356. What is a thread pool?

- Object pools are used in java for storage of objects that we want to use dynamically.
- The thread pool is one of these object pools

- Object pools controls the lifecycle of its resources.
- Many of the java.util.concurrent libraries use thread pools implementing thread pools of worker threads
- Worker threads are separate objects from Runnable and Callable tasks and are usually used for multiple tasks.

357. What are filters?
    - A filter in the context of java servlets is an object that performs filtering tasks on either the request to a resource or on the response from a resource. It's used to limit input and output from a resource to only requests and responses fitting a set of rules defined by the filter.

358. What is extended mapping?
    - ???????????

Angular 4 and Typescript

359. What is Typescript?
    - A very high level view
        - Typescript is a superset of javascript which primarily provides the option of static typing, classes, and interfaces.
        - Typescript is a transpiled language that is open source.
        - Typescript is compiled into javascript to be served on a webpage
        - Typescript adds a lot of benefits to what javascript is while still being familiar to javascript users
        - Benefits include:
            - Optional static typing and type inference.
            - Enhanced IDE support.
            - Strict null checks.
            - JavaScript interoperability.
        - Typescript also has many frameworks such as angular, Aurelia, Ionic, NativeScript to name a few, built on top of it for even more added functionality.

360. Is Typescript Loosely or Strongly Typed?
    - Strongly typed
    - Or loosely typed (any is a type)

361. What does the "let" keyword mean?
    - Let is similar to var in some respects but let forces local scoping that var can cause problems with.

362. What does the Arrow Function (=>) do?
    - Defines a lambda function.
    - Don't need to type "function"
    - Lexically captures the meaning of this.

- Lexically captures the meaning of arguments.

363. Compiling vs Transpiling?
- Transpiling is a special kind of compiling where instead of being converted into machine code it's compiled into another language.
- Compiling is when source code is converted into machine code.

364. What is Angular?
- Angular is a framework built on top of typescript using declarative templates, dependency injection, end to end tooling, and integrated best practices to speed up the development process.
- Angular defines components and uses dependency injection to carry out DOM manipulation for you.

365. AngularJS vs Angular4
- Angular JS is 1.0-1.6, Angular is 2+
- Both use dependency injection
- Angular JS is in JavaScript
- Angular is in TypeScript
- Decorators are unique to Angular2

366. What does it mean to be a Single Page Application
- A single page application (SPA) is a web application or web site that interacts with the user dynamically by rewriting the current page rather than loading a whole new page from a server.
- Interaction with the single page application often involves dynamic communication with the web server behind the scenes.

367. What is a Webpack?
- A Webpack is a module bundler. Takes care of bundling your application for deployment alongside a separate task runner.

368. What is minification?
- The process of removing unnecessary or redundant data without affecting how the resources is processed by the browser.
- Example would be removing of white space, comments, unused code, shorter variable and function names and so on.
- Used to compress source code files for deployment.

369. What is a module?
- In angular a module is a mechanism to group components, directives, pipes, and services that are related, in such a way that can be combined with other modules to create an application.
- Modules can be easily imported and exported for use.

370. What are components?

- Components are the most basic building block of a UI in an angular application.
- Everything in angular components and the application is represented as a tree of these components.

371. What are decorators?

- Decorators are a core concept of Angular 2+.
- The whole purpose of the decorator is to store metadata about a class, method, or property.
- The component decorator tells angular that a component is a component and then it knows what it needs to do with it.
- It’s more of an identifier than anything which sounds simple but it’s very important so that angular can differentiate components and their purposes.

372. What are some common decorators?

- @Component
- @NgModule
- @Input
- @Output
- @Inject

373. What is a directive?

- Directives allow you to attach behavior to elements in the DOM.
- Directives are identified with the @Directive decorator.
- Directives must be present in the declarations field of your NgModule for it to be usable by another directive, component, or application.

374. What is a Structural Directive?

- Must be annotated with @Directive decorator.
- Structural directives are responsible for HTML layout. They shape or reshape the DOM’s structure by adding, removing, or manipulating elements.
- Structural directives is how Angular does DOM manipulation.
- Actions of Structural directives also affect children elements.
- Examples are:
    - NgIf
    - NgFor
    - NgSwitch

375. What is a Attribute Directive?

- Attribute Directives change the appearance or behavior of a DOM element, component, or another directive.
- Attribute directives are used as attributes of elements
- Must be annotated with the @Directive decorator

- A good example is the NgStyle directive which is built-in to angular.

376. What is ngModel?
    - NgModel is an Angular Directive
    - Creates a FormControl instance form a domain model and binds it to a form control element.
    - The FormControl instance will track the value, user interaction, and validation status of the control and keep the view synced to the model.
    - Can be used by itself or as part of a larger form, if so it will also register itself with the form as a child control.
    - Often used to achieve two-way databinding.

377. What is a Dependency Injection?
    - Dependency Injection is a form of Inversion of Control, where you give up control to an outside entity. The outside entity provides you the instance of the dependency that you need, so that you don't worry about the implementation of your resource, but instead focus on the business logic you are constructing. Dependency Injection is beneficial because it keeps your classes loosely coupled and isolates test cases.

378. What is Data Binding?
    - In angular 4 data binding is how data manipulation in an angular component or web-page is done.
    - One-way data binding
        - When properties of the model get updated so does the UI or the opposite not both.
        - Can flow two ways from html doc to scripts or from scripts to html doc.
            - Doc to scripts
                - When the user changes a field on an webpage you can capture it quickly and easily with a one way data binding done in angular 4 with [name].
            - Scripts to doc
                - When you want to display the value of a variable in your script in the html doc but but the values is calculated or volatile meaning there's no guarantee you know what it is you could use a one way data binding from the script to the page.
                - This is done using interpolation represented by {{ varname }}.
    - Two-way data binding
        - Means that when properties of the model get updated, so does the UI and the reverse is true.
        - This is done in angular 4 using the ngModel directive.

379. What are the types of Data Bindings?

- One-way
- Two-way

380. How do you make a class a component?
  - Simply specify that a class is a component by annotating it with the @Component decorator.

381. How do you make asynchronous calls using Angular?
  - Assuming you're using a service with a predefined function that is returning an observable or a promise.
  - Make the call to the service and subscribe to it with the subscribe method.
  - Inside the subscribe method i would define a callback to execute when the service call is resolved and inside that you can store it in a variable or do any manipulation you want.

382. What is a Pipe?
  - A pipe takes in data as input, does some transformations, returns some output
  - Used for writing formatting tasks or other repetitive tasks.
  - That way you only have two write it once and then import it and make a call to it wherever you need it.

383. What is a Service?
  - Services are angular's way of fetching/storing data.
  - Angular has services so that components don't have to both fetch data and display/present it in the view.
  - Reduces code bloat.
  - Increases modularity.
  - Can use the same service in multiple components or applications.

384. What is a Binding?
  - Binding the function is a way of setting a function context.
    - .bind() when you want that function to later be called with a certain context, useful in events.
    - Use .call() or .apply() when you want to invoke the function immediately, and modify the context.
    - Call/apply call the function immediately, whereas bind returns a function that when later executed will have the correct context set for calling the original function. This way you can maintain context in async callbacks, and events.
    - The first argument in the bind method is the method that will be bound to this.
  - Data-binding
    - One-way and two-way binding
    - When the state of a field changes on the web page the value of the variable in the script is updated as well.

