500 Most Important Interview Questions with Answers

Q1. What is the difference between JDK and JRE?

- JDK stands for Java Development Kit and is a full-featured software development kit for Java, including the JRE (Java Runtime Environment), an interpreter/loader (Java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc), and other tools needed for Java development.
- JRE stands for Java Runtime Environment and provides the libraries, Java Virtual Machine (JVM), and other components to run applications and applets written in Java.

Q2. Why do people say that Java is a 'write once and run anywhere' language?

- This phrase is used because Java is designed to be portable, meaning that Java code can run on any device that has a compatible JVM, without the need for recompilation. This is achieved through the use of bytecode, which is compiled from Java code and can be interpreted on any platform by the JVM.

Q3. Do you think 'main' used for the main method is a keyword in Java?

- No, 'main' is not a keyword in Java. It's an identifier chosen by convention as the name of the method that the JVM looks for as the starting point of the program.

Q4. Can we write the main method as public void static instead of public static void?

- No, the main method must be declared as 'public static void'. This is the signature the JVM expects as the entry point of the application. The 'void' keyword indicates that the main method does not return any value.

Q5. In Java, if we do not specify any value for local variables, then what will be the default value of the local variables?

- Local variables in Java do not have a default value. If you try to use a local variable without initializing it, the compiler will give you an error, forcing you to initialize it before you can use it.

Q6. What are the main principles of Object-Oriented Programming?

The main principles of Object-Oriented Programming (OOP) are Encapsulation, Abstraction, Inheritance, and Polymorphism.

- Encapsulation is the mechanism of restricting access to some of the object's components, which means that the internal representation of an object can't be seen from outside of the objects' definition. Access to this data is typically through a public interface.
- Abstraction means dealing with the level of complexity by hiding unnecessary details from the user. This can be achieved using abstract classes and interfaces.
- Inheritance is the process where one class acquires the properties and behaviors of another class.
- Polymorphism allows objects to be treated as instances of their parent class rather than their actual class. The specific implementation that gets executed is determined at runtime.

Q7. In Java, what is the default value of an object reference defined as an instance variable in an Object?

- The default value of an object reference variable is null if it is defined as an instance variable in a class and not explicitly initialized.

Q8. Why do we need a constructor in Java?

- A constructor in Java is needed to create new objects and initialize them with default or specific values. It’s called when an instance of a class is created and is used to set up the initial state of the object.

Q9. Why do we need a default constructor in Java classes?

- A default constructor is provided by Java if no other constructors are provided. It’s needed so that an instance of the class can be created with default settings. If a class doesn't have any explicit constructor, the Java compiler will automatically provide a no-argument constructor, known as the default constructor.

Q10. What is the value returned by the Constructor in Java?

- Constructors do not return values and do not have a return type, not even void. They are only used for initializing new objects.

Q11. What is the purpose of the 'this' keyword in Java?

- The this keyword in Java is a reference to the current object, the one that is being used to call the method or constructor. It is used to differentiate between class attributes and parameters with the same name, or to pass the current instance to another method or constructor.

Q12. Which class in Java is the superclass of every other class?

- The Object class is the root of the class hierarchy in Java. Every class has Object as a superclass, all objects, including arrays, implement the methods of this class.

Q13. Why does Java not support multiple inheritance?

- Java does not support multiple inheritance to avoid the complexity and ambiguity caused by it, such as the "diamond problem," where a class inherits the same method from multiple superclasses. However, Java allows multiple inheritance of types, which can be achieved by implementing multiple interfaces.

Q14. What is the purpose of the 'super' keyword in Java?

- The super keyword in Java is used to refer to the immediate parent class of the current class. It can be used to call the superclass's methods and constructors.

Q15. In Java, why do we use a static variable?

- A static variable is shared among all instances of a class; it belongs to the class rather than a particular object. This means that only one instance of a static field exists regardless of how many times the class has been instantiated. Static variables are often used to store common properties of all objects (for instance, a company name that's the same for all employees).

Q16. What happens when the static modifier is not mentioned in the signature of the main method?

- If the static modifier is omitted from the main method signature, the JVM will not be able to invoke the main method without creating an instance of the class, which goes against the Java specification for program entry points. The program will compile, but the JVM will throw an error when it attempts to run the application, saying that it cannot find the main method with the correct signature.

Q17. What is the other name of Method Overloading?

- Method Overloading is also referred to as compile-time polymorphism or static polymorphism. This is because the method to execute is determined at compile-time based on the method signature.

Q18. How will you implement method overloading in Java?

- Method overloading in Java is implemented by defining multiple methods in the same class, with the same name but different parameters. They may differ in the number of parameters, the type of parameters, or both.

Q19. What kinds of argument variations are allowed in Method Overloading?

In method overloading, the argument variations allowed include:

- Different number of parameters in each overloaded method.
- Different types of parameters in each method.
- If the number of parameters is the same, at least one parameter must have a different type.

Q20. Is it allowed to overload the main() method in Java?

- Yes, the main method can be overloaded with different parameter lists. However, the JVM will always call the original main method with a single string array as an argument. Other overloaded main methods can be called like any other method with overloading.

Q21. How do we implement method overriding in Java?

- Method overriding in Java is done by creating a method in the subclass with the same name, return type, and parameters as a method in its superclass. The @Override annotation, while not required, can be used to explicitly declare an overriding method.

Q22. Are we allowed to override a static method in Java?

- Static methods cannot be overridden in the way that instance methods can be because they are not dispatched on the instance of a class, but rather the class itself. However, a static method can be hidden by another static method in a subclass, which is sometimes referred to as method hiding rather than overriding.

Q23. Is it allowed to override an overloaded method?

- Yes, an overloaded method in the superclass can be overridden in the subclass as long as it follows the overriding rules, meaning it must have the same name, return type, and parameter list.

Q24. What is an abstract class in Java?

- An abstract class in Java is a class that cannot be instantiated on its own and is declared with the abstract keyword. It can contain abstract methods—methods without an implementation—as well as implemented methods. It's used as a base class for other classes that are expected to implement the abstract methods.

Q25. Is it allowed to mark a method abstract without marking the class abstract?

- No, if a class contains an abstract method, then the class itself must also be declared as abstract. You cannot have an abstract method in a non-abstract class.

Q26. Is it allowed to mark a method abstract as well as final?

- No, a method cannot be both abstract and final. An abstract method is one that is declared without an implementation (without braces, and followed by a semicolon), meaning it must be overridden in a subclass. A final method cannot be overridden by subclasses, so the two modifiers are mutually exclusive.

Q27. Can we instantiate an abstract class in Java?

- No, you cannot instantiate an abstract class directly. However, you can instantiate an anonymous class that extends the abstract class, or you can instantiate a subclass that provides implementations for the abstract class's abstract methods.

Q28. What is an interface in Java?

- An interface in Java is an abstract type that is used to specify a behaviour that classes must implement. Interfaces are declared using the interface keyword, and they can contain constant declarations and method signatures, which are by default abstract and public.

Q29. Is it allowed to mark an interface method as static?

- Yes, starting with Java 8, you can define static methods in interfaces which can be called independently of any object. These are not abstract methods and must provide a body.

Q30. Why can an Interface not be marked as final in Java?

- An interface cannot be marked as final because the very purpose of an interface is to be implemented by classes. Marking an interface as final would contradict its intended use as it would prevent any class from implementing it.

Q31. What is a marker interface?

- A marker interface is an interface with no methods or fields declaration; its sole purpose is to convey metadata about a class. Examples from the Java standard library include Serializable and Cloneable.

Q32. What is the difference between an abstract class and an interface in Java?

- An abstract class can have instance methods that implement a default behavior. Interfaces can only declare methods (until Java 8, after which they can also have default and static methods). A class can inherit from just one abstract class, but it can implement multiple interfaces.

Q33. Does Java allow us to use private and protected modifiers for variables in interfaces?

- Before Java 9, variables in interfaces are implicitly public, static, and final. Since Java 9, you can have private methods, which means you can encapsulate common code between default methods, but you cannot have private or protected variables.

Q34. How can you change the value of a final variable in Java?

- Once a final variable has been assigned a value, it cannot be changed. If the final variable is a reference, this means the reference cannot change, but the object it points to can be modified if it is mutable.

Q35. Can a class be marked final in Java?

- Yes, a class can be marked final. A final class cannot be subclassed, which is useful when creating an immutable class or securing a class against modification.

Q36. How can we prohibit inheritance in Java?

- Inheritance can be prohibited by marking a class as final, which means no class can extend it.

Q37. Is it allowed to declare the main method as final?

- Yes, you can declare the main method as final. This will prevent any subclass from hiding the main method, though it's generally not a concern since the main method is static and often resides in a final class or a class that is not intended to be subclassed.

Q38. Is it mandatory to import java.lang package every time?

- No, it is not mandatory to import the java.lang package as it is imported automatically by the Java compiler for every program.

Q39. Can you import the same package or class twice in your class?

- Yes, you can technically import the same package or class more than once, but it has no effect. The Java compiler is smart enough to ignore duplicate imports.

Q40. What is serialization?

- Serialization is the process of converting an object into a byte stream to save an object's state to a file or send it over the network, which can later be restored through deserialization.

Q41. What is Deserialization?

- Deserialization is the reverse process of serialization, where a byte stream is converted back into an object. This is typically used to retrieve the object's state from a file or a network.

Q42. What is transient?

- The transient keyword in Java is used to indicate that a field should not be serialized. When an object is serialized, transient fields are skipped and not included in the serialized representation.

Q43. What is Garbage Collection in Java?

- Garbage Collection (GC) in Java is the process of automatically freeing memory by destroying objects that are no longer reachable or used in the program. This helps in managing memory efficiently and reduces the chances of memory leaks.

Q44. Why do we use the finalize() method in Java?

- The finalize() method is called by the garbage collector before an object is destroyed. It can be used to perform any cleanup operations such as releasing resources (like closing files or database connections) before the object is garbage collected.

Q45. What is the meaning of Immutable in the context of the String class in Java?

- In Java, an immutable object is one that cannot be modified after it is created. The String class is an example of an immutable class. Once a String object is created, its value cannot be changed. Any method that seems to modify a String actually creates a new String object.

Q46. How many ways are there in Java to create a String object?

- There are two ways to create a String object in Java:
    - By using a string literal: String str = "Hello";
    - By using the new keyword: String str = new String("Hello");

Q47. What is the basic difference between a String and StringBuffer object?

- The main difference is that String objects are immutable, while StringBuffer objects are mutable. This means that StringBuffer objects can be modified after they are created, which is useful for scenarios where you need to make frequent modifications to a string of characters.

Q48. How will you create an immutable class in Java?

To create an immutable class in Java:

- Declare the class as final so it can't be extended.
- Make all fields private and final.
- Do not provide setter methods.
- Ensure that methods that modify the state of the object return a new object instead of modifying the current one.
- If the class has mutable object fields, ensure that these are not directly exposed or modified.

Q49. In Java, what are the differences between Checked and Unchecked Exceptions?

- Checked exceptions are checked at compile-time and must be either caught in a try-catch block or declared to be thrown by the method using the throws keyword. Examples include IOException and SQLException.
- Unchecked exceptions are not checked at compile-time and do not need to be declared or caught. They are a subclass of RuntimeException, and common examples include NullPointerException and ArrayIndexOutOfBoundsException.

Q50. What is the base class for Error and Exception classes in Java?

- The base class for both Error and Exception classes in Java is the Throwable class.

Q51. What is a finally block in Java?

- A finally block in Java is a block that is used to execute important code such as closing resources (e.g., files, database connections) regardless of whether an exception is thrown or caught. It is always executed after the try and catch blocks, even if an unexpected exception occurs.

Q52. Can we create a finally block without creating a catch block?

- Yes, a finally block can be used without a catch block. In such cases, the finally block will be executed after the try block completes. However, if an exception is thrown in the try block and not caught, it will still propagate after the finally block is executed.

Q53. Do we have to always put a catch block after a try block?

- No, it's not mandatory to have a catch block after a try block. You can have either a catch block, a finally block, or both after a try block. However, if you don't handle an exception with a catch block, you should at least provide a finally block to clean up resources.

Q54. In what scenarios, a finally block will not be executed?

- A finally block will not be executed if the JVM exits unexpectedly (e.g., System.exit() is called) or if a fatal error occurs that causes the process to abort.

Q55. What is the difference between throw and throws in Java?

- throw is a keyword used to explicitly throw an exception from a method or any block of code.
- throws is a keyword used in the method signature to declare that the method might throw one or more exceptions.

Q56. What is the difference between Collection and Collections Framework in Java?

- The Collection is an interface in Java that represents a group of objects known as its elements.
- The Collections Framework is a set of classes and interfaces that implement commonly reusable collection data structures and algorithms, which includes interfaces like List, Set, and Map, and classes like ArrayList, LinkedList, HashSet, and HashMap.

Q57. What is the root interface of the Collection hierarchy in Java?

- The root interface of the Collection hierarchy in Java is the Collection interface itself. It is at the top of the collection hierarchy and is extended by other collection interfaces like List, Set, and Queue.

Q58. How will you convert a List to a Set?

- You can convert a List to a Set in Java by passing the list as an argument to the constructor of a Set implementation, like HashSet. For example:

```
List<Integer> list = Arrays.asList(1, 2, 3, 4);
Set<Integer> set = new HashSet<>(list);
```

Q59. What are the differences between the two data structures: a Vector and an ArrayList?

Both Vector and ArrayList are implementations of the List interface. The main differences are:

- Vector is synchronized, meaning it is thread-safe, while ArrayList is not synchronized and is not thread-safe.
- Vector has a legacy design and is considered slower than ArrayList because of its synchronized methods.
- ArrayList is generally preferred over Vector for non-threaded applications due to its better performance.

Q60. In which scenario, LinkedList is better than ArrayList in Java?

- LinkedList is better than ArrayList when you need faster insertion and deletion operations, as these operations are generally O(1) in LinkedList and O(n) in ArrayList. LinkedList is also better when you need to frequently add or remove elements from the beginning or middle of the list.

Q61. What are the differences between a List and Set collection in Java?

List:

- Allows duplicate elements.
- Maintains the order of insertion.
- Examples: ArrayList, LinkedList.

Set:

- Does not allow duplicate elements.
- Does not guarantee the order of elements (except in some implementations like LinkedHashSet).
- Examples: HashSet, TreeSet, LinkedHashSet.

Q62. What are the differences between a HashSet and TreeSet collection in Java?

HashSet:

- Uses a hash table for storage.
- Does not maintain any order of elements.
- Offers constant time performance for basic operations (add, remove, contains) under ideal conditions.

TreeSet:

- Uses a red-black tree for storage.
- Maintains a sorted order of elements .
- Offers logarithmic time performance for basic operations.

Q63. In Java, how will you decide when to use a List, Set, or a Map collection?

- List: Use when you need an ordered collection that allows duplicates.
- Set: Use when you need a collection that does not allow duplicates and order is not important (or use LinkedHashSet if insertion order matters).
- Map: Use when you need to store key-value pairs and want fast retrieval based on keys.

Q64. What are the differences between a HashMap and a Hashtable in Java?

HashMap:

- Not synchronized (not thread-safe).
- Allows one null key and multiple null values.
- Iterators are fail-fast.

Hashtable:

- Synchronized (thread-safe).
- Does not allow null keys or null values.
- Enumerations are not fail-fast.

Q65. How does hashCode() method work in Java?

- The hashCode() method in Java returns an integer hash code for an object, which is used in hashing-based collections like HashMap, HashSet, and Hashtable to bucket objects. The method is designed to distribute objects evenly across the buckets for efficient retrieval.

Q66. Is it a good idea to use Generics in collections?

- Yes, using generics in collections is a good idea because it provides compile-time type safety, reducing the risk of ClassCastException. It also makes the code more readable and eliminates the need for type casting when retrieving elements.

Q67. How will you copy elements from a Source List to another list?

You can use the addAll method to copy elements from one list to another. For example:

```
List<Integer> sourceList = Arrays.asList(1, 2, 3);
List<Integer> destinationList = new ArrayList<>();
destinationList.addAll(sourceList);
```

Q68. What is the difference between an Iterator and ListIterator in Java?

Iterator:

- Can be used with any collection.
- Supports forward iteration only.
- Provides methods hasNext(), next(), and remove().

ListIterator:

- Can be used with lists only.
- Supports both forward and backward iteration.
- Provides additional methods like hasPrevious(), previous(), nextIndex(), previousIndex(), add(), and set().

Q69. What is the reason for overriding equals() method?

- The equals() method is overridden to define the equality of instances of a class based on their content rather than their memory addresses. This is especially important for classes used in collections that rely on equality checks (e.g., keys in a HashMap).

Q70. How will you reverse a List in Java?

You can use the Collections.reverse() method to reverse a list in Java. For example:

```
List<Integer> list = Arrays.asList(1, 2, 3);
Collections.reverse(list);
```

Q71. What are the main differences between HashMap and ConcurrentHashMap in Java?

HashMap:

- Not thread-safe.
- Allows one null key and multiple null values.
- Iterators are fail-fast.

ConcurrentHashMap:

- Thread-safe without locking the entire map.
- Does not allow null keys or null values.
- Provides better concurrency by partitioning the map.

Q72. Why does the Map interface not extend the Collection interface in Java?

- The Map interface does not extend the Collection interface because it represents a mapping of keys to values, which is a different data structure compared to collections that store individual elements. Maps have unique keys, and each key maps to a single value, which is a different concept compared to collections like lists and sets.

Q73. What is the contract of hashCode() and equals() methods in Java?

- The contract between hashCode() and equals() methods is:
    - If two objects are equal according to the equals() method, then calling the hashCode() method on each of the two objects must produce the same integer result.
    - If two objects are not equal according to the equals() method, there is no requirement on the hashCode() method, but it is desirable for the method to produce distinct integer results for distinct objects to improve the performance of hash tables.

Q74. How can you make a Collection class read-only in Java?

You can make a collection class read-only by using the unmodifiable wrappers provided by the Collections class. For example:

```
List<Integer> list = new ArrayList<>(Arrays.asList(1, 2, 3));
List<Integer> readOnlyList = Collections.unmodifiableList(list);
```

Q75. When is UnsupportedOperationException thrown in Java?

- The UnsupportedOperationException is thrown when an unsupported operation is attempted on a collection that does not support that operation. For example, attempting to add an element to an unmodifiable collection will throw this exception.

Q76. Let's say there is a Customer class. We add objects of the Customer class to an ArrayList. How can we sort the Customer objects in ArrayList by using the customer firstName attribute of the Customer class?

You can sort the Customer objects in the ArrayList by using the Collections.sort() method with a custom comparator that compares the firstName attribute of the Customer class. For example:

```
Collections.sort(customerList, new Comparator<Customer>() {
 @Override
 public int compare(Customer c1, Customer c2) {
 return c1.getFirstName().compareTo(c2.getFirstName());
 }
});
```

Q77. Can you explain how HashMap works in Java?

- A HashMap in Java works by using a hash table. When you put a key-value pair into the map, the key's hash code is used to determine where to store the entry in the underlying array. If two keys have the same hash code or different hash codes that result in the same index after hashing, a collision occurs, and the entries are stored in a linked list or tree at that index. When retrieving a value, the same hash code is used to find the correct index and then the list or tree is searched for the entry with the matching key.

Q78. What is the default priority of a thread in Java?

- The default priority of a thread in Java is 5, which is the constant Thread.NORM_PRIORITY.

Q79. What are the three different priorities that can be set on a Thread in Java?

The three different priorities that can be set on a thread in Java are:

- Thread.MIN_PRIORITY (value 1)
- Thread.NORM_PRIORITY (value 5)
- Thread.MAX_PRIORITY (value 10)

Q80. Is it possible to call run() method instead of start() on a thread in Java?

- Yes, it is possible to call the run() method directly instead of start() on a thread in Java. However, doing so will not create a new thread of execution; instead, the run() method will be executed in the current thread, just like any other method call.

Q81. Can we start a thread two times in Java?

- No, once a thread has been started, it cannot be started again. If you attempt to start a thread that has already been started, a java.lang.IllegalThreadStateException will be thrown.

Q82. In Java, is it possible to lock an object for exclusive use by a thread?

- Yes, in Java, it is possible to lock an object for exclusive use by a thread using the synchronized keyword. When a thread enters a synchronized method or block, it acquires the lock on the specified object, and no other thread can enter a synchronized method or block on the same object until the lock is released.

Q83. What is the life cycle of a Thread?

The life cycle of a thread in Java includes the following states:
New: The thread is created but not yet started.
Runnable: The thread is ready to run and is waiting for CPU time.
Running: The thread is executing its run method.

Blocked: The thread is blocked waiting for a monitor lock.
Waiting: The thread is waiting indefinitely for another thread to perform a particular action.
Timed Waiting: The thread is waiting for a specified amount of time.
Terminated: The thread has completed its execution and is terminated.

Q84. What is synchronization in Java?

- Synchronization in Java is a mechanism to control access to shared resources by multiple threads. It is used to ensure that only one thread can access a particular resource at a time, preventing data inconsistency and thread interference. This is typically achieved using the synchronized keyword.

Q85. Can you explain how HashMap works in Java?

- A HashMap in Java works by using a hash table. When you put a key-value pair into the map, the key's hash code is used to determine where to store the entry in the underlying array. If two keys have the same hash code or different hash codes that result in the same index after hashing, a collision occurs, and the entries are stored in a linked list or tree at that index. When retrieving a value, the same hash code is used to find the correct index and then the list or tree is searched for the entry with the matching key.

Q86. What is the difference between final, finally, and finalize in Java?

final: A keyword that can be applied to variables, methods, and classes. A final variable cannot be reassigned, a final method cannot be overridden, and a final class cannot be subclassed.
finally: A block that is used to execute important code such as closing resources, regardless of whether an exception is thrown or caught. It is always executed after the try and catch blocks.
finalize: A method that is called by the garbage collector before an object is destroyed. It is used to perform any cleanup operations such as releasing resources.

Q87. How do you ensure thread safety in a Java application?

- Use synchronization to control access to shared resources.
- Use thread-safe classes provided by the Java Collections Framework, such as ConcurrentHashMap.
- Use atomic variables provided by the java.util.concurrent.atomic package.
- Use lock objects provided by the java.util.concurrent.locks package.

Q88. What is the purpose of the volatile keyword in Java?

- The volatile keyword is used to indicate that a variable's value will be modified by different threads. It ensures that the value of the variable is always read from and written to main memory, providing visibility of changes to all threads.

Q89. Can you explain the difference between shallow copy and deep copy in Java?

- Shallow copy: Creates a new object that is a copy of the original object, but the fields are not copied; instead, both objects share the same field references.
- Deep copy: Creates a new object that is a copy of the original object, and all fields of the original object are also copied, so the new object is completely independent of the original.

Q90. What is the significance of the toString() method in Java?

- The toString() method returns a string representation of an object. It is often overridden in classes to provide a meaningful description of the object, which is useful for debugging and logging.

Q91. How do you handle exceptions in a Java application?

- Use try-catch blocks to catch and handle exceptions.
- Use the throws keyword in the method signature to declare that a method might throw an exception.
- Use the throw keyword to explicitly throw an exception.

Q92. What are the differences between a Checked Exception and an Unchecked Exception in Java?

- Checked Exception: Must be either caught in a try-catch block or declared to be thrown by the method using the throws keyword. Examples include IOException and SQLException.
- Unchecked Exception: Do not need to be declared or caught. They are a subclass of RuntimeException, and common examples include NullPointerException and ArrayIndexOutOfBoundsException.

Q93. Can you explain the concept of autoboxing and unboxing in Java?

- Autoboxing: The automatic conversion of primitive types (e.g., int) to their corresponding wrapper class objects (e.g., Integer) by the Java compiler.
- Unboxing: The automatic conversion of wrapper class objects to their corresponding primitive types.

Q94. What is the use of the super keyword in Java?

- The super keyword is used to refer to the immediate parent class of the current class. It can be used to call the superclass's methods and constructors.

Q95. How do you implement a Singleton pattern in Java?

A Singleton pattern ensures that a class has only one instance and provides a global point of access to it. It can be implemented by:

- Making the constructor private.
- Creating a private static instance of the class.
- Providing a public static method that returns the instance.

Q96. What is the difference between an abstract class and an interface in Java?

- An abstract class can have instance methods that implement a default behavior, whereas interfaces can only declare methods (until Java 8, after which they can also have default and static methods). A class can inherit from only one abstract class but can implement multiple interfaces.

Q97. How can you create an immutable object in Java?

To create an immutable object in Java:

- Declare the class as final so it can't be extended.
- Make all fields private and final.
- Do not provide setter methods.
- Ensure that methods that modify the state of the object return a new object instead of modifying the current one.
- If the class has mutable object fields, ensure that these are not directly exposed or modified.

Q98. What are the benefits of using Generics in Java?

- Generics provide compile-time type safety, reducing the risk of ClassCastException. They also make the code more readable and eliminate the need for type casting when retrieving elements from collections.

Q99. Can you explain the concept of the Java Memory Model?

- The Java Memory Model (JMM) defines how threads interact with memory in a Java application. It ensures visibility of changes to variables between threads and establishes rules for reordering of instructions to ensure consistent and predictable behavior in concurrent programs.

Q100. What is the use of the transient keyword in Java?

- The transient keyword is used to indicate that a field should not be serialized when an object is converted to a byte stream. This is useful for fields that contain sensitive information or are derived from other fields and do not need to be persisted.

These Are the most important and asked interview questions in java,After learning concepts come here and check whether you are good at answering this questions!!!

Next,we are moving to important coding questions asked in interviews.

Q101. How do you reverse a string in Java?

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello";
        char[] chars = str.toCharArray();
        for (int i = chars.length - 1; i >= 0; i--) {
            System.out.print(chars[i] + " ");
        }
    }
}
```

Q102. How do you determine if a string is a palindrome?