385. What is Interpolation?
- The populating of a placeholder with values from a string or variable to be viewed by the user.
- Done with {{ varname }}

386. What is an Observable?
- A collection that arrives over time.
- Often used to handle async data requests.

387. What is a Provider?
- A provider is an object or class that provides a component with data.
- In short terms it is a service that is registered to a component or module.
- Providers is a field or attribute in component or module decorators that tell angular what services to inject into the component.

388. What is an Injectable?
- A Decorator that tells angular that the class is available to the Injector for creation.
- Objects that are injectable can be created and injected into a component or service.

389. What is Routing? How do we implement it?
- Routing is the ability to send users from one page in an application to another page in the application and back.

390. What is CDN?
- ??????

391. What is the lifecycle of a component?
- The lifecycle of a component is managed by angular itself, creation, rendering, data-bound properties, and it has hooks for you to allow you to respond to these events
- The most used is probably ngOnInit which is generated for every component when using ng g c.

392. What are the Lifecycle Hooks?
- `ngOnChanges` - called when an input binding value changes
- `ngOnInit` - after the first `ngOnChanges`
- `ngDoCheck` - after every run of change detection
- `ngAfterContentInit` - after component content initialized
- `ngAfterContentChecked` - after every check of component content
- `ngAfterViewInit` - after component's view(s) are initialized
- `ngAfterViewChecked` - after every check of a component's view(s)
- `ngOnDestroy` - just before the component is destroyed

393. What are the data types in Angular?
- None, Angular is a framework.

394. What is CLI?
- CLI stands for Command Line Interface.
- Set of commands that you can run on the command line
- The Angular CLI is used for constructing the scaffolds of apps and components from the command line.

395. What are template literals?
- ??????

396. What is linting?
- Is a practice to encourage best practices when writing code.
- An automated process that checks practices like spacing, naming, variable declaration, etc.

397. What is NPM?
- NPM is the Node Package Manager
- Npm is a package manager used for web technologies
- Npm has a software registry with a bunch of packages (modules)
- Used to share reusable code and keep it up to date.
- Npm could refer to a couple of things, the website, the registry, or the client CLI.

398. What are Event Emitters?
- EventEmitter is a class in angular that's used by directives and components to emit custom events.
- Event emitters are defined as output events which is where you can create them and you can reference those events in the component and specify what they emit.

399. What is bootstrapping?
- Bootstrapping is the creation of specified components when the app is loaded and the addition of those components to the browser DOM.
- Components to be bootstrapped are specified by listing them in the NgModule's bootstrap array.

400. What are the data types in Typescript?
- Boolean
- String
- Number
- Array
- Enum
- Any
- Void

401. CLI commands to generate components and services.

- Ng g c component-name - generates a component.
- Ng g service service-name - generates a service.

402. a) ng generate component name
403. b) ng generate service name

404. When you start the angular application, what is the entry point of it, where does it start? (bootstrap).
- The base application bootstraps with AppComponent defined as the only endpoint by default.
- This can be changed by adding a new component to the bootstrap array in NgModule.

405. What is a template?
- A template is a scaffold for how something is supposed to look or act.
- In angular 4 you can define HTML templates for the applications components, you can define template contexts in interpolation expressions and many other things.
- You can load the templates into the application using the ng-template tag.

406. Is it possible to have module.ts in every component you create?
- Yes it is possible but there's not really a reason to do that, best practice is to have one module for you application and then import other modules to that module and inject it into the components and services that need them.

407. a) Not ideal, you should have one main module and then any custom module, like the routing module or anything that you want to modularize in your application.

408. How to inject a template into a view?
- Using the ng-template tags, or the router module can also inject template views for entire components.

409. a) (You normally do this with routing, and depending on a route the router will inject the template associated with a specific component).

410. Is it mandatory to export every component?
- No but if you don't export it you can't import it into any other component that needs it.

411. Explain the Http module.
- The Http Module is deprecated and it's now common practice to use the HttpClient Module instead.
- The HttpClient module is used to send Http requests and receive responses.
- It's set up by injecting it into the component meant to use it, done by placing a private variable in the constructor.
- You can reference the http variable to send http request and return observables and promises.

412. What is a Decorator?

- Essentially decorators are to angular as annotations are to java
- They're identifiers with metadata that tells angular what to do with a component or class.

413. How do I create a Module, what does it receive?

414. How do I include a Module into my application?

- Import the module in your app module using import { modulename } 'module path';
- Then list it as an import in the imports array of the NgModule
- After that you can inject it where you need it when/if necessary.

415. What is a Component?

- Everything!
- Everything in angular is a component.
- Components are simply typescript classes that are decorated with the @Component decorator which allows you to add more metadata.

416. How do I create a component, what does it receive?

- You can create components using the angular cli or create a ts class and decorate it with the @Component decorator.
- Using the cli will generate the html, css, and spec files for you but those files don't make a component a component and you could create all of those files yourself if you wanted.

417. What is a Directive? Which are the types? Examples

- Directives allow you to attach behavior to elements in the DOM.
- Directives must be specified in the NgModule in the declarations array.
- There are three types of Directives
    - Component
    - Structural
    - Attribute

418. What is Routing?

- In the context of a web application routing is the process of directing users through the different pages of the application and controlling where the user goes when clicking a link or a button.

419. How do I inject an object into another object?

- If you were to inject a service into one of your components for example:
    - Create a service (services are annotated with the injectable decorator)
    - Add the service class to the providers array of NgModule
    - Declare the service in the constructor of the component it's to be injected into
    - Angular takes care of the rest, you reference it with this.serviceName.method/property.

420. What is Data Binding?

- Data binding is essentially creating a connection between the application UI and the business logic.
- When a front end reference changes the state of the bound object or variable the variable itself updates with the changes

421. Which are the types of Data Binding?

- Interpolation - property name is enclosed in double curly braces {{ propertyName }}.
- Property
    - Used to bind values of component/model properties to the HTML element
    - might change the behavior of the element depending on values
    - [property]=”expression”.
- Attribute
    - set the values of an attribute directly, must only be used when the element doesn’t have the property itself.
    - Done in a way similar to property binding but using an attribute prefix
    - [attr.attr-name] = ‘expression’
- Class
    - add and remove CSS class names from HTML elements
    - Similar to attribute binding but starts with the class prefix not the attribute prefix
    - [class.Class-Name]=’expression’
- Style
    - Used to set inline styles to the HTML element
    - Style itself adds only a single line to the inline style
    - To add multiple styles with angular you can use the NgStyle directive.

422. Syntax for all types of data binding.

- Interpolation - {{ value }}
- Property - [property] = ‘expression’
- Attribute - [attr.attrName]=’expression’
- Class - [class.className]=’expression’
- Style - [style.styleProperty]=’expression’

423. HTTP calls in Angular.

- Get
- Post
- Put
- Delete

424. What is Observable?

- An observable is a way of handling async calls for data.
- Observables are lazy collections of multiple values that populate over time.

- In angular Observables are a class of generic type that you can return from an asynchronous call to make sure the data is there when you need it.
- You can subscribe to the call and when the observable populates you can get the data you need from it
- Observables are often prefered over promises because they can handle multiple events whereas promises are for a single event.
- Observables are also cancellable where a promise is not

425. Publisher/Subscriber.
- A messaging pattern where senders of messages (publishers) categorize published messages into classes without knowing how many subscribers there may be or which one's are using their data.
- Subscribers express interest in one or more classes and only grab messages that are of interest without knowing how many or which publisher they're getting data from.

426. How do I bundle my Angular application?
- Ng build command for the cli will bundle the application into a dist/ folder
- Specifying a production built with --prod will be more optimized for user interaction
- Not specifying production will not be optimized but it will be enough to pass around the development team.

427. What does TypeScript transpile into?
- Plain JavaScript

428. Access Modifiers in TypeScript?
- Typescript has two
    - Public and private

429. Non Access Modifiers in TypeScript.

430. TypeScript data types.
- DUPLICATE

431. Generics in TypeScript.
- They exist
- Similar to how java does it

DevOps

432. What is DevOps?
- DevOps is the practice of operations and development developers participate in together throughout the service lifecycle, form design to deployment to production support.

433. Why would you need DevOps?

- Devops describes a culture and a set of processes that bring development and operations teams together to complete software development.
- Once practices are in place development is extremely fast compared to other development processes, allowing multiple iterations.
- This allows you to build the base product and ad multiple features and fixes in quick succession.
- DevOps pipelines work in a cycle allowing for continuous development, integration, and deployment.
- Netflix and spotify have extremely effective build pipelines.

434. What is CI?
    - CI is short for continuous integration
    - Continuous integration is a development practice that requires developers to integrate code into a shared repository several times after a period called check-ins, these check-ins take pushed code and run an automated build to find problems early
    - The growing code base becomes more efficient with integrating new features and fixes, hence the name continuous integration.

435. What is CD?
    - Continuous Deployment
        - The next step after integration that is aimed at minimizing lead time.
        - Lead time is the time elapsed from a new push and that new code being used by live users in production
        - Continuous deployment is achieved by automating the various steps leading up to deployment so that after integration is successfully complete the live application is updated with the new code.
    - Continuous Delivery
        - Very similar to continuous deployment with the exception that continuous delivery doesn't mean that the code is deployed just that it is deployable at a given time.

436. What is CDD?
    - ?????

437. What are the 7 processes to DevOps?
    - ?????

438. What is IT Infratructure?
    - Refers to the composite hardware, software, network resources, and services required for the existence, operation, and management of an enterprise IT environment

439. What are the 3 parts of a business that are associated with DevOps?
    - ?????

440. What is Jenkins?

- Jenkins is a continuous integration and/or delivery environment.
- Runs on a server, uses hooks to pull code from a repository when the code is pushed, and it will run the tests and provided they pass it can then deploy it, depending on the implementation.

441. What is Maven?
- Maven is a project management tool that comes with a dependency manager to import plugins and external dependencies to your project.
- Comes with a cli that can be used to test, build, and package your application.

442. Where are Maven Dependencies Stored?
- Stored in a configuration file called the pom
- Pom is an xml file
- Pom stands for project object model.

443. What are the Maven Goals?
- Experience with maven is limited, from what i understand, goals are:
    - Processes that that maven binds to certain lifecycle phases that execute when calling a specific phase and in a specific order.
    - Goals can be defined and added

444. What are the Maven Lifecycles?
- 3 lifecycles
    - Default
    - Build
    - Clean
    - Site

445. What are the phases in the Build Lifecycle?
- There's a bunch:
    - Validate
    - Compile
    - Test
    - Package
    - Integration-test
    - Verify
    - Install
    - Deploy

446. What is GREP?
- A unix/linux command that searches files for the occurence of a string that matches a specific pattern.

447. What are the different package management systems in UNIX?

- Dpkg
- Entropy
- Flatpack
- Apt-get (no longer apt-get i think just apt)
- Homebrew

448. How do you connect to EC2 from Putty?
- Assuming your ec2 is up and running and you have putty installed properly and open:
    - Download your ec2 key pair file from the aws website
    - Convert the pem file to ppk file using puttygen make sure to save it somewhere safe.
    - Open up putty
    - Enter the ec2 hostname usually in the format of user_name@public_dns_name. (changes based on AMI type)
    - Open the auth page/tab and select your ppk for authentication
    - Once that’s done click open to start the SSH session.

449. How do you configure Jenkins?
- Assuming jenkins is already installed:
    - Run Jenkins as a service
    - Create Users - this is for added security and role management, not anyone can deploy or rollback builds, not everyone can see build results, etc.
    - Create a Jenkins project - this is the actual project configuration of jenkins (assuming you have necessary plugins)
        - Click new item
        - Select the project type:
            - Freestyle project
            - Maven project
            - External Job
            - multi-configuration project
        - Set up the source repository for this you can generate a github hook that jenkins can use on github.
        - Next you can add build steps that jenkins should take when activated.
        - After that save your changes and whenever that hook is activated jenkins will pull the repo and execute the specified build steps.

450. What is Agile?
- Agile is a development strategy based on the idea of incremental delivery using technology such as continuous integration, development, delivery, and deployment.
- The strategy is to break the application into a set of features and add features to the basic application in a series of sprints to build the software overtime.
- Using continuous integration and deployment tools sprints can go very fast and helps detect bugs and problems as you go making bug fixing quicker.

451. What is SCRUM?

- Scrum is the practice of the agile strategy
- Agile is the idea, scrum is agile in practice.
- In Scrum there is a scrum master who holds meetings to track the progress of the sprint
- The scrum master also breaks the sprint into features and assigns them points according to difficulty
- Team members pick features to work on and their work is classified from in progress, to testing, to fixing or debugging, and finally to complete/finished.
- Finished features are pulled to the staging branch which will run a set of integration tests and deploy.

452. What are the important parts of SCRUM?

- There are three main components:
    - The product owner: the company
    - The scrum master: project manager
    - The scrum team: the developers
    - The master defines sprints and features with difficulty points
    - The team takes features and develops them
    - Progress is tracked throughout the development process with standups and tests.

453. What is the Waterfall Methodology?

- The development lifecycle for an entire application
- An alternative to the agile methodology.
- Instead of breaking up the application into features and sprints it brakes the development process up into steps.
- Steps are:
    - System engineering
    - Analysis
    - Design
    - Code
    - Testing
    - Maintenance
- All steps must be completed one at a time, you can’t go to the next step until the entire application is finished on the step before.
- you have to go all the way to the bottom of the chain before you can go back up to the top and make a new application or version.
- Typically less favored in software development because the pace.

454. What is the U-Model Methodology?

- Didn’t discuss this

455. What is the College Model Methodology?

- Didn’t discuss this

AWS

456. What is AWS?
- Stands for Amazon Web Services
- A secure cloud services platform
- Offers a ton of services for:
    - Cloud computing
    - Storage
    - Databases
    - Migration
    - Management tools
    - Mobile services
    - Customer Engagement
    - Business Productivity
    - Machine Learning
    - Analytics
    - Desktop and App streaming
    - IoT
    - Game Development
    - Etc.