```
public class Main {
    public static void main(String[] args) {
        String str = "madam";
        String reversed = "";
        for (int i = str.length() - 1; i >= 0; i--) {
            reversed =reversed + str.charAt(i);
        }
        boolean isPalindrome = str.equals(reversed);
        System.out.println("Is palindrome: " + isPalindrome);
    }
}
```

Q103. Find the number of occurrences of a character in a String?

```
public class Main {
 public static void main(String[] args) {
 String str = "Hello";
 char ch = 'l';
 int count = 0;
 for (int i = 0; i < str.length(); i++) {
 if (str.charAt(i) == ch) {
 count++;
 }
```

```
 }
 System.out.println("Number of occurrences: " + count);
 }
}
```

Q104. How to find out if the given two strings are anagrams or not?

```
public class Main {
 public static void main(String[] args) {
 String str1 = "listen";
 String str2 = "silent";
 char[] chars1 = str1.toCharArray();
 char[] chars2 = str2.toCharArray();
 Arrays.sort(chars1);
 Arrays.sort(chars2);
 boolean isAnagram = Arrays.equals(chars1, chars2);
 System.out.println("Are anagrams: " + isAnagram);
 }
}
```

Q105. How do you calculate the number of vowels and consonants in a String?

```
public class Main {
 public static void main(String[] args) {
 String str = "Hello World";
 int vowels = 0, consonants = 0;
 for (int i = 0; i < str.length(); i++) {
 char ch = Character.toLowerCase(str.charAt(i));
 if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch ==
'u'){
 vowels++;
 } else if (ch >= 'a' && ch <= 'z') {
 consonants++;
 }
 }
 System.out.println("Vowels: " + vowels + ", Consonants: " +
consonants);
 }
}
```

Q106. How do you get the matching elements in an integer array?

```
public class Main {
 public static void main(String[] args) {
 int[] arr1 = {1, 2, 3, 4, 5};
 int[] arr2 = {3, 4, 5, 6, 7};
 Set<Integer> set = new HashSet<>();
 for (int num : arr1) {
 set.add(num);
```

```
 }
 System.out.print("Matching elements: ");
 for (int num : arr2) {
 if (set.contains(num)) {
 System.out.print(num + " ");
 }
 }
 }
}
```

Q107. How do you reverse an array?

```
public class Main {
    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 4, 5};
        System.out.print("Reversed array: ");
        for (int i = arr.length - 1; i >= 0; i--) {
            System.out.print(arr[i] + " ");
        }
    }
}
```

Q108. How would you swap two numbers without using a third variable?

```
public class Main {
 public static void main(String[] args) {
 int a = 5, b = 10;
 System.out.println("Before swap: a = " + a + ", b = " + b);
 a = a + b;
 b = a - b;
 a = a-b;
 System.out.println("After swap: a = " + a + ", b = " + b);
 }
}
```

Q109. Print a Fibonacci series using recursion?

```
public class Main {
 public static int fibonacci(int n) {
 if (n <= 1) return n;
 return fibonacci(n - 1) + fibonacci(n - 2);
 }

 public static void main(String[] args) {
 int n = 10; // Number of terms in the Fibonacci series
 for (int i = 0; i < n; i++) {
```

```
 System.out.print(fibonacci(i) + " ");
 }
 }
}
```

Q110. How do you find the factorial of an integer?

```
public class Main {
    public static void main(String[] args) {
        int n = 5;
        int factorial = 1;
        for (int i = 1; i <= n; i++) {
            factorial=factorial*i;
        }
        System.out.println("Factorial of " + n + " is " +
factorial);
    }
}
```

Q111. How would you find the second largest number in an array?

```
public class Main {
    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 4, 5};
        // Manual sorting (Bubble Sort)
        for (int i = 0; i < arr.length; i++) {
            for (int j = 0; j < arr.length - 1 - i; j++) {
                if (arr[j] > arr[j + 1]) {
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            }
        }
//The second largest number is the second last element in the sorted array
        int secondLargest = arr[arr.length - 2];
        System.out.println("Second largest number: " +
secondLargest);
    }
}
```

Q112. How do you remove all occurrences of a given character from the input string?

```
public class Main {
 public static void main(String[] args) {
 String str = "Hello World";
 char ch = 'o';
 String result = "";
 for (int i = 0; i < str.length(); i++) {
 if (str.charAt(i) != ch) {
```

```
 result += str.charAt(i);
 }
 }
 System.out.println("Result: " + result);
 }
}
```

Q113. Showcase Inheritance with the help of a program?

```
class Animal {
 void eat() {
 System.out.println("Eating...");
 }
}

class Dog extends Animal {
 void bark() {
 System.out.println("Barking...");
 }
}

public class Main {
 public static void main(String[] args) {
 Dog d = new Dog();
 d.eat();
 d.bark();
 }
}
```

Q114. Explain overloading and overriding with the help of a program?

```
class Overload {
 void demo(int a) {
 System.out.println("a: " + a);
 }

 void demo(int a, int b) {
 System.out.println("a and b: " + a + "," + b);
 }
}

class Override {
 void eat() {
 System.out.println("Animal is eating");
 }
}

class DogOverride extends Override {
 @Override
```

```
 void eat() {
 System.out.println("Dog is eating");
 }
}

public class Main {
 public static void main(String[] args) {
 Overload obj = new Overload();
 obj.demo(10);
 obj.demo(10, 20);

 Override a = new Override();
 DogOverride d = new DogOverride();
 a.eat();
 d.eat();
 }
}
```

Q115. How do you check if the given number is prime?

```
public class Main {
    public static void main(String[] args) {
        int n = 29;
        boolean isPrime = true;
        if (n <= 1) {
            isPrime = false;
        } else {
            for (int i = 2; i <= Math.sqrt(n); i++) {
                if (n % i == 0) {
                    isPrime = false;
                    break;
                }
            }
        }
        System.out.println(n + " is prime: " + isPrime);
    }
}
```

Q116. How do you sum all the elements in an array?

```
public class Main {
 public static void main(String[] args) {
 int[] arr = {1, 2, 3, 4, 5};
 int sum = 0;
 for (int num : arr) {
 sum += num;
 }
```

```
 System.out.println("Sum of array elements: " + sum);
 }
}
```

Q117. Write a Java program to check if a vowel is present in a String.

```
public class Main {
 public static void main(String[] args) {
 String str = "Hello World";
 boolean hasVowel = false;
 for (int i = 0; i < str.length(); i++) {
 char ch = Character.toLowerCase(str.charAt(i));
 if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch ==
'u') {
 hasVowel = true;
 break;
 }
 }
 System.out.println("Contains vowel: " + hasVowel);
 }
}
```

Q118. How do you implement a binary search in Java?

```
public class Main {
 public static int binarySearch(int[] arr, int key) {
 int low = 0, high = arr.length - 1;
 while (low <= high) {
 int mid = (low + high) / 2;
 if (arr[mid] == key) {
 return mid;
 } else if (arr[mid] < key) {
 low = mid + 1;
 } else {
 high = mid - 1;
 }
 }
 return -1;
 }

 public static void main(String[] args) {
 int[] arr = {1, 2, 3, 4, 5};
 int key = 3;
 int result = binarySearch(arr, key);
 if (result != -1) {
 System.out.println("Element found at index " + result);
 } else {
 System.out.println("Element not found");
```

```
 }
 }
}
```

Q119. Write a Java program that sorts HashMap by value.

```
import java.util.*;

public class Main {
 public static void main(String[] args) {
 HashMap<String, Integer> map = new HashMap<>();
 map.put("One", 1);
 map.put("Two", 2);
 map.put("Three", 3);

 List<Map.Entry<String, Integer>> list = new
ArrayList<>(map.entrySet());
 list.sort(Map.Entry.comparingByValue());

 System.out.println("Sorted HashMap by value: ");
 for (Map.Entry<String, Integer> entry : list) {
 System.out.println(entry.getKey() + ": " + entry.getValue());
 }
 }
}
```

Q120. Can you prove that a String object in Java is immutable programmatically?

```
public class Main {
 public static void main(String[] args) {
 String str1 = "Hello";
 String str2 = str1;
 str1 = str1 + " World";

 System.out.println("str1: " + str1);
 System.out.println("str2: " + str2);
 System.out.println("Is String immutable: " + !str1.equals(str2));
 }
}
```

Q121. How do you illustrate a try-catch example in Java?

```
public class Main {
 public static void main(String[] args) {
 try {
 int result = 10 / 0;
 System.out.println("Result: " + result);
 } catch (ArithmeticException e) {
```

```
 System.out.println("ArithmeticException caught: " +
e.getMessage());
 }
 }
}
```

Q122. How do you create a functional interface?

```
@FunctionalInterface
interface MyFunctionalInterface {
 void execute();
}

public class Main {
 public static void main(String[] args) {
 MyFunctionalInterface myFunc = () ->
System.out.println("Executing functional interface method");
 myFunc.execute();
 }
}
```

Q123. Write a program to remove duplicate elements from an array in Java.

```
import java.util.*;

public class Main {
 public static void main(String[] args) {
 int[] arr = {1, 2, 2, 3, 4, 4, 5};
 Set<Integer> set = new LinkedHashSet<>();
 for (int num : arr) {
 set.add(num);
 }
 int[] uniqueArr = new int[set.size()];
 int i = 0;
 for (int num : set) {
 uniqueArr[i++] = num;
 }
 System.out.println("Array without duplicates: " +
Arrays.toString(uniqueArr));

}
```

Q124. How do you swap the first and last elements of an array in Java?

```
public class Main {
 public static void main(String[] args) {
 int[] arr = {1, 2, 3, 4, 5};
```

```
 int temp = arr[0];
 arr[0] = arr[arr.length - 1];
 arr[arr.length - 1] = temp;

 System.out.println("Array after swapping first and last elements:
");
 for (int num : arr) {
 System.out.print(num + " ");
 }
 }
}
```

Q125. How do you check if a string contains only digits in Java?

```
public class Main {
 public static void main(String[] args) {
 String str = "12345";
 boolean isNumeric = true;
 for (int i = 0; i < str.length(); i++) {
 if (!Character.isDigit(str.charAt(i))) {
 isNumeric = false;
 break;
 }
 }
 System.out.println("String contains only digits: " + isNumeric);
 }
}
```

126. What is a Servlet ?

The servlet is a Java programming language class used to process client requests and generate dynamic web content. Servlets are mostly used to process or store data submitted by an HTML form, provide dynamic content and manage state information that does not exist in the stateless HTTP protocol.

127. Explain the architecture of a Servlet.

The core abstraction that must be implemented by all servlets is the javax.servlet.Servlet interface. Each servlet must implement it either directly or indirectly, either by extending javax.servlet.GenericServlet or javax.servlet.http.HTTPServlet. Finally, each servlet is able to serve multiple requests in parallel using multithreading.

128. What is the difference between an Applet and a Servlet ?

An Applet is a client side java program that runs within a Web browser on the client machine. On the other hand, a servlet is a server side component that runs on the web server.An applet can use the user interface classes, while a servlet does not have a user interface. Instead, a servlet waits for client's HTTP requests and generates a response in every request.

129. What is the difference between GenericServlet and HttpServlet ?

GenericServlet is a generalized and protocol-independent servlet that implements the Servlet and ServletConfig interfaces. Those servlets extending the GenericServlet class shall override the service method. Finally, in order to develop an HTTP servlet for use on the Web that serves requests using the HTTP protocol, your servlet must extend the HttpServlet instead. Check Servlet examples here.

130. Explain the life cycle of a Servlet.

On every client's request, the Servlet Engine loads the servlets and invokes its init methods, in order for the servlet to be initialized. Then, the Servlet object handles all subsequent requests coming from that client, by invoking the service method for each request separately. Finally, the servlet is removed by calling the server's destroy method.

131. What is the difference between doGet() and doPost() ?

doGET: The GET method appends the name-value pairs on the request's URL. Thus, there is a limit on the number of characters and subsequently on the number of values that can be used in a client's request. Furthermore, the values of the request are made visible and thus, sensitive information must not be passed in that way. doPOST: The POST method overcomes the limit imposed by the GET request, by sending the values of the request inside its body. Also, there is no limitations on the number of values to be sent across. Finally, the sensitive information passed through a POST request is not visible to an external client.

132. What is meant by a Web Application ?

A Web application is a dynamic extension of a Web or application server. There are two types of web applications: presentation-oriented and service-oriented. A presentation-oriented Web application generates interactive web pages, which contain various types of markup language and dynamic content in response to requests. On the other hand, a service-oriented web application implements the endpoint of a web service. In general, a Web application can be seen as a collection of servlets installed under a specific subset of the server's URL namespace.

133. What is a Server Side Include (SSI) ?

Server Side Includes (SSI) is a simple interpreted server-side scripting language, used almost exclusively for the Web, and is embedded with a servlet tag. The most frequent use of SSI is to include the contents of one or more files into a Web page on a Web server. When a Web page is accessed by a browser, the Web server replaces the servlet tag in that Web page with the hyper text generated by the corresponding servlet.

134. What is Servlet Chaining ?

Servlet Chaining is the method where the output of one servlet is sent to a second servlet. The output of the second servlet can be sent to a third servlet, and so on. The last servlet in the chain is responsible for sending the response to the client.

135. How do you find out what client machine is making a request to your servlet ?

The ServletRequest class has functions for finding out the IP address or host name of the client machine. getRemoteAddr() gets the IP address of the client machine and getRemoteHost() gets the host name of the client machine.

136. What is the structure of the HTTP response ?

The HTTP response consists of three parts: Status Code: describes the status of the response. It can be used to check if the request has been successfully completed. In case the request failed, the status code can be used to find out the reason behind the failure. If your servlet does not return a status code, the success status code, HttpServletResponse.SC_OK, is returned by default. HTTP Headers: they contain more information about the response. For example, the headers may specify the date/time after which the response is considered stale, or the form of encoding used to safely transfer the entity to the user. See how to retrieve headers in Servlet here. Body: it contains the content

of the response. The body may contain HTML code, an image, etc. The body consists of the data bytes transmitted in an HTTP transaction message immediately following the headers.

137. What is a cookie ? What is the difference between session and cookie ?

A cookie is a bit of information that the Web server sends to the browser. The browser stores the cookies for each Web server in a local file. In a future request, the browser, along with the request, sends all stored cookies for that specific Web server.The differences between session and a cookie are the following: The session should work, regardless of the settings on the client browser. The client may have chosen to disable cookies. However, the sessions still work, as the client has no ability to disable them in the server side. The session and cookies also differ in the amount of information the can store. The HTTP session is capable of storing any Java object, while a cookie can only store String objects.

138. Which protocol will be used by browser and servlet to communicate ?

The browser communicates with a servlet by using the HTTP protocol.

139. What is HTTP Tunneling ?

HTTP Tunneling is a technique by which, communications performed using various network protocols are encapsulated using the HTTP or HTTPS protocols. The HTTP protocol therefore acts as a wrapper for a channel that the network protocol being tunneled uses to communicate. The masking of other protocol requests as HTTP requests is HTTP Tunneling.

140. What's the difference between sendRedirect and forward methods ?

The sendRedirect method creates a new request, while the forward method just forwards a request to a new target. The previous request scope objects are not available after a redirect, because it results in a new request. On the other hand, the previous request scope objects are available after forwarding. FInally, in general, the sendRedirect method is considered to be slower compare to the forward method.

141. What is URL Encoding and URL Decoding ?

The URL encoding procedure is responsible for replacing all the spaces and every other extra special character of a URL, into their corresponding Hex representation. In correspondence, URL decoding is the exact opposite procedure.

142. What is a JSP Page ?

A Java Server Page (JSP) is a text document that contains two types of text: static data and JSP elements. Static data can be expressed in any text-based format, such as HTML or XML. JSP is a technology that mixes static content with dynamically-generated content. See JSP example here.

143. How are the JSP requests handled ?

On the arrival of a JSP request, the browser first requests a page with a .jsp extension. Then, the Web server reads the request and using the JSP compiler, the Web server converts the JSP page into a servlet class. Notice that the JSP file is compiled only on the first request of the page, or if the JSP file has changed.The generated servlet class is invoked, in order to handle the browser's request. Once the execution of the request is over, the servlet sends a response back to the client. See how to get Request parameters in a JSP.

144. What are the advantages of JSP ?

The advantages of using the JSP technology are shown below: JSP pages are dynamically compiled into servlets and thus, the developers can easily make updates to presentation code. JSP pages can be pre-compiled. JSP pages can be easily combined to static templates, including HTML or XML fragments, with code that generates dynamic content. Developers can offer custom JSP tag libraries that page authors access using an XML-like syntax. Developers can make logic changes at the component level, without editing the individual pages that use the application's logic.

145. What are Directives ? What are the different types of Directives available in JSP ?

Directives are instructions that are processed by the JSP engine, when the page is compiled to a servlet. Directives are used to set page-level instructions, insert data from external files, and specify custom tag libraries. Directives are defined between <%@and%>.The different types of directives are shown below: Includedirective: it is used to include a file and merges the content of the file with the current page. Pagedirective: it is used to define specific attributes in the JSP page, like error page and buffer. Taglib: it is used to declare a custom tag library which is used in the page.

146. What are JSP actions ?

JSP actions use constructs in XML syntax to control the behavior of the servlet engine. JSP actions are executed when a JSP page is requested. They can be dynamically inserted into a file, re-use JavaBeans components, forward the user to another page, or generate HTML

for the Java plugin.Some of the available actions are listed below: jsp:include – includes a file, when the JSP page is requested. jsp:useBean – finds or instantiates a JavaBean. jsp:setProperty – sets the property of a JavaBean. jsp:getProperty – gets the property of a JavaBean. jsp:forward – forwards the requester to a new page. jsp:plugin – generates browser-specific code.

147. What are Scriptlets ?

In Java Server Pages (JSP) technology, a scriptlet is a piece of Java-code embedded in a JSP page. The scriptlet is everything inside the tags. Between these tags, a user can add any valid scriptlet.

148. What are Declarations ?

Declarations are similar to variable declarations in Java. Declarations are used to declare variables for subsequent use in expressions or scriptlets. To add a declaration, you must use the sequences to enclose your declarations.

149. What are Expressions ?

A JSP expression is used to insert the value of a scripting language expression, converted into a string, into the data stream returned to the client, by the web server. Expressions are defined between <%=and%> tags.

150. What is meant by implicit objects and what are they ?

JSP implicit objects are those Java objects that the JSP Container makes available to developers in each page. A developer can call them directly, without being explicitly declared. JSP Implicit Objects are also called pre-defined variables.The following objects are considered implicit in a JSP page: application page request response session exception out config pageContext.

151. What is JDBC?

JDBC (Java Database Connectivity) is an API that allows Java applications to interact with databases using SQL.

152. What are the main interfaces of JDBC?

The main interfaces of JDBC are the Driver Manager, Driver, Connection, Statement, and ResultSet.

153. Which Java package contains the DriverManager class, and what is its purpose?

The DriverManager class is found in the java.sql package.

154. What is a JDBC Connection?

A JDBC Connection is an interface that represents a session with a specific database, enabling SQL execution and transaction management.

155. How do you create a JDBC connection?

You create a JDBC connection using DriverManager.getConnection(url, user, password), where url is the database URL, and user and password are the database username and password.

156. Why do we need a MySQL Connector jar?

MySQL Connector/J is the official JDBC driver for MySQL, allowing Java applications to connect and interact with MySQL databases. We should add the MySQL jar file in the project's classpath or by adding it as a dependency in build tools like Maven or Gradle.

157. What is the JDBC URL format for MySQL?

The JDBC URL format for MySQL is jdbc:mysql://hostname:port/dbname, for example, jdbc:mysql://localhost:3306/mydb.

158. What is a PreparedStatement in JDBC?

A PreparedStatement is a precompiled SQL statement that allows you to execute parameterized queries, improving performance and security.

159. What is the difference between Statement and PreparedStatement?

Statement is used for executing simple SQL queries, while PreparedStatement is used for precompiled, parameterized queries, reducing SQL injection risks and improving efficiency.

160. What is ResultSet in JDBC?

ResultSet is an interface that represents the result set of a query, allowing retrieval of data from the database row by row.

161. What is a CallableStatement in JDBC?

CallableStatement is used to execute stored procedures in the database, supporting IN, OUT, and INOUT parameters.

162. Which interface is responsible for transaction management in JDBC?

The Connection interface provides methods for transaction management such as commit(), rollback() etc.

163. What is the most common exception in JDBC and how do you handle it?

`SQL exception` is the most common exception in JDBC. It is handled using try-catch blocks, specifically catching `SQLException` to manage database errors.

164. What is batch processing in JDBC?

Batch processing in JDBC allows executing multiple SQL statements as a batch, reducing the number of database hits and improving performance.

165. How do you use batch processing in JDBC?

Batch processing is used by adding SQL statements to a batch with addBatch() and executing them with executeBatch() on a Statement or PreparedStatement.

166. What is the JDBC ResultSetMetaData interface?

The ResultSetMetaData interface returns the information of the table such as the total number of columns, column name, column type, etc.

167. What is the major difference between java.util.Date and java.sql.Date data type?

The major difference between java.util.Date and java.sql.Date is that, java.sql.Date represents date without time information whereas, java.util.Date represents both date and time information.

168. Explain the usage of the getter and setter methods in ResultSet.

Getter methods are used to get specific table column values from the ResultSet. Either the column name or the column index value should be given as a parameter.

Setter methods are used to set the value in the database. It is almost identical to getter methods, but in this case, you must also give the name of the column or its index value in addition to the data or values for the specific column you want to insert into the database.

169. What is SQL injection, and how can it be prevented?

SQL injection is a security vulnerability that occurs when an attacker can insert or manipulate SQL queries in an application's database layer, often due to improper input sanitization. This can lead to unauthorized access or manipulation of data.

170. Out of statement and prepared statement which is safe from SQL injection?

Prepared statement provides protection against SQL injection as it has parameterized queries which are pre compiled.

171. What is Hibernate?

Hibernate is an ORM (Object-Relational Mapping) framework for Java that simplifies database interactions by mapping Java objects to database tables.

172. Explain important interfaces you come across in hibernate?

Hibernate comprises many interfaces such as Configuration, SessionFactory, Session, Transaction, etc.

173. What is a Hibernate Session?

Hibernate Session is a single-threaded, short-lived object representing a conversation between the application and the database.

174. Explain the difference between get() and load() methods in Hibernate.

The get() method fetches data from the database immediately, while the load() method fetches it lazily and throws an exception if the object doesn't exist.

175. What is the purpose of the Hibernate Configuration File (hibernate.cfg.xml)?

The hibernate.cfg.xml file contains database configurations, such as connection settings and Hibernate properties, and mappings between Java classes and database tables.

176. What is HQL (Hibernate Query Language)?

HQL is an object-oriented query language in Hibernate, similar to SQL but operates on persistent objects instead of tables.

177. What are the different states of an object in Hibernate?

Hibernate objects can be in one of three states: transient, persistent, or detached, depending on their lifecycle and session association.

178. What is the purpose of the @Entity annotation in Hibernate?

The @Entity annotation marks a Java class as a persistent entity, making it eligible for mapping to a database table**.**

179. What is a primary key and how is it defined in Hibernate?
A primary key uniquely identifies a record in a table. In Hibernate, it is defined using the @Id annotation and can be auto-generated with @GeneratedValue.

180. What is the use of the @Table annotation in Hibernate?

The @Table annotation specifies the table in the database to which the entity is mapped, allowing customization of the table name and schema.

181. What is the difference between openSession() and getCurrentSession() methods in Hibernate?

openSession() always opens a new session, while getCurrentSession() returns the current session bound to the context (typically used in transaction management).

182. Can you explain what lazy loading is in hibernate?

Lazy loading is mainly used for improving the application performance by helping to load the child objects on demand. It is to be noted that, since Hibernate 3 version, this feature has been enabled by default. This signifies that child objects are not loaded until the parent gets loaded.

183. What is a Criteria Query in Hibernate?

A Criteria Query is a type-safe, object-oriented API for constructing and executing queries in Hibernate without the need for HQL.

184. How does the second-level cache differ from the first-level cache in Hibernate?

The first-level cache is session-specific and exists only for the duration of a Hibernate session, while the second-level cache is shared across sessions and persists as long as the SessionFactory is alive.

185. How do you enable the second-level cache in Hibernate?

To enable the second-level cache, you need to configure it in the Hibernate configuration file (`hibernate.cfg.xml`) and specify a cache provider, such as Ehcache, Infinispan, or another supported provider.

186. Mention some of the advantages of using ORM over JDBC.

ORM has the following advantages over JDBC:

- You need not mention primary key as it can be auto generated
- Transaction is managed by ORM

SQL query is auto generated by ORM
Development becomes fast as lot of boilerplate code is readily available

187. Explain about three states of the object in hibernate?