457. What is IAM?
- Stands for Identity and Access Management
- Enables users to manage access to AWS services and resources securely
- You can use IAM to create and manage user groups with permissions for AWS resources.

458. What is SonarQube?
- Essentially a linting tool
- Used to encourage best practices, helps with things like format and identifying potential bugs.

459. What does AWS focus on?
- ?????

460. What is Cloud Computing?
- The use of remote servers through a network to do computation tasks like program execution, storage, management and processing of data

461. What is PaaS?
- Platform as a service
- A category of cloud computing services that provides a platform for customers to develop run and manage applications without the complexity of developing the infrastructure associated with developing and launching an application.

462. What is IaaS?
- Infrastructure as a service
- A form of cloud computing that provides a virtualized computing resources over the internet.

463. What is SaaS?
- Software as a service is a software distribution model in which a third-party provider hosts applications and makes them available to customers over the internet. (google docs, gmail, etc.)

464. What is EC2?
- Elastic Compute Cloud
- Essentially a virtual machine on a server with a specific allocation for hardware resources based on plans
- The more resources the more expensive
- Used for running a server to host an application that users can connect to.

465. What is EBS?
- Elastic Block Store
- A traditional ‘block’ storage resource that is provisioned to other AWS services.

466. What is AMI?
- Amazon Machine Image - an OS image to run on an EC2, that is the operating system that the ec2 has
- There are a few to choose from including amazon linux, ubuntu, windows, etc.

467. What is ELB?
- Elastic Load Balancing
- A load balancing service for AWS.
- ELB automatically distributes incoming application traffic and scales resources to meet traffic demands.

468. What are the types of ELB?
- Application load balancers
- Network load balancers
- Classic load balancers

469. What is Autoscaling?
- A method used in cloud computing where the number of instances for a server scales automatically based on the connection and request load.

470. What is a Security Group?

- A virtual firewall that controls the traffic for one or more instances. When an instance is launched you associate a security group to it for it to filter traffic down to only IPs matching that group.

471. What is S3?
- Amazon's Simple Storage Service.
- Designed with a simple web service interface that can be used to store and retrieve data.
- Can also be used for static web hosting.

472. EBS vs S3?
- EBS is not a standalone service and must be used in conjunction with EC2 or other associated services meant to store data in volumes of a predefined size like a hard drive on a physical machine.
- S3 is standalone and typically meant for large storage functions like database backups although that's not its sole purpose.

473. What is RDS?
- Relational Database Service
- Makes it easy to set up, operate, and scale a relational database like postgreSQL, mySQL, or OracleSQL.

474. Region vs Availability Zones
- AWS Regions are a separate geographical area and has multiple isolated locations that are availability zones
- Availability zones are the locations that are available to put instances
- For example US East is a region, and Ohio and N. Virginia are availability zones.

475. What are the EC2 instance types?
- Three main types
    - T2
    - M5
    - M4
    - Each have small medium and larger subdivisions.

476. What is Route 53?
- Amazon route 53 is a scalable and highly available DNS
- There you can register domain names for you applications
- Route internet traffic to your domain
- And check on the health of your application.

477. How do you setup an AWS EC2?
- Go to the AWS site
- sign in to the console

- go to the EC2 service page
- Click on instances then launch instance
- Select your AMI
- After the page reloads select your instance type (resources and purpose)
- Next click review and launch, make sure that everything you see is indeed what you want, then click launch
- Upon clicking launch you can generate an ssh key pair and save it somewhere safe because that's how you'll access the EC2.
- There will be some dialogue options for different setting configurations but you can click ok through all of those and configure them later
- Once you're done with that the instance will take a few minutes to spin up but your job is basically done.

478. How do you setup an RDS EC2?
    - ??????

479. What is resiliency?
    - A systems capability to handle disaster such as weather, power, or other catastrophe that can affect the systems and their services so that the application can stay available to users.

Hibernate

480. What is Hibernate?

Hibernate is an Object Relationship Mapping Framework/Tool. It allows a program to map a class to a relational database. In doing so, the programmer creates a sessionfactory. The sessionfactory contains the connection info for the database, as well as the driver, and dialect. You can map files either by a mapping file, or by JPA annotations in the classes you want mapped.

481. What does it abstract?

Hibernate is an abstraction of JDBC.

482. How do you configure Hibernate?

You can configure hibernate via a configuration xml, or programmatically. In the xml, you would set up the session factory with the sessionfactory tag. It would contain all the info for your database, like driver, dialect, connection info, and any and all mappings.

483. Why do you need ORM tools like Hibernate?

You need ORM tools like Hibernate because it would take a long time to setup all of the DAOs in a JEE project/environment. Because it abstracts JDBC, it sets up all of your daos after you map your classes. Hibernate can even create the database for you.

484. What is ORM?

ORM stands for Object Relational Mapping. It is a way to relate a programs beans to a relational database. You can you annotations like @Id to specify that a particular variable is the ID for that table and so on.

485. What does ORM consists of?

ORM consists of taking an existing bean, and mapping it to a relational database. To start, you would annotate the class with the @Entity annotation. You would then annotate the class @Table and give it a table name. As for the variables of the class, you must annotate the ID with @Id, and all of the other variables with @column.

486. What are the ORM Levels?

The ORM levels are:

Pure Relational Mapping - stored procedure
Light ORM Mapping - like JDBC
Medium ORM Mapping
Full ORM Mapping alike Hibernate

487. What are the Hibernate Object States?

The Hibernate Object States are:

Transient - Object exists outside of session and the row
Persistent - Object in the session that represents a row in the database
Detached - Object is removed from the session, any actions on the object are not saved to the database.

488. What are Application Transactions?

Application transactions are any action that needs to modify data stored in the database. There are different ways of ensure ACID properties of these transactions.

489. What are the query types in Hibernate?

Criteria API
Native SQL
HQL

490. What is HQL?

HQL is Hibernate Query Language. HQL is hibernates version of SQL. It is made to act more like OOP.

491. What is Native SQL?

Native SQL is a way of running regular SQL queries in Hibernate.

492. What is the Criteria API? How does it differ from HQL?

The Criteria API is one main interfaces of Hibernate. It allows for programmatic querying of the database. They use restrictions and projections. Restrictions act like a where clause, while projections act like aggregate functions. Criteria API differs from HQL in that it is fully programmatic. HQL is still structured like SQL. It just understands OOP principles..

493. What is so special about Hibernate 4.0+?

- Has to do with sessionfactory creation

494. What are the benefits to Hibernate?

Hibernate abstracts away some of the redundantness of JDBC. You don't need to program every DAO with Hibernate. Hibernate also has built in support for transactions and the sessions have built in methods such as get and load.

495. What do you need to include in the hibernate.cfg.xml file?

The Hibernate config file must have all the details to setup the sessionfactory. These details are the connection info, dialect, and driver. It must also have all mappings in a mapping tag.

- Driver
- Dialect
- Username
- Password
- URL
- Link to mapping file

496. What is Hibernate.hbm2ddl?

Hibernate.hbm2ddl is a property you can set in the cfg that will tell hibernate whether or not to make the database. It is best not to use this in anything that isn't a development environment.

497. What is HBM?

HBM stands for Hibernate Mapping. This tells hibernate how to map a given object to a relational database.

498. What do you need to include in the HBM?

The HBM must have a class name, and what entity it relates to in the database. It must have at least an ID specified, and how to generate the ID.

499. What is mapped to the database?
500. What are some common annotations?

- @OnetoMany
- @ManytoOne
- @ManytoMany
- @OnetoOne
- @Entity
- @Table
- @Column

501. What are the important interfaces?
502. What are the different fetching types?
503. Lazy Fetching vs Eager Fetching
504. What is a Proxy?
505. What are the relationships in Hibernate?
506. What is a Cache?
507. What are the types of Caches? What are the differences? What are the levels? L1, L2
508. What are some providers for L2 Caching?
509. When should you use javax.persistence vs org.hibernate?
510. What is a Restriction?
511. What is a Projection?
512. What does a Session do?
513. What does a SessionFactory do?
514. What does a Transaction do?
515. What is a Query Interface? What methods does it provide?
516. What is a Criteria Interface? What methods does it provide?
517. Sorting vs Order

518. Merge vs Update
- Both are updating information that exists in the database
- Merge - can be called on objects that are persisted
- Update - can only be called on objects that are transient
- SaveOrUpdate - calls either save or update based on if the identifier exists
- Save - store the object into the database; persists transient instance and returns id of the entity created

519. What are the Cascade Types?
- JPA Cascade Type
    - All - shorthand for ‘all of the above cascade operations’
    - Remove - delete the child when the parent is deleted
    - Refresh - same thing as the refresh operation
    - Persist - save or persist operations cascade to related entities
    - Merge - related entities are merged when the owning entity is merged
    - Detach - detaches all related entities if a manual detach occurs
- Hibernate Cascade Type

520. What is Dirty Checking?
521. What are the Hibernate Collection Types?
522. How do you register an HBM file?
523. How do you register an annotated POJO?
524. What is mappedBy?
525. Which relationship requires mappedBy parameter referencing?
526. What is @Embedded annotation used for?
527. What is @AttributeOverride annotation for?

528. What is @AttributeOverrides annotation for?
529. What is the @JoinTable annotation for?
530. What are some org.hibernate.annotations.*?
531. What is the @CollectionId annotation for?
532. What is @GenericGenerator annotation for?
533. What is the @Type annotation for?
534. What is the @NotFound annotation for?
535. Cache vs Cacheable
536. What is the general flow of Hibernate to the RDBMS?
537. get() vs load()
    - get() returns a proxy, if no row is found throws ObjectNotFoundException. Load always hits the database returns null if no row found.

538. What is cascade?
539. What is inverse?
540. How do you invoke a Stored Procedure?
541. How do you switch between RDBMS's?
542. Sorted vs Ordered Collection
543. How would you express a join in Hibernate?
544. Dynamic Insert vs Dynamic Update
545. What is Transactional-Write Behind?
546. What is a Callback Interface?
547. What are the types of Inheritance Models?
548. JDBC vs Hibernate
549. How do I set up an entity?
550. How do I relate entities? (Cardinality Mapping)

551. How to create queries?
    - HQL - OOP way of writing query
        - Hibernate Query Language
        - Uses bean names and properties
    - Native SQL
    - Criteria - programmatically create queries

552. Session vs SessionFactory.
553. L1 vs L2.
    - Caching - won't have to hit database each time
    - L1 caching - session only
    - L2 caching - session factory basis; information you would want to access throughout sessions
        - Expensive operation to store; lots of memory to store
    - Query caching - query resultset

554. How do I configure Hibernate?
555. How do I map a class without annotations?

556. Hibernate interfaces.
557. Session methods.
558. Difference between Criteria and HQL.
559. Object states.
560. What will Criteria or HQL return (list() of a POJO expected).
561. Default fetching strategy in Hibernate.

562. Hibernate exceptions.
- ObjectNotFoundException - load
- LazyInitializationException - with lazy, you have to do a join in the query otherwise the referencing object won't be available to query later and you'll get the lazy initialization exception
- OptimisticLockException - detects conflicting update of an entity at the same time
- UnknownIDGenerator
- QuerySyntaxException

563. How do you handle exceptions using Hibernate?
564. Update vs Merge.
565. All methods in the Session interface.
566. How do you configure ehcache?
567. Dynamic insert vs update.

568. Named queries.
- @NamedQuery in the bean
- Access it quickly
- Similar to stored procedure

569. How do you get an instance of the SessionFactory?
570. Can you perform joins in Hibernate?

571. How do you create a bean in Hibernate?
- Annotations
    - @Entity
    - @Table (name="")
    - @Column
    - @Id
    - @GeneratedValue (specify strategy)
    - @JoinColumn
    - Specify multiplicity @OneToMany, @ManyToOne, @ManyToMany
    - @MappedBy
- XML

572. What's the difference between sorted and ordered collections?
- Sorted - natural ordering; use it for smaller collections; expensive to sort

- Ordered - use it for larger collections; specify by the ORDER BY clause
- Collections - resultset

573. Collections you can use in hibernate
- Bag - retrieved randomly
- Array
- Map
- Set
- List

574. What is a hibernate proxy?
- The proxy attribute enables lazy initialization of persistent instances of the class

575. What is an automatic dirty checking?
- A feature that updates the db when we modify the state of an object in a transaction
- Saves us effort for asking hibernate to update the database after a modification

576. What is a transactional write-behind?
- Hibernate uses a sophisticated algorithm to determine an efficient ordering that avoids db foreign key constraint violation, but it's still sufficiently predictable to the user

577. What're advantages of Hibernate over JDBC?
- Helpful error messages
- Object Relational Mapping - change databases easily
- Reduces human error / stringification

578. Different kinds of fetching
- Lazy fetching - fetched when needed; used in conjunction with joins
    - Lazy initialization exception
- Eager fetching - fetched at the start; automatically joins

Spring

571. What is Spring?
- Framework created to address the complexity of enterprise level application
- Includes an Inversion of Control container
- Involves layered architecture that users can select which aspects of Spring to use
- Components/Modules - Test, Core Container, Web, Data Access and Integration, AOP

572. Spring is loosely coupled. Explain why we say this.

573. Bean Scopes.
- Prototype -
    - Properties changed in one instance won't affect other instances of that bean
- Singleton -

    - Properties changed on a global level
- Request - (web service) scope of a single HTTP request
- Session - (web service) within a session
- GlobalSession - (web service) lifecycle of a global HTTP session

574. Spring Modules
- Spring Web
- Spring Core
- Spring Context
- Spring Data
- Spring Security
- Spring AOP
- Spring MVC

575. Bean Lifecycle.
- Spring finds bean definition
- Instantiate
- Populates properties using dependency injection
- If the bean implements bean name aware -> set bean name
- Bean factory aware - make the factory aware of the bean -> set bean factory
- Pre init
- Post processor
- Init
- Custom init
- Ready - being used
- Destroy
- Custom destroy
- Finalize

576. What is the Spring Container?
- IOC Container
- Application Context
- Bean Factory

577. ApplicationContext vs BeanFactory.
- Spring's IOC containers
- Bean Factory - no longer in use; lazy initialization for beans (created upon request)
- ApplicationContext - supports Spring's newer modules; eager initialization for beans (can also tell it to do lazy initialization); includes new features such as messaging

578. Inversion of Control.
- The idea that parts of the architecture you set up yourself, you pass on to some other entity to other handle for you.
- Don't worry about the implementation, focus on the business logic.