There are 3 states of the object (instance) in hibernate.

1.Transient: The object is in a transient state if it is just created but has no primary key (identifier) and is not associated with a session.
2.Persistent: The object is in a persistent state if a session is open, and you just saved the instance in the database or retrieved the instance from the database.
3.Detached: The object is in a detached state if a session is closed. After the detached state, the object comes to a persistent state if you call lock() or update() method.

188. Is session thread safe?

No, Session is not a thread-safe object which means that any number of threads can access data from it simultaneously.

189. What is a one-to-one mapping in Hibernate?

A one-to-one mapping in Hibernate establishes a relationship where one entity instance is associated with exactly one instance of another entity. It is typically implemented using the `@OneToOne` annotation.
Ex:- One User can have only One Aadhar details

190. What is the difference between unidirectional and bidirectional one-to-one mapping?

In unidirectional one-to-one mapping, only one entity knows about the relationship, while in bidirectional mapping, both entities are aware and can navigate the relationship, typically using `@OneToOne` on both sides.

191. What is a one-to-many mapping in Hibernate?

A one-to-many mapping in Hibernate establishes a relationship where one entity instance is associated with multiple instances of another entity. It is typically implemented using the `@OneToMany` annotation.
Ex:- One person can own many vehicles

192. What is a many-to-one mapping in Hibernate?

A many-to-one mapping in Hibernate is the inverse of a one-to-many relationship, where multiple instances of one entity are associated with a single instance of another entity, often

representing a parent-child relationship. It is implemented using the `@ManyToOne` annotation.
Ex:- Many vehicles can be owned by one person.

193. What is a many-to-many mapping in Hibernate?

A many-to-many mapping in Hibernate represents a relationship where multiple instances of one entity are associated with multiple instances of another entity. It is typically implemented using the `@ManyToMany` annotation.
Ex:- Multiple students taking part in multiple courses and vice versa.

194. How do you implement a many-to-many mapping in Hibernate?

Implement a many-to-many mapping by using the `@ManyToMany` annotation on a collection field in both entities, along with `@JoinTable` to define the join table and foreign key columns. The `@JoinTable` annotation is used to define the join table and the foreign key columns in many-to-many mappings, specifying how the relationship is represented in the database.

195. What is the difference between `FetchType.LAZY` and `FetchType.EAGER`?

`FetchType.LAZY` delays the loading of the associated entities until they are accessed, while `FetchType.EAGER` loads the associated entities immediately with the parent entity.

196. What is the Spring Framework?

The Spring Framework is an open-source framework for building Java applications, providing comprehensive infrastructure support including dependency injection, aspect-oriented programming, and transaction management.

197. Explain the concept of Dependency Injection (DI).

Dependency Injection (DI) is a design pattern where an object's dependencies are provided externally rather than the object creating them itself, promoting loose coupling.

198. Name a few Spring sub projects you know?

Some of the most important sub projects in spring are : Spring core, JDBC, ORM, web, MVC, Security, AOP

199. How do you define a Spring Bean using XML configuration?

Define a Spring Bean in XML using `<bean>` tags with attributes such as `id`, `class`, and `scope`, and optionally configure properties and dependencies.

200. What are the different types of Dependency Injection?

The main types are Constructor Injection, Setter Injection, and Field Injection.

201. Which Is the Best Way of Injecting Beans and Why?

The recommended approach is to use constructor arguments for mandatory dependencies and setters for optional ones. This is because constructor injection allows injecting values to immutable fields and makes testing easier.

202. What is a Spring Bean?

A Spring Bean is an object that is instantiated, managed, and configured by the Spring IoC (Inversion of Control) container.

203. What is Inversion of Control (IoC)?

Inversion of Control (IoC) is a design principle in which the control of object creation and dependency management is transferred from the application code to a framework or container. In Spring, IoC is implemented through Dependency Injection (DI), where the Spring container is responsible for managing the lifecycle and dependencies of beans, allowing for more flexible and decoupled code.

204. What Is the Default Bean Scope in Spring Framework?

By default, a Spring Bean is initialised as a singleton.

205. What is a BeanFactory?

BeanFactory is the core interface in Spring that provides the mechanism to create and manage beans, offering basic functionalities for dependency injection and bean lifecycle management.

206. What is an ApplicationContext?

ApplicationContext is an extension of BeanFactory that provides additional functionalities, such as event propagation, declarative mechanisms, and integration with Spring's various features.

207. Name a few classes that extend Application context.

ClassPathApplicationContext which can be used while working with XML and AnnotationConfigApplicationContext can be used while working with java configuration.

208. What are the different scopes of Spring Beans?

Bean scopes include `singleton`, `prototype`, `request`, `session`, and `globalSession`.

209. What is the singleton scope in Spring?

In the `singleton` scope, only one instance of the bean is created and shared across the entire Spring container.

210. What is the prototype scope in Spring?

In the `prototype` scope, a new instance of the bean is created each time it is requested from the container.

211. Are Singleton Beans Thread-Safe?

No, singleton beans are not thread-safe, as thread safety is about execution, whereas the singleton is a design pattern focusing on creation. Thread safety depends only on the bean implementation itself.

212. What Is the Spring Java-Based Configuration?

It’s one of the ways of configuring Spring-based applications in a type-safe manner. It’s an alternative to the XML-based configuration.

213. What is autowiring in spring? What are the autowiring modes?

Autowiring enables the programmer to inject the bean automatically. We don't need to write explicit injection logic.
Ex:- <bean id="student" **class**="com.flm.Student" autowire="byName" />

214. What are different modes of autowiring? Explain them.

Different modes of autowiring are byType, byName and constructor

byName: injects the bean based on the property name. It uses the setter method.
byType: injects the bean based on the property type. It uses the setter method.
Constructor: it injects the bean using constructor.

215. What is the @Autowired annotation used for?

The `@Autowired` annotation is used for automatic dependency injection, allowing Spring to resolve and inject dependencies into a bean's properties, constructors, or methods.

216. What is the purpose of the @Qualifier annotation?

The `@Qualifier` annotation is used to specify which bean should be injected when multiple beans of the same type exist, providing finer control over dependency injection.

217. What is the @Bean annotation used for?

The @Bean annotation is used in Java-based configuration to define a bean and its lifecycle within the Spring container.

218. What are the advantages of JdbcTemplate in spring?

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

219. How do you execute a simple query using JdbcTemplate?

Use JdbcTemplate's queryForObject or query method to execute a query. For example, jdbcTemplate.queryForObject("SELECT name FROM users WHERE id = ?", new Object[]{id}, String.class).

220. What is the role of the `RowMapper` interface in Spring JDBC?

The `RowMapper` interface is used to map rows of a ResultSet to Java objects, allowing custom mapping of query results to entity classes.

221. What is NamedParameterJdbcTemplate?

NamedParameterJdbcTemplate is an extension of JdbcTemplate that supports named parameters in SQL queries, providing more readable and maintainable code.

222. How do you configure a `DataSource` in Spring?

Configure a `DataSource` using XML or Java-based configuration, defining properties such as URL, username, and password, and optionally setting connection pooling options.

223. What is Spring ORM?

Spring ORM (Object-Relational Mapping) is a module in the Spring Framework that provides integration with ORM frameworks like Hibernate, JPA, and MyBatis, simplifying data access and transaction management.

224. What is `HibernateTemplate`?

`HibernateTemplate` is a class provided by the Spring Framework that simplifies interactions with Hibernate by handling common tasks like opening and closing sessions, managing transactions, and error handling.

225. How do you define a `SessionFactory` bean in Spring?

Define a `SessionFactory` bean in a Spring configuration class using `LocalSessionFactoryBean` or XML configuration, specifying properties such as data source, dialect, and annotated classes.

226. What are the core components of Spring MVC?

The core components are the DispatcherServlet, Controllers, Models, Views, and ViewResolvers, which work together to handle and process web requests.

227. How does the `DispatcherServlet` work in Spring MVC?

`DispatcherServlet` is the front controller that receives all incoming requests, delegates them to appropriate handlers (controllers), and resolves views to render the response.

228. What is the role of `@Controller` annotation in Spring MVC?

The `@Controller` annotation is used to define a class as a Spring MVC controller, responsible for handling web requests and returning a model and view or a view name.

229. What is the purpose of `@RequestMapping` annotation?

`@RequestMapping` maps HTTP requests to handler methods in controllers, allowing you to specify request paths, methods, and parameters for routing requests.

230. How do you configure view resolvers in Spring MVC?

Configure view resolvers using `@Bean` methods in a configuration class or in the `applicationContext.xml` file, specifying the view prefix and suffix to resolve logical view names to actual view files.

231. What is `ModelAndView` and how is it used?

`ModelAndView` is a class that holds both the model data and the view name, allowing controllers to return both model attributes and the logical view name in a single return object.

232. What is @ResponseBody used for in Spring MVC?

`@ResponseBody` indicates that the return value of a controller method should be written directly to the HTTP response body, often used for RESTful APIs.

233. What is the Spring Web module?

The Spring Web module provides functionalities for building web applications, including support for web MVC, REST APIs, and web-related utilities like multipart file uploads and initialization.

234. What Is Spring WebFlux?

Spring WebFlux is Spring's reactive-stack web framework, and it's an alternative to Spring MVC.
In order to achieve this reactive model and be highly scalable, the entire stack is non-blocking.

235. What are Spring Data JPA repositories?

Spring Data JPA repositories are interfaces that provide CRUD operations and custom queries for JPA entities, reducing boilerplate code and simplifying data access.

236. How do you create a custom query in Spring Data JPA?

Create a custom query using the `@Query` annotation on a repository method, specifying the JPQL or SQL query to be executed.

237. What is Spring AOP (Aspect-Oriented Programming)?

Spring AOP is a programming paradigm that enables modularization of cross-cutting concerns, such as logging and transaction management, by separating them from business logic through the use of aspects.

238. What is an aspect in Spring AOP?

An aspect is a module that defines cross-cutting concerns and encapsulates advice, which can be applied to multiple join points throughout an application.

239. What are join points in Spring AOP?

Join points are specific points in the execution of a program, such as method calls or object instantiations, where aspects can be applied.

240. What is advice in Spring AOP?

Advice is the action taken by an aspect at a particular join point. It can be pre-processing, post-processing, or around the execution of the join point.

241. What are the different types of advice in Spring AOP?

Types of advice include `@Before` (executed before a join point), `@After` (executed after a join point), `@AfterReturning` (executed after a successful join point), `@AfterThrowing` (executed if a join point throws an exception), and `@Around` (wraps the join point).

242. How do you define a pointcut in Spring AOP?

Define a pointcut using expressions with annotations like `@Pointcut` or within the `@Aspect` class, specifying which join points the advice should apply to.

243. What is Spring Security?

Spring Security is a framework that provides comprehensive security services for Java applications, including authentication, authorization, and protection against common security vulnerabilities.

244. What is the role of the `SecurityContextHolder` in Spring Security?

`SecurityContextHolder` holds the security context, which contains information about the currently authenticated user and their authorities, allowing for access control decisions.

245. What is the `@PreAuthorize` annotation in Spring Security?

The `@PreAuthorize` annotation allows for method-level security by evaluating SpEL (Spring Expression Language) expressions to determine if the method should be executed based on user roles or other conditions.

246. How do you implement authentication in Spring Security?

Implement authentication by configuring an `AuthenticationManager`, defining user details with `UserDetailsService`, and setting up authentication mechanisms such as form-based login or JWT.

247. What is the role of `UserDetailsService` in Spring Security?

`UserDetailsService` is an interface used to retrieve user-related data by implementing the `loadUserByUsername` method, providing user information for authentication and authorization.

248. How does Spring Security handle authorization?

Spring Security handles authorization by using access control mechanisms, such as method security annotations (`@Secured`, `@PreAuthorize`), URL-based security rules, and role-based access controls.

249. What are security filters and how do they work in Spring Security?

Security filters are components that intercept HTTP requests and responses to apply security measures, such as authentication and authorization. They are managed by the `FilterChain` and configured in the security filter chain.

250. What is CSRF protection and how is it configured in Spring Security?

CSRF (Cross-Site Request Forgery) protection prevents unauthorized actions from being performed on behalf of authenticated users. It is enabled by default in Spring Security and can be configured or disabled using `csrf()` in the security configuration.

251. What is CORS (Cross-Origin Resource Sharing)?

CORS is a security feature implemented by web browsers that allows or restricts resources on a web page to be requested from another domain outside the domain from which the resource originated.

252. What is Spring Boot and what are its main features?

Spring Boot is an extension of the Spring Framework that simplifies the development of stand-alone, production-ready Spring applications. Its main features include auto-configuration, starter dependencies, embedded servers, and simplified configuration.

253. How does Spring Boot simplify the configuration of Spring applications?

Spring Boot simplifies configuration by providing sensible defaults and auto-configuration based on the classpath, beans, and property settings, reducing the need for manual setup.

254. What is a Spring Boot starter, and how is it used?

A Spring Boot starter is a dependency module that includes a set of commonly used dependencies for a specific functionality, like `spring-boot-starter-web` for web applications. They simplify dependency management.

255. What is the purpose of the `@SpringBootApplication` annotation?

`@SpringBootApplication` is a convenience annotation that combines `@Configuration`, `@EnableAutoConfiguration`, and `@ComponentScan`, setting up a Spring Boot application with sensible defaults.

256. What is the `@RestController` annotation?

`@RestController` is a convenience annotation that combines `@Controller` and `@ResponseBody`, indicating that the class is a web controller and its methods return JSON or XML data instead of views.

257. What is the purpose of the `@RequestParam` annotation in Spring Web?

`@RequestParam` is used to bind request parameters to method parameters in a controller, allowing you to extract query parameters or form data from HTTP requests..

258. How do you use `@PathVariable` in Spring Web?

`@PathVariable` is used to extract values from URI templates, allowing dynamic segments of the URL to be mapped to method parameters, e.g., `@GetMapping("/users/{id}")`.

259. What is the role of `application.properties` or `application.yml` in a Spring Boot project?

These files are used to define configuration properties for the application, such as database connection details, server settings, and custom application-specific settings.

260. How do you define custom properties in Spring Boot and access them?

Custom properties are defined in `application.properties` or `application.yml` and can be accessed using `@Value` annotation or by binding them to POJOs using `@ConfigurationProperties`.

261. What is Spring Boot's auto-configuration and how does it work?

Auto-configuration is a mechanism in Spring Boot that automatically configures Spring application components based on the dependencies present on the classpath and the application's settings.

262. What are embedded servers in Spring Boot, and why are they useful?

Embedded servers, like Tomcat or Jetty, allow Spring Boot applications to run as standalone applications without needing an external server. They simplify deployment and development.

263. How do you change the default port of a Spring Boot application?

Change the default port by setting the property `server.port` in `application.properties` or `application.yml`.

264. What is a Spring Boot profile, and how do you use it?

A Spring Boot profile is a way to segregate parts of the application configuration and beans, allowing them to be activated based on the environment (e.g., development, production). Use `@Profile` annotation or `spring.profiles.active` property.

265. What is a `CommandLineRunner` in Spring Boot?

`CommandLineRunner` is an interface used to execute code after the Spring Boot application has started. It provides a `run` method where the startup logic can be implemented.

266. How can you create a RESTful web service using Spring Boot?

Create a RESTful web service by defining REST controllers using `@RestController` and mapping HTTP requests to handler methods using `@RequestMapping` or `@GetMapping`, `@PostMapping`, etc.

267. How does Spring Boot support data access, and what are Spring Data JPA and Spring Data REST?

Spring Boot supports data access through Spring Data JPA, which simplifies database interactions, and Spring Data REST, which automatically exposes JPA repositories as RESTful endpoints.

268. How do you deploy a Spring Boot application?

Deploy a Spring Boot application as an executable JAR or WAR file on a server, or run it as a standalone application using an embedded server. Cloud and container deployments are also supported.

269. How does Spring Boot support testing, and what is `@SpringBootTest`?

Spring Boot supports testing through annotations like `@SpringBootTest`, which loads the application context and provides an environment for testing Spring components.

270. What is the use of `spring-boot-devtools`?

`spring-boot-devtools` is a Spring Boot module that provides development-time features such as automatic restarts, live reload, and configurations specific to development environments.

271. How do you configure logging in a Spring Boot application?

Configure logging using properties in `application.properties` or `application.yml`, specifying the logging level, format, and output destinations. Spring Boot supports various logging frameworks like Logback, Log4j2, and Java Util Logging.

272. What is the role of `@ExceptionHandler` in Spring Web?

`@ExceptionHandler` is used to define methods in a controller or `@ControllerAdvice` class that handle specific exceptions thrown by controller methods, providing custom error responses.

273. How do you set up a custom error page in Spring Web?

Set up custom error pages by configuring `ErrorController` or by adding entries in `application.properties` or `web.xml` to map error codes to specific view pages.

274. How do you use `@Conditional` annotations in Spring Boot?

Use `@Conditional` annotations to conditionally include beans, configuration classes, or methods based on specific conditions, such as the presence of certain classes or properties.

275. How do you perform scheduling in Spring Boot?

Perform scheduling using `@EnableScheduling` on a configuration class and `@Scheduled` on methods to execute tasks at fixed intervals, cron expressions, or with fixed delays.

276. What are microservices, and how do they differ from monolithic architecture?

Microservices are a software architecture style where applications are composed of small, independent services, each responsible for a specific business function. Unlike monolithic architectures, microservices are loosely coupled, can be developed and deployed independently, and offer better scalability and flexibility.

277. What are the key benefits of using a microservices architecture?

Benefits include improved scalability, better fault isolation, independent deployment, technology diversity, and ease of continuous delivery and integration.

278. What are some common challenges associated with microservices?

Challenges include increased complexity in managing multiple services, inter-service communication, data consistency, service discovery, load balancing, and handling distributed transactions.

279. What is Spring Cloud Gateway and what are its primary functions?

Spring Cloud Gateway is a gateway service developed by the Spring Cloud team to handle routing, monitoring, resiliency, and security for microservices. It provides a simple yet powerful way to route requests, handle cross-cutting concerns, and apply filters to incoming requests.

280. How do you configure routes in Spring Cloud Gateway?

Routes in Spring Cloud Gateway can be configured using the application.yml file or programmatically using Java configuration. Routes define how requests are matched and routed to the downstream services.

281. What are filters in Spring Cloud Gateway and how are they used?

Filters in Spring Cloud Gateway are used to modify requests and responses. They can be applied globally or to specific routes, handling tasks like authentication, logging, and modifying headers.

282. What is `RestTemplate` in Spring and what is it used for?

RestTemplate is a synchronous client to perform HTTP requests in Spring applications. It simplifies communication with HTTP servers, making it easy to consume RESTful services.

283. How can you perform a GET request with `RestTemplate`?

A GET request can be performed using methods like getForObject(), getForEntity(), or exchange() to retrieve data from a URL and map it to Java objects.

284. What is OpenFeign and how does it integrate with Spring Cloud?

OpenFeign is a declarative HTTP client developed by Netflix. It integrates with Spring Cloud to simplify the development of REST clients by using annotations to define service interfaces and their endpoints.

285. How do you define a Feign client in Spring Cloud?

A Feign client is defined using the `@FeignClient` annotation, specifying the service name and optionally a URL. Interface methods are annotated with HTTP methods like `@GetMapping` or `@PostMapping`.

286. What is the purpose of `@EnableFeignClients`?

The `@EnableFeignClients` annotation is used to enable Feign client support in a Spring Boot application, scanning for interfaces annotated with `@FeignClient` to generate implementations.

287. What is a circuit breaker in the context of microservices?

A circuit breaker is a design pattern used to protect applications from cascading failures by preventing calls to a failing service, thereby providing fallback options and improving system resilience.

288. How does the circuit breaker pattern work in Spring Cloud?

In Spring Cloud, circuit breakers can be implemented using tools like Resilience4j or Hystrix. They monitor calls to external services and open the circuit if failures exceed a threshold, returning fallback responses or errors.

289. How do you enable and configure circuit breakers in a Spring Boot application?

Enable circuit breakers by including the relevant dependencies (e.g., Resilience4j or Hystrix) and configuring them using annotations like `@CircuitBreaker`, `@EnableCircuitBreaker`, and specifying properties in configuration files.

290. What is Spring Cloud Config Server and what is its purpose?

Spring Cloud Config Server provides server-side support for externalised configuration in a distributed system. It serves configuration properties from a central location, allowing clients to fetch configurations dynamically.

**291. How do you set up a Spring Cloud Config Server? **

Set up a Config Server by creating a Spring Boot application with the `spring-cloud-config-server` dependency, and annotate the main class with `@EnableConfigServer`. Configuration can be stored in a Git repository or another supported storage.

292. What is Netflix Eureka and how does it function in a microservices architecture?

Netflix Eureka is a service registry used for service discovery. It allows microservices to register themselves at runtime as they come up and deregister themselves on shutdown, enabling dynamic discovery of service instances.

293. How do you set up a Eureka Server in a Spring Cloud application?

Set up a Eureka Server by including `spring-cloud-starter-netflix-eureka-server` and annotating the main application class with `@EnableEurekaServer`. Clients use `@EnableEurekaClient` to register with the server.

294. What is a fallback mechanism in circuit breakers, and why is it important?

> A fallback mechanism provides an alternative response or action when a service call fails or the circuit breaker is open, ensuring that the system can handle failures gracefully without complete disruption.

295. How do you monitor and manage microservices using Spring Boot Actuator?

Spring Boot Actuator provides endpoints for monitoring and managing applications, including health checks, metrics, and environment information. These endpoints can be exposed over HTTP and integrated with monitoring tools.

296. How can you implement load balancing in a Spring Cloud microservices architecture?

Load balancing can be implemented using tools like Netflix Ribbon, Spring Cloud LoadBalancer, or external solutions like Nginx. These tools distribute incoming requests across multiple instances of a service.

297. How do you handle versioning in microservices APIs using Spring Cloud?

API versioning can be managed using URI versioning, request parameters, or headers. It's essential for maintaining backward compatibility and supporting multiple API versions.

298. How do you implement security in a microservices architecture with Spring Cloud?

Security can be implemented using Spring Security for authentication and authorization, OAuth2 for delegated security, and API Gateway for centralised security measures like token validation and SSL termination.

299. What is containerization, and why is it important for microservices?

Containerization involves packaging a service and its dependencies into a container, ensuring consistency across different environments. It is crucial for microservices as it allows for easy deployment, scaling, and isolation.

300. What is the role of Docker in microservices?

Docker provides a platform for containerization, allowing microservices to be packaged and run consistently across various environments, improving deployment flexibility and resource efficiency.

SQL INTERVIEW QUESTIONS

301. What is Relational Database Management System(RDBMS)?

A. A Relational Database Management System (RDBMS) is a type of database management system that stores data in a structured format, using rows and columns. This format allows for the efficient retrieval, manipulation, and management of data. Ex:- Oracle,MySQL,PostgreSQL,DM2 etc…

302. What is Structured Query Language?

A. Structured Query Language (SQL) is a standardized programming language specifically designed for managing and manipulating relational databases. SQL is used to perform various operations such as querying data, updating records, and managing database objects.

303. What is a Database?

A. A database is a structured collection of data that is stored and accessed electronically. Databases are designed to efficiently store, retrieve, manage, and manipulate large amounts of data.

304. What is primary key?

A. Primary key is a constraint that uniquely identifies each record in a table. The primary key constraint automatically enforces uniqueness and non-nullability for the specified columns.It can be created at column level and table level.

305. What is a unique key?

A. Unique key is a type of database constraint that ensures all values in a column or a group of columns are distinct across the table. It doesnot allows duplicate values but accepts null values.

306. What is a foreign key?

A. A foreign key is used to establish a relationship between two tables. This means that the foreign key ensures that the value in one table matches a value in another table, thereby maintaining consistent and valid relationships between tables.

307. Explain the difference between spreadsheets and databases.

A. Spreadsheets and databases are both tools for storing, organizing, and managing data, but they have different features, use cases, and capabilities.
— A spreadsheet is a digital worksheet that contains cells in a grid of rows and columns, typically used for calculations, data analysis, and simple data storage.
— A database is a structured collection of data that is stored electronically and managed by a Database Management System.

308. What are table and fields?

A. Tables and fields are fundamental components that organize and structure data.
Tables:- A table is a collection of related data entries that consists of rows and columns. It is the primary structure for storing data in a relational database.
Fields:- A field is a single piece of data or an attribute within a table. Each field represents one type of data that is stored in the table.

309. Explain the various SQL languages.

A. SQL is the standard language for managing and manipulating relational databases.
SQL consists of 5 sub languages:- 1. DDL(Data Definition Language)
2. DML(Data Manipulation Language)
3. DRL/DQL(Data Retrieval/Query Language)
4. TCL(Transaction Control Language)
5. DCL(Data Control Language)

310. What is normalization?

A. Normalization is a database design process used to organize data within a database to reduce redundancy and improve data integrity. The primary goal of normalization is to structure data in such a way that it is stored efficiently and is free from undesirable anomalies, such as update, insert, and delete anomalies.

311. What is denormalization?

A. Denormalization is the process of deliberately introducing redundancy into a database design by merging tables or adding redundant data in order to optimize performance, typically for read-heavy operations.

312. Explain the different types of normalization.

A. There are 5 types of normalization. They are:

1.First Normal Form (1NF): Ensure that the table structure is organized so that each column contains only atomic (indivisible) values and each row is unique.
2. Second Normal Form (2NF): Eliminate partial dependencies, where a non-key attribute depends on only a part of a composite primary key.
3. Third Normal Form (3NF): Eliminate transitive dependencies, where a non-key attribute depends on another non-key attribute rather than just the primary key.
4. Boyce-Codd Normal Form (BCNF): Resolve anomalies not covered by 3NF, ensuring that every determinant is a candidate key.
5. Fourth Normal Form (4NF): Eliminate multi-valued dependencies, where one attribute in a table has multiple values that are independent of other attributes.

313. What are views in SQL?

A. A view is a virtual table that is defined by a query. It does not store data itself but provides a way to present data from one or more tables in a specific format.

314. What is join? Explain the different types.

A. A join is used to combine rows from two or more tables based on a related column between them. Joins are essential for querying relational databases where data is distributed across multiple tables. There are different types of joins. They are:
1.INNER JOIN: Returns rows with matching values in both tables
2.LEFT JOIN: Returns all rows from the left table, and matched rows from the right table.
3.RIGHT JOIN: Returns all rows from the right table, and matched rows from the left table.
4.FULL JOIN: Returns all rows from both tables, with NULLs for non-matching rows.
5.CROSS JOIN: Returns Cartesian product of both tables.
6.SELF JOIN: Joins table with itself

315. What are the different types of indexes?

A. Indexes in databases are used to improve the speed of data retrieval operations.Here are the main types of indexes commonly used in SQL databases:
1.B-Tree Index:The most common type of index, based on a balanced tree structure. B-Trees maintain sorted data and allow searches, insertions, and deletions in logarithmic time.
2. Hash Index: Uses a hash table where keys are hashed to a specific location. It is designed for fast exact-match queries.
3. Bitmap Index: Uses bitmaps (arrays of bits) to represent the presence of values. Each distinct value of a column is represented by a bitmap.
4. Unique Index: Ensures that the indexed column(s) contain only unique values. It automatically enforces uniqueness constraints.
5. Composite Index (Multi-Column Index): An index on two or more columns of a table. It allows for efficient querying on multiple columns.
6. Full-Text Index: Specialized index for efficient text searching. It is used for full-text search operations, allowing you to perform searches on large text fields.
7. Spatial Index: Used for indexing spatial data types (e.g., geometrical shapes, geographic coordinates). It supports spatial queries and operations.

8. Clustered Index: Defines the order in which data is physically stored in a table. A table can have only one clustered index.
9. Non-Clustered Index: Creates a separate structure from the data rows that points to the location of the actual data. A table can have multiple non-clustered indexes.

316. What is a cursor in SQL?

A. A cursor is a database object used to retrieve, manipulate, and navigate through a result set row by row. Cursors provide a way to handle query results in a more granular and procedural manner.

317. What is query?

A. A query in the context of databases is a request for data or information from a database. Queries are used to retrieve, manipulate, and interact with data stored in relational or non-relational databases.

318. What is a subquery?

A. A subquery, also known as a nested query or inner query, is a query embedded within another SQL query. It allows you to perform complex data retrieval and manipulation operations by breaking down a larger query into smaller, more manageable parts.

319. What is a trigger?

A. A trigger is a special type of stored procedure in a database that automatically executes or fires in response to certain events on a table or view. Triggers are used to enforce rules, automate tasks, and maintain data integrity. They can be set to run before or after specific events such as `INSERT`, `UPDATE`, or `DELETE` operations.

320. Differentiate between the DELETE and TRUNCATE commands.

A. DELETE Command: Removes specific rows from a table based on a `WHERE` condition. If no condition is specified, it will remove all rows from the table but not the table structure itself.
TRUNCATE Command: Removes all rows from a table, essentially resetting it to an empty state. Unlike `DELETE`, it does not use a `WHERE` condition.

321. What are constraints?

A. Constraints help keep data accurate and consistent in a database. They set rules to make sure data follows the required standards. By using constraints, you can avoid entering incorrect or inconsistent data, ensuring the database works correctly.

322. What is data integrity?

A. Data integrity means keeping data accurate, consistent, and reliable in a database. It ensures that data stays correct and trustworthy from the moment it's created, stored, retrieved, and changed.

323. What is auto increment?

A. Auto-increment is a feature in databases that automatically creates a unique number for a column, often for primary keys. It makes it easy to assign unique IDs to records without having to enter them manually.

324. What is a data warehouse?

A. A data warehouse is a central system that stores and manages large amounts of data from different sources. It helps with business intelligence tasks, like reporting and analysis, by giving a single, unified view of the data.

325. What is the difference between DROP and TRUNCATE statements?

A. DROP Statement: Completely removes a table, view, or other database objects (like indexes) from the database.
TRUNCATE Statement: Removes all rows from a table but keeps the table structure for future use.

326. What is alias in SQL?

A. An alias is a temporary name given to a table or column for the duration of a query. Aliases are used to simplify complex queries, make them more readable, and sometimes to resolve naming conflicts. They are not permanent and only exist for the duration of the query execution.

327. How can we create tables in SQL?

A. The basic syntax for creating a table is:
CREATE TABLE table_name (column1 data_type constraints, column2 data_type constraints, ... columnN data_type constraints);
Ex:- CREATE TABLE Employees (EmployeeID INT PRIMARY KEY, FirstName VARCHAR(50), LastName VARCHAR(50), HireDate DATE);

328. How can we insert data in SQL?

A. Inserting data into a table is done using the `INSERT INTO` statement. We can insert data in multiple ways.
1.Insert a Single Row: INSERT INTO table_name (column1, column2, column3, ...) VALUES (value1, value2, value3, ...);
2. Insert Multiple Rows: INSERT INTO table_name (column1, column2, column3, ...)