- Spring uses dependency injection, which implements inversion of control.

579. Types of Dependency Injection
   - Constructor-Based
   - Setter-Based: When the container calls setter methods on your beans after it has invoked either a no-arg constructor, or a no-arg static factory method to instantiate that bean

580. Different types of Bean wiring.
   SETTERS AND CONSTRUCTORS
   - Bean wiring tells the IOC container the beans' relationship through setter or constructor injection
   - Autowiring
     - Using @Autowire

581. Autowiring. How to configure it?
   - By name
   - By type
   - No (default)
   - By constructor

582. How do I create a bean in the container?
   - If you are using XML-based configuration metadata, you can specify the type (or class) of object that is to be instantiated using the 'class' attribute of the <bean/> element.
   - You can declare beans using the @Bean annotation in a configuration class.
   - Or you can mark the class with one of the annotations from the org.springframework.stereotype package and leave the rest to component scanning.

583. How do I enable/configure annotations?

584. Stereotype Annotations.
   - Below are class-level
   - @Component
   - @Controller
   - @Service
   - @Repository
     - exception handler included, so don't need to write try-catch block. (because of automatic translation feature)

585. @Autowired.
   - Marks a constructor, field, setter method or config method as to be autowired by Spring's dependency injection facilities.

- Only one constructor (at max) of any given bean class may carry this annotation, indicating the constructor to autowire when used as a Spring bean. Such a constructor does not have to be public.

586. @Autowired vs @Inject vs @Resource.
- @Autowired - Spring property annotation that inject a resource by-type, i.e. by the class or by the interface of the annotated field or constructor.
- @Inject - Identifies injectable constructors, methods, and fields. This annotation is an almost complete drop-in replacement for Spring's @Autowired annotation. So, instead of using the Spring-specific @Autowired annotation, you might choose to use @Inject. One of the differences between @Autowired and @Inject is that @Inject does not have the required field so in case we fail to find a suitable object to injected it will fail while @Autowired can used required=false and allow null-able field (only if required!).
- @Resource - Is quite similar to @Autowired and @Inject, but the main difference is the execution paths taken to find out the required bean to inject. @Resource will narrow down the search first by name then by type and finally by Qualifiers (ignored if match is found by name). @Autowired and @Inject will narrow down the search first by type then by qualifier and finally by the name.

587. Contextual Session.
- How you set up your Daos in Spring
- Repository
- Two approaches:
    - Annotations -
        - @Repository dao implementation class
        - @Transactional
        - Implement CrudRepository, JPARepository
    - XML -
        - Bind transactions to crud methods in application context
- How do you integrate Spring and Hibernate using Contextual Sessions?

588. How do I integrate Spring and Hibernate?

589. How do I make Spring handle transactions for me? configuration?

590. JSR-303.

591. JndiObjectFactoryBean.

592. What is Spring MVC?
- Request from client
- Dispatcher Servlet
- Handler Mapper - searches for controllers
- @Controller or @RestController

    - @RequestMapping - url, data types
    - @RequestBody
    - View Resolver for Controller
    - Data only for RestController
    - @PathVariable - if the param is called the same name as the path param, you don't need the extra definition
        - Purpose: identify a specific object (ie id 1)
    - @RequestParam (query param)
        - Purpose: use for filtering (ie. searching for all blue suits, query param = blue)

593. Spring MVC Workflow.
594. Spring MVC Annotations
595. What is @RequestMapping?
    - Marks a method as an endpoint to which a client can connect, where you can specify the request type it will handle and the path of the endpoint.
596. InternalResourceViewResolver.
597. ContextLoaderListener.
598. How to configure Spring MVC?

599. What is AOP?
    - Aspect Oriented Programming
    - Code injection
    - Address cross-cutting concerns
        - Transactions
        - Authorization
        - Logging
    - Aspect, Advice, JoinPoint, Pointcut

600. How to configure AOP?@Aspect at the class
    - Advice annotations - include path of execution

601. Cross Cutting Concerns.
    - Code that can't be cleanly decomposed from business logic (concern - part of the system divided on the basis of functionality)
    - Logging - needing access to certain variables in that class

602. Aspect.
    - The encapsulation of a cross cutting concern
    - The combination of Pointcut and Advice

603. Advice.
    - Before

- After (Finally)
- After Returning
- After Throwing
- Around

604. Advice Kinds.
605. @Before vs @Around.
606. @After vs @AfterReturning vs @AfterThrowing.

607. JoinPoint vs Pointcut.
   - Pointcut - point of execution in which a cross cutting concern needs to be applied
   - Joinpoint - where you're looking for your pointcut

608. Spring AOP vs AspectJ.

609. Load time vs Compile time Weaving.
   - Compile-time weaving is the simplest approach. When you have the source code for an application, ajc will compile from source and produce woven class files as output. The invocation of the weaver is integral to the ajc compilation process. The aspects themselves may be in source or binary form. If the aspects are required for the affected classes to compile, then you must weave at compile-time. Aspects are required, e.g., when they add members to a class and other classes being compiled reference the added members.
   - Load-time weaving (LTW) is simply binary weaving defered until the point that a class loader loads a class file and defines the class to the JVM. To support this, one or more "weaving class loaders", either provided explicitly by the run-time environment or enabled through a "weaving agent" are required.
610. Path parameters.

611. Transaction propagation strategies
   - Required - supports a transaction and creates a new one if it none exists
   - Mandatory - supports a current transaction and will throw an exception if none exists
   - Never - executes non-transactionally and throws an exception if a transaction exists
   - Supported - Supports a current transaction if one exists but will execute non-transactionally if none exists.
   - Not Supported - executes non-transactionally and suspends the current transaction if none exists
   - Requires New - Will suspend the current transaction and create a new transaction if it exists
   - Nested - it's nested, part of the transaction

612. Set up Application Context
   - New Application Context
     - XML files

- ClasspathXMLApplicationContext
- FilesystemXMLApplicationContext
- WebXMLApplicationContext
- Annotations - mark it @Configuration; each bean mark @Bean
    - AnnotationConfigApplicationContext

Web Services

611. What is a Web Service?
- A web service is any piece of software that makes itself available over the internet and uses a messaging system.
- Web services are self contained, modular, distributed, dynamic applications that can be described published, located, or invoked over the network to create products, processes, and supply chains. These applications can be local, distributed, or web-based.
- Uses a standardized XML messaging system at the very least.

612. What is SOAP?
- Simple object access protocol. Communication protocol to communicate via Internet.
- Only supports XML messaging protocol via HTTP.
- Can be both stateless and stateful.
- Built-in error handling.
- XML document contains envelope, header, body, and fault.

613. What is REST?
- An architecture
- Constraints defining a restful system
    - Client server architecture
    - Stateless
    - Cacheability
    - Layered system - a client can't tell if it's connected directly to the RESTful service or an intermediary
    - Code on demand - Servers can temporarily extend or customize the functionality of a client by transferring executable code
    - Uniform interface

614. What is RESTful?
- Uniform interface
- Client-Server independence
- Stateless
- Cacheable
- Layered System
- Code on demand(optional)

615. SOAP vs REST
- *Hard to make a ***direct*** comparison since you're comparing a protocol with an architectural style/pattern
- Main difference is the ***degree of coupling*** between the client and server.
- SOAP clients are heavily coupled to the server, essentially have a ***rigid contract*** with each other

- REST client is more ***generic***, which knows how to use a protocol and standard methods ***(aka protocol independent)***
- Stack Overflow: SOAP vs REST

616. What is JAX-RS?
617. What is JAX-WS?
    - Java API for XML Web Services (JAX-WS) is one of a set of Java technologies used to develop Web services.
618. What is JAX-RPC?
619. What is JERSEY?
620. What is an Endpoint?
621. What is Idempotent?
622. What are the annotations used in Web Services?
623. What is Code-On-Demand?
    - Sending static representations of resources in forms of XML or JSON.
624. What is the lifecycle of a Service?
625. What are the RESTful characteristics?
626. What does it mean to be Layered?
627. What does it mean to be Stateless?
628. What does it mean to be Cacheable?
629. What is Uniform-Interface?
630. What is a URI?
631. What are some REST characteristics?
632. What are some benefits to using REST?
633. What the HTTP Methods? What CRUD operations do they correspond to?
634. What do you need to make a REST application into a RESTful service?`
635. What is WSDL?
636. What are the types of elements in WSDL?
637. What are the portType elements?
638. What is a SOAP Envelope?
639. What tools should I use to test a Web Service?
640. What is Postman?
641. What is SoapUI?
642. What is XML Namespace?
643. How do I call a resource from Angular?
644. Complex Types vs Simple Types
645. How do you build a RESTful Web Service using Maven?
646. How do you consume a Web Service?
647. How do you expose a Web Service?
648. What is the Web Service Protocol Stack?
649. What is UDDI?
650. What is SAAJ?
651. What are the APIs used to implement Web Services?
652. What is an XML Schema?
653. What is DTD?

654. XML Schema vs DTD
655. What is the basic structure of SOAP 1.1
656. What are the required components of JERSEY?
657. HTTP Error Codes: 200, 202, 204, 300, 400, 401, 403, 404, 415
658. What are the components of an HTTP Request?
659. What are the components of an HTTP Response?
660. PUT vs POST
661. Why do we use Web Services?
662. Protocol stack.
a. Transport Protocol.
b. Messaging Protocol.
c. Description Protocol.
d. Discovery Protocol.
663. How do you create a SOAP Web Service?

- Use JAX-WS
- Service interface, add @WebService and @SOAPBinding
- Implement service, add @WebService
- Main method - publish endpoints (endpoint.publish), give it the url and instance of implementation class
- Generate WSDL

a. Explain all the steps to expose and consume.
b. Talk about contract first and contract last.
c. Apache CXF for contract last.
664. How do you create a RESTful Web Service?
a. Explain all the steps to expose and consume.
665. Annotations in REST.
666. How do you handle exceptions in REST?
667. What's the structure of the HTTP Request?
668. What's the structure of the HTTP Response?
669. What’s the difference between @RestController and @Controller?
670. How do you handle exceptions in REST?
671. HTTP methods.
672. GET vs POST.
673. Idempotency.
674. What media types does REST support?
675. What does stateless mean?
676. Is SOAP stateless?

- Yes. No state is saved by client or server in SOAP.

677. What are endpoints in SOAP?
678. WSDL Tags in detail.

- Definition − It is the root element of all WSDL documents. It defines the name of the web service, declares multiple namespaces used throughout the remainder of the document, and contains all the service elements described here.

- Types − The data types to be used in the messages are in the form of XML schemas.
- Message − It is an abstract definition of the data, in the form of a message presented either as an entire document or as arguments to be mapped to a method invocation.
- Port type − It is an abstract set of operations mapped to one or more end-points, defining the collection of operations for a binding; the collection of operations, as it is abstract, can be mapped to multiple transports through various bindings.
- Operation − It is the abstract definition of the operation for a message, such as naming a method, message queue, or business process, that will accept and process the message.
- Binding − It is the concrete protocol and data formats for the operations and messages defined for a particular port type.
- Service − It is a collection of related end-points encompassing the service definitions in the file; the services map the binding to the port and include any extensibility definitions.

679. What is the difference between portType and binding?
680. SOAP Binding styles.
681. SOAP message structure.
682. What is the header of the message used for?
683. What is the fault of the message used for? Where is it in the structure?
684. Contract First - Contract Last, how to implement it? (JAX-WS within Apache CXF)
685. SOAP exception handling.
    - Fault code will be communicated in the SOAP-ENV:Fault tag.
    - A fault string describing the exception is communicated in the same tag.
    - A fault actor provides information about where in the message path the fault occurred.
686. Differences between SOAP and REST.
687. What media types does SOAP support?
688. What protocols does SOAP allow?
    - SOAP can operate over any protocol.

Microservices

689. Main advantages of using microservices.
    - Services are independently deployable and scalable
    - It allowing for different services to be written in different programming languages.
    - Scalability and reusability
    - Work well with containers such as Docker
    - Better fault isolation; if one microservice fails, the other will continue to work
    - Handling traffic well - instances of services created as needed

- Loose coupling

690. Any disadvantage in using microservices.

- DevOps complexity - Due to several moving parts of the application. It becomes complex.
- Increased Resource use - Initial investment to run these applications are high because all the independently running components need their own containers
- Ease overhead - people need to communicate to make sure update doesn't occur error on other services, Cost to monitor each individual service
- Databases are more difficult to keep normalized

691. Dependencies of Microservices (what do we need to use).

a) Eureka (Discovery).

- client side discovery pattern
    - client obtains the location of a service instance by querying a service registry, which knows the locations of all service instances.
    - Ribbon creates requests by load balancing algorithms

b) Zuul (Gateway).

- Zuul acts as an API gateway. It receives all the requests coming from the UI and then delegates the requests to internal microservices.
- The client calls this service as a proxy for an internal microservice, then this service delegates the request to the appropriate service
- The advantage of this type of design is that common aspects like CORS, authentication, and security can be put into a centralized service.
- we can implement any routing rules or any filter implementation

c) Configuration Server.

- Configuration server to centralise management of all the configuration files
- mainly responsible for providing properties files to each microservice connected to the discovery server
- ex) if we need to update just a username for a database in property file, we need to first locate where the module is stored, then go to that server, change the property for that module and rebuild the whole module. In microservices, there will be a huge number of modules so it is very difficult. so In microservices, a new layer has been introduced to manage all things related to properties files.

d) Kafka (Messaging).

- Apache Kafka is an open-source stream processing software platform
- For scale-out and high availability broker architecture with cluster = broker
- Consumer pull from the broker (other message system is broker push to consumer)
- The project aims to provide a unified, high-throughput, low-latency platform for handling real-time data feeds.
    - 4 major API in Kafka
    - Producer API
    - Consumer API
    - Connector API

        - Streams API

e) Hystrix (Monitoring).

- Circuit breaker module
- wrap a protected function call in a circuit breaker object, monitors for failures,. and when it happen it trips and further calls to circuit breaker return an error without call being made at all. This will prevent cascade error across multiple system by run out of critical resources.

f) Docker (Orchestration).

- Docker is software which lets you create an image(template for virtual machine)

692. Annotations of Eureka and what they do.

- @EnableEurekaServer - to stand up a registry that other applications can talk to
- @EnableEurekaClient - to in the list of registered applications

693. How to implement Registry and Discovery services.

694. What exception will get thrown if you try to run the client (single microservice) before the server (eureka)?

695. How to implement Gateway service.

696. What happens if you don't have a gateway? Can you have microservices without this?

- Technically, yes, you can have a microservice architecture without a gateway, but because your gateway helps you to route to each of your microservice instance, without the gateway, the client would have to know the url of each microservice instance, but even then, it wouldn't know if the microservice instance was active because there's no gateway to query the discovery service, which receives heartbeats from the microservice. The client can't query the discovery service. The discovery service doesn't receive requests from the client, only heartbeats from the microservices and sends a list of the active ones to the gateway.

697. Annotations of Zuul and what they do.

- @EnableZuulProxy - sets up Zuul so that you can configure routing without Eureka; will turn the Gateway into a reverse proxy that forwards relevant calls to other services

698. How do you make Zuul to employ discovery for the routes to services?

- @EnableDiscoveryClient - to be used with Discovery to enable dynamic routing

699. How to implement Monitoring service.

700. Annotations of Hystrix and what they do?

- @HystrixCommand(fallbackMethod = "methodName") - place this annotation above a method you want Spring Cloud Netflix Hystrix to look for

        - The annotation wraps that method in a proxy connected to a circuit breaker so that Hystrix can monitor it
        - ONLY WORKS in a class marked with @Component or @Service
        - Example of fallbackMethod is reliable; specify it to provide a placeholder of what you need for users

701. How to implement the Orchestration service?
702. What is Docker?
703. Workflow of a Docker? How does an image reach the container?
704. Explain each component in Docker containers.
705. Docker commands.
706. Image vs Instance.
707. Some properties in a docker-compose file.
708. How to make an image available in the main Docker instance.
709. Some commands in a docker file.
710. How do you create a docker image?
711. How to implement the Messaging Service.
712. Publisher/Subscriber in Kafka.
713. Annotations/Classes of Kafka and what they do?
714. How do we enhance the natural resiliency of the Microservices architecture?
715. Explain Spring Boot.
    - Opinionated view of the Spring platform to make it easy to configure Spring applications for various use cases.

716. How do you configure the custom port in Spring Boot? (server.port = number in application.properties).
717. How do you reload any changes in Spring Boot without having to restart it?
718. How do you perform any database operations with Spring Boot?
719. What are actuators in Spring Boot?

720. What is Spring Data?
    - Spring module for abstracting database connection and querying operations using Hibernate as its ORM (Object Relational Mapper)

721. How do you define your custom Repository?

722. What is a microservice?
    - Modular in design, can be running many instances
    - Not monolithic
    - Easy to scale
    - Self-contained
    - Well-defined scope - perform one action
    - Part of a larger architecture
    - Micro doesn't refer to its size, but of its part in the larger whole

723. How do the microservices register with the Discovery?
- Microservices register themselves
- Registry
- Heartbeats - lets Discovery know it's still alive

724. Name some annotations for microservice components.
- @EnableEurekaServer
- @EnableEurekaClient
- @EnableZuulProxy
- @EnableDiscoveryClient
- @HystrixCommand
- @LoadBalance
- @RibbonClient

725. Zuul Filter
- Types:
    - Pre - filters execute before routing to the origin
        - Ex: request authentication, choosing origin servers, logging debug info
    - Routing - filters handle routing the request to an origin
    - Post - filters execute after the request has been routed to the origin
        - Ex: adding standard HTTP headers to the response, gathering statistics and metrics, and streaming the response from the origin to the client
    - Error - filters execute when an error occurs during one of the other phases
- Characteristics
    - Type - defines the stage during the routing flow of WHEN the filter will be applied (can be a custom string too)
    - Execution Order - applied within the Type, defines the order of execution across multiple filters
    - Criteria - the conditions required in order for the filter to be executed
    - Action - the action to be executed if the Criteria is met
- Filters share a state through a RequestContext which is unique to each request

Pivotal Cloud Foundry

725. What is PCF?
- Pivotal's implementation of CF
- Platform As A Service - only have to upload your application; don't have to set up your environment

726. Different services
- PAAS - upload application
    - Heroku
    - PCF
    - OpenShift

- IAAS - setup environment
    - AWS
    - Azure
- SAAS - use the application
    - Gmail
    - Google Docs

727. What are some things you would want in a cloud platform?
- Scalability
- Security and integration
- Self-healing
- Statistics / Metrics
- Logging
- Application Lifecycle
- Portability

728. Components of CF
- Router - direct incoming traffic to cloud controller
- Authentication - OAuth
- Cloud Controller
- NSync - receives message from cloud controller when you scale the app
- BBS - handles desired LRP == actual LRP; creates/destroys instances to keep those numbers equal; stores disposable data (application status, heartbeats)
- Cell Replication - gives the actual LRP
- Storage:
    - Blob Store - stores binary files
    - Diego Cell - manages lifecycle of containers (Garden containers) and processes running in them; also reports their status to the BBS; makes logs and metrics to Loggregator
- Service Broker - look up services
- Consul - long term information; ip address
- Loggregator - streams application logs to the developer
- Metrics Collector

729. How to deploy your app on PCF
- Manifest
    - Name
    - Instances
    - RandomRoute
    - Memory
    - Path (of your jar)
- Cf push
    - Pushes your jar to pcf to deploy your application

730. Annotations to know

-> Web Service
    - @WebService
    - @WebMethod
-> Restful Web Service
    - @Path
    - @GET
    - @PUT
    - @POST
    - @DELETE
-> Spring
    - @Autowired
    - @Bean
    - @Transactional
    - @Scope
    -> Stereotypes
        - @Service
        - @Repository
        - @Component
        - @Controller
    - MVC
        -> @RestController
        -> @RequestMapping
        -> @PathVariable
        -> @RequestsParam
        -> @ModelAttribute
        -> @SessionAttributes
        -> @RequestBody/@ResponseBody
    - AOP
        -> @Aspect
        -> @Around
    - Data
        -> @Entity
        -> @Table
        -> @Column
        -> @ID
        -> @OneToOne/@OneToMany/@ManyToOne/@ManyToMany
        -> @GeneratedValue
    - Security
        -> @PreAuthorize
    - Boot
        -> @EnableAutoConfiguration
        -> @SpringBootApplication
-> Eureka
    - @EnableEurekaServer

- @EnableEurekaClient

-> Zuul

- @EnableZuulProxy
-@EnableDiscoveryClient