```
VALUES (value1a, value2a, value3a, ...),
    (value1b, value2b, value3b, ...),
    ...;
```

329. How can we change a table name in SQL?

A. Changing of a name uses the `ALTER TABLE` statement with the `RENAME TO` clause:
Syntax: ALTER TABLE old_table_name RENAME TO new_table_name;

330. What are nested queries?

A. Nested queries, also known as subqueries, are queries embedded within other queries. They allow you to perform more complex operations and retrieve data based on the results of another query. Subqueries can be used in various parts of an SQL statement, including the `SELECT`, `WHERE`, `HAVING`, and `FROM` clauses.

331. What is the difference between CHAR and VARCHAR2 data types ?

A. The `CHAR` and `VARCHAR2` data types in SQL are both used to store character data, but they have distinct characteristics and use cases. Here's a comparison of the two:
CHAR Data Type: `CHAR` is a fixed-length data type. If you specify a `CHAR(n)` column, it will always store data with a length of `n` characters.
VARCHAR2 Data Type: `VARCHAR2` is a variable-length data type. If you specify a `VARCHAR2(n)` column, it can store up to `n` characters but uses only as much space as needed for the actual data.

332. What is difference between SQL and PL/SQL?

A. SQL (Structured Query Language) and PL/SQL (Procedural Language/SQL) are both integral to working with relational databases. Here's a comparison between SQL and PL/SQL:
SQL (Structured Query Language): SQL is a standard language used for querying, updating, and managing data in relational databases. It is used for operations like retrieving data, inserting records, updating data, and deleting records.
PL/SQL (Procedural Language/SQL): PL/SQL is an extension of SQL provided by Oracle that adds procedural programming features to SQL. It allows for the creation of complex scripts, stored procedures, functions, and triggers.

333. What is the difference between SQL and MySQL?

A. SQL and MySQL are related but represent different aspects of working with databases. Here's a detailed comparison:

SQL (Structured Query Language): SQL is a standard language used for querying and managing data in relational databases. It is used to perform operations such as querying data, inserting records, updating records, and deleting data.
MySQL: MySQL is a relational database management system (RDBMS) that uses SQL as its query language. It is a specific implementation of a database system that stores, retrieves, and manages data using SQL.

334. What is cross join?

A. A `CROSS JOIN` is a type of join in SQL that produces a Cartesian product of two tables. This means that it returns all possible combinations of rows from the two tables involved. Each row from the first table is paired with every row from the second table.
Syntax: SELECT column1, column2, ...
FROM table1
CROSS JOIN table2;

335. What are user defined functions?

A. User-defined functions (UDFs) are custom functions that users create to perform specific tasks in a database. They let you add your own logic for calculations or data processing, and you can use them in SQL queries just like the built-in functions provided by the database system.

336. What is a CLAUSE?

A. A clause in SQL is a section of a SQL statement that sets a condition or specifies an action. Clauses help build queries by defining what data to get or change and how to do it. Each clause has a unique job and helps make the SQL statement work. Common SQL Clauses are: — SELECT,FROM,WHERE,ORDER BY, GROUP BY, HAVING, JOIN, LIMIT.

337. What is recursive stored procedure?

A. A recursive stored procedure is one that calls itself to solve problems by breaking them into smaller, similar problems. It's useful for tasks that fit a recursive approach, such as handling hierarchical data or performing recursive calculations.

338. Explain UNION, MINUS and INTERSECT commands?

A. UNION: Combines the results of two or more SELECT statements into a single result set. It removes duplicate rows by default.

Syntax: SELECT column1, column2, … FROM table1 UNION SELECT column1, column2, … FROM table2;

MINUS: Returns rows from the first `SELECT` statement that are not present in the second `SELECT` statement.

Syntax: SELECT column1, column2, … FROM table1 MINUS SELECT column1, column2, … FROM table2;

INTERSECT: Returns rows that are common to both `SELECT` statements.

Syntax: SELECT column1, column2, … FROM table1 INTERSECT SELECT column1, column2, … FROM table2;

339. How can we select unique records from a Table?

A. To select unique records from a table in SQL, you can use the `DISTINCT` keyword. This keyword ensures that the results of your query include only unique rows by removing duplicate records.

Ex:- SELECT DISTINCT column1, column2 FROM table_name;

340. What is the main difference in the BETWEEN and IN condition operators?

A. `BETWEEN` Operator: Filters data within a range of values. It is used to select values that fall between two specified values.

Syntax:- SELECT column1, column2 FROM table_name WHERE column_name BETWEEN value1 AND value2;

`IN` Operator: Filters data based on a set of specific values. It is used to select rows where a column matches any value within a specified list.

Syntax:- SELECT column1, column2 FROM table_name WHERE column_name IN (value1, value2, ...);

341. What is the difference between JOIN and UNION?

A. JOIN: Combines rows from two or more tables based on a related column between them. It is used to retrieve related data from multiple tables in a single query.

Syntax:- SELECT columns FROM table1 INNER JOIN table2 ON table1.common_column = table2.common_column;

UNION: Combines the results of two or more `SELECT` statements into a single result set. It is used to stack results from different queries vertically.

Syntax: SELECT column1, column2 FROM table1 UNION SELECT column1, column2 FROM table2;

342. What is the difference between order and group by?

A. ORDER BY: Sorts the result set of a query based on one or more columns. It determines the order in which rows appear in the final result.

   Syntax:- SELECT column1, column2 FROM table_name ORDER BY column1 [ASC|DESC], column2 [ASC|DESC];

   GROUP BY: Groups rows that have the same values in specified columns into summary rows. It is often used with aggregate functions (e.g., `COUNT`, `SUM`, `AVG`) to perform calculations on each group of rows.

   Syntax:- SELECT column1, aggregate_function(column2) FROM table_name GROUP BY column1;

343. What is the command used to fetch the first 5 characters of the string?

A. SELECT SUBSTR(column_name, 1, 5) AS First5Chars FROM table_name;

344. How to use LIKE in SQL?

A. The `LIKE` operator in SQL is used to search for a specified pattern in a column. It's often used with wildcard characters to match a range of values or patterns in a column.

   Syntax:- SELECT column1, column2 FROM table_name WHERE column_name LIKE pattern;

   Ex:- To find all values starting with 'A': SELECT * FROM Employees WHERE LastName LIKE 'A%';

345. Write an SQL query to fetch three max salaries from a table.

A. SELECT DISTINCT Salary FROM Employees ORDER BY Salary DESC LIMIT 3;

346. Write an SQL query to show the second highest salary from a table.

A. SELECT Salary FROM Employees ORDER BY Salary DESC LIMIT 1 OFFSET 1;

347. Can you join a table by itself?

A. Yes, you can join a table to itself in SQL, and this is known as a **self-join**. A self-join is useful when you need to compare rows within the same table or when you want to relate rows to other rows in the same table based on a certain condition.

Syntax:- SELECT a.column1, b.column2

FROM table_name a JOIN table_name b ON a.common_column = b.common_column WHERE some_condition;

348. Write an SQL query to create a new table with data and structure copied from another table.

A. CREATE TABLE new_table AS SELECT * FROM existing_table;

349. Write an SQL query to fetch employee names having a salary greater than or equal to 20000 and less than or equal to 10000.

A. SELECT EmployeeName FROM Employees WHERE Salary >= 20000 AND Salary <= 30000;

350. How would you select all the users whose phone number is NULL?

A. SELECT * FROM Users WHERE PhoneNumber IS NULL;

351.What is HTML and What does it stand for ?

HTML stands for HyperText Markup Language. It is the standard language used to create and design documents on the web.

352.What is the difference between HTML and HTML5?

HTML5 is the latest version of HTML. It introduces new elements, attributes, and behavior, improves support for multimedia, and offers better error handling, among other enhancements.

353.What is the purpose of the <meta> tag?

The <meta> tag provides metadata about the HTML document, such as character set, author, description, and keywords.

354.What is semantic HTML?

Semantic HTML is using HTML tags that convey meaning beyond just formatting to both the browser and developer, making the structure of the webpage more understandable.

355.What are HTML entities?

HTML entities are special codes used to display characters that have a special meaning in HTML, such as <,>,& etc.

356.Explain the difference between the <head> and <body> tags?

The <head> tag contains meta-information about the document, such as title, links to stylesheets, and scripts. The <body> tag contains the content that is visible on the webpage.

357.What is the difference between a local and a session storage in HTML5?

Local storage stores data with no expiration date, while session storage stores data only for the duration of the page session

358.What is a self-closing tag? Give an example?

A self-closing tag is an HTML tag that doesn't require a separate closing tag.
Example:
<img src=”image.jpg” alt=”Image”>

359.Explain the purpose of the <iframe> tag?

The <iframe> tag is used to embed another HTML document within the current document. It is commonly used to display the content from another website.

360.What is the purpose of <noscript> tag?

The <noscript> tag provides alternative content for users who have disabled scripts in their browser or whose browser does not support scripting.

361.Explain the difference between absolute and relative URLs.

Absolute URLs specify the full web address of a resource, including the protocol (http/https). Relative URLs specify the path to a resource relative to the current page.

362.What is the purpose of the `alt` attribute in images?

The alt attribute provides alternative text for an image if it cannot be displayed.

363.What is the difference between block level and inline elements?

Block-level elements start on a new line and take up the full width available (,<h1>,<p>), whereas inline elements do not start on a new line and only take up as much width as necessary

364.Explain the difference between `name` and `id` attribute in form elements?

The name attribute specifies the name of a form element, which is used to identify the data in the form submission. The `id` attribute provides a unique identifier for the element with in the document, which can be used for linking labels or styling with CSS

365.What is the purpose of the pre element in HTML?

The <pre> element preserves whitespaces and displays text in a fixed-width font.

366.Explain the purpose of the `<!DOCTYPE>` declaration in HTML?

The `<DOCTYPE>` declaration is an instruction to the web browser about the version of HTML the page is written in.

367.Explain the purpose of the `action` attribute in a form element?

The `action` attribute specifies the URL to which the form data is sent when the form is submitted.

368.What is the use of `target` attribute in anchor tags?

The `target` attribute specifies where to open the linked document.
Example: `target=_blank"` opens the link in a new tab.

369.What is the `novalidate` attribute in form elements?

The `novalidate` attribute, when added to a `<form>` element, disables the browser's default validation for that form.

370.Explain HTML5 form validations and how it works?

HTML5 form validations use attributes like `required`, `pattern`, `min`, `max`, `minlength`, `maxlength`, and type to enforce input constraints. The browser automatically checks these constraints before form submission

371.How can you set a default value for an input field in an HTML form?

Use the `value` attribute in the `<input>` element.

372.What is CSS?

CSS stands for Cascading Style Sheets. It is a stylesheet language used to describe the presentation of a document written in HTML.

373.What are the different ways to include CSS in an HTML document?

CSS can be included in an HTML document using inline styles, internal stylesheets (using the style tag), or external stylesheets (linked using the <link> tag).

374.What is the CSS Box Model?

The CSS Box Model describes the rectangular boxes generated for elements in the document tree. It consists of:

- Content: The innermost part where text and image appears
- Padding: Space between the content and the border
- Border: Surrounds the padding and content
- Margin: Space outside the border, separating the element from other elements.

375.What is the difference between `display: none` and `visibility: hidden`?

`display: none` removes the element from the document flow, meaning it takes up no space and does not affect the layout. `visibility: hidden` hides the element but still takes up space in the layout

376.Explain the difference between display: block, display: inline, and display: inline-block?

display: block makes an element a block-level element, taking up the full width available, and starting on a new line. display: inline makes an element an inline-level element, allowing it to flow within the text. display: inline-block makes an element an inline-level block container, allowing it to have block-like behavior while flowing within the text.

377.What is the purpose of the position property in CSS?

The `position` property in css is used to specify how an element is positioned in a document. It has five main values:

1. static: The default value; elements are positioned according to the normal document flow.
2. relative: The element is positioned relative to its normal position.
3. absolute: The element is positioned relative to its nearest positioned ancestor or the initial containing block, and it is removed from the normal document flow.
4. fixed: The element is positioned relative to the viewport and remains in the same position even when the page is scrolled.
5. sticky: The element switches between `relative` and `fixed` positioning depending on the scroll position.

378.Explain the difference between margin and padding?

Margin is the space outside the border of an element, while padding is the space between the element`s content and its border.

379.What is the CSS selector specificity?

CSS selector specificity determines which CSS rule takes precedence when multiple rules apply to the same element. It is calculated based on the type of selector used.

380.What are CSS pseudo-classes? Give an example?

Pseudo-classes are keywords added to selectors to specify a special state of an element

Example:

a:hover {

color: red;

}

381.What are CSS pseudo-elements? Give an example?

Pseudo-elements target specific parts of an element.

Example:

p::first-line {

font-weight: bold;

}

382.What is the `Z-index` property?

`Z-index` controls the stack order of elements positioned with `position: relative`, `position: absolute`, or `position-fixed`. Higher values are in front of lower values

383.What is the `box-sizing` property?

`box-sizing` defines how the width and height of an element are calculated. `box-sizing: border-box;` includes padding and border in the element's total width and height

384.What is the difference between `em` and `rem` units?

`em` is relative to the font-size of its parent element, while `rem` is relative to the font-size of the root element (`<html>`)

385.What is CSS sprite?

A CSS sprite is a technique used to combine multiple images into a single image file, reducing the number of server requests and improving performance.

386.What is the purpose of the @import rule in CSS?

The @import rule is used to import an external style sheet into another style sheet. It can be used to modularize stylesheets and improve organization.

387.What is the purpose of the @media rule in CSS?

The @media rule is used to apply different styles for different media types/devices, such as screen, print, or handheld devices.

388.What is the purpose of the CSS calc() function?

The calc() function allows for mathematical calculations to be performed within CSS property values. It is commonly used for dynamic sizing and positioning.

389.What is the `opacity` property?

The `opacity` property sets the transparency level of an element. Values range from `0` (full transparent) to 1 (full opaque).

390.What are `:before` and `:after` pseudo-elements used for?

`:before` and `:after` add content before or after an element's content,typically used for decorative purposes.

391.Explain the `:nth-of-type` pseudo-class?

The `:nth-of-type` pseudo-class matches the elements of specific type, based on their position among the siblings of the same type

392.What is the object-fit property used for?

The object-fit property sets how the content of a replaced element, such as an <img> or <video>, should be resized to fit its container.

393.How do you create a flex container that centers its items both vertically and horizontally?

.container {

display: flex;

justify-content: center;

align-items: center;

}

394.How do you reverse the order of flex-items in a flex-container?

Set the `flex-direction` property to `row-reverse` or `column-reverse`.

Example:

.container {

display: flex;

flex-direction: row-reverse;

}

395. Explain the difference between `justify-content: space-between` and `justify-content: space-around`.

`justify-content: space-between` distributes items evenly with the first item at the start and last item at the end. `justify-content: space-around` distributes items with equal space around them, including half the space on the edges.

396. How do you make a flex item take up the remaining space?

Use the flex-grow property with a value greater than `0`.

Example:

.item {

flex-grow: 1

}

397. How do you center a block element horizontally within its container?

Use `margin: auto;` on the block element.

Example

.block {

width: 50%;

margin: auto;

}

398.What is the purpose of the CSS grid layout model?

The CSS grid layout model is used for creating grid based layouts where elements are placed into rows and columns, allowing for more complex and flexible layouts than traditional methods.

399. What is the purpose of `grid-template-columns` and `grid-template-rows`?

`grid-template-columns` and `grid-template-rows` define the number and size of the columns and rows in a grid container, respectively.

Example:

.container {

grid-template-columns: 100px 200px 100px;

grid-template-rows: 50px 100px;

}

400. What are `grid-column` and `grid-row` properties?

`grid-column` and `grid-row` are shorthand properties for defining the start and end lines of a grid item in the columns and rows, respectively.

Example:

.item {

grid-column: 1 / 3;

grid-row: 2 / 4;

}

401.What is the purpose of the CSS transform property?

The transform property allows for the transformation of elements in two-dimensional or three-dimensional space, including scaling, rotating, skewing, and translating.

402.Explain the purpose of the CSS transition property?

Transitions enable you to define the transitions between two states of an element. Different states may be defined using pseudo-classes like :hover or :active or dynamically set using javascript.

403.Explain the concept of specificity in CSS and how it is calculated?

Specificity in CSS determines which style rule takes precedence when multiple rules apply to the same element. It is calculated based on the type of selector used, with inline styles having the highest specificity, followed by IDs, classes, and elements.

404.What are the advantages and disadvantages of using CSS preprocessors like Sass or Less?

Advantages of CSS preprocessors include the ability to use variables, mixins, nesting, and functions, which improve code maintainability and reusability. Disadvantages include the need for an additional build step and potential complexity for beginners.

405.Explain the difference between CSS transitions and CSS animations?

CSS transitions are used to smoothly change CSS property values over a specified duration, while CSS animations allow for more complex and customized animations by defining keyframes and animation properties.

406.What are vendor prefixes in CSS, and why are they used?

Vendor prefixes are prefixes added to CSS property names to specify which browser engine the property applies to. They are used to provide experimental or vendor-specific implementations of CSS features before they are standardized.

407.What is the purpose of the CSS specificity hack?

The CSS specificity hack is a technique used to increase the specificity of a selector to override styles applied by other selectors with lower specificity.

408.Explain the purpose of the CSS @supports rule?

The @supports rule checks if the browser supports a specific CSS property and value combination, allowing for the conditional application of styles.

409.Explain the concept of responsive web design and how CSS is used to achieve it?

Responsive web design is an approach to design and development that ensures web pages render well on various devices and window or screen sizes. CSS media queries are used to apply different styles based on the device`s characteristics, such as width, height and orientation.

410.What is the purpose of the CSS perspective-origin property?

The perspective-origin property specifies the origin point of the perspective for 3D transforms, determining where the viewer is looking at the 3D scene.

411.What is JavaScript?

JavaScript is a high-level, interpreted programming language primarily used for creating interactive web pages.

412.What are different data types in JavaScript?

The primary data types in javaScript are:

Primitive: String, Number, Boolean, Null, Undefined, Symbol, BigInt.

Non Primitive: Object, Array, Function.

413.What is the difference between `let`, `var`, and `const`?

`var` is function-scoped and can be re-declared and updated.
`let` is block-scoped and can be updated but not re-declared within the same scope.
`const` is block-scoped and cannot be updated or re-declared.

414.What is closure in JavaScript?

A closure is a function that remembers its lexical scope even when the function is executed outside that lexical scope

Example:

```
function outer() {
  let counter=0;
  return function inner() {
    counter++;
    console.log(counter);
  }
};
const increment= outer();
increment();  //1
increment();  //2
```

415.Explain hoisting in JavaScript?

Hoisting is JavaScript`s default behavior of moving declarations to the top of the current scope. This means variable and function declarations are processed before any code is executed.

416.What are arrow functions?

Arrow functions provide a concise syntax for writing function expressions. They do not have their own `this` context and cannot be used as constructors.

417.What are higher-order functions in JavaScript?

Higher-order functions are functions that operate on other functions, either by taking them as arguments or by returning them.

418.What is a callback function?

A callback function is a function passed as an argument to another function, which is then invoked inside the outer function to complete some kind of routine or action.

419.What is the purpose of the `typeof` operator?

The `typeof` operator returns a string indicating the type of the unevaluated operand.

420.What are template literals?

Template literals allow embedded expressions and multiline strings using backticks (```);

421.What is event delegation in JavaScript?

Event delegation is a technique where a single event listener is attached to a parent element to listen for events that occur on its children. This is particularly useful when dealing with dynamically added elements.

422.Explain the concept of prototypal inheritance in JavaScript?

Prototypal inheritance is the mechanism by which objects in JavaScript inherit properties and methods from other objects, known as prototypes.

423.What is `this` keyword in JavaScript?

The `this` keyword refers to the object on which a method is being invoked or the context in which a function is called.

424.How does JavaScript handle asynchronous operations?

JavaScript uses mechanisms such as callbacks, promises, and async/await to handle asynchronous operations.

425.What are JavaScript promises?

Promises are objects representing the eventual completion or failure of an asynchronous operation, and they allow for more readable and manageable asynchronous code.

426.Explain the difference between `null` and `undefined` in JavaScript?

• `null` is an assigned value indicating no value.

• `undefined` means a variable has been declared but not assigned a value.

427.What is a “rest parameter” in JavaScript?

The rest parameter syntax allows a function to accept an indefinite number of arguments as an array.

428.What is the difference between a `for` loop and a `for…in` loop?

• `for` loop: Iterates over elements of an array.
• `for…in` loop: Iterates over the enumerable properties of an object.

429.What is the difference between `Function Declaration` and `Function Expression`?

`Function Declaration`: Defines a named function.
`Function Expression`: Defines a function inside an expression.

430.What is the difference between `typeof` and `instanceof`?

• `typeof`: Returns a string indicating the type of the operand.
• `instanceof`: Tests whether the prototype property of a constructor appears anywhere in the prototype chain of an object.

431.What are ES6 modules?

ES6 modules are a standard for modular JavaScript code. They use `export` and `import` statements to share code between files.

432.What is the async keyword?

The `async` keyword is used to declare asynchronous functions. An `async` function returns a promise, and the `await` keyword is used to pause the execution of the function until the promise is resolved.

433.What is “Event Loop”?

The event loop is a mechanism that javascript runtime uses to manage the execution of multiple pieces of code, handling events, and executing queued tasks.

434.What is the difference between `==` and `===`?

`==` performs type coercion and checks for equality in value, while `===` checks for both value and type equality without type coercion.

435.Explain the concept of prototypal inheritance and how it differs from classical inheritance?

Prototypal inheritance is a mechanism in JavaScript where objects inherit properties and methods from other objects through prototype chains. It differs from classical inheritance, which involves the creation of classes and instances, by being more flexible and dynamic.

436.What are closures and how can they lead to memory leaks in JavaScript?

Closures are functions that have access to their outer function`s scope even after the outer function has finished executing. They can lead to memory leaks if they hold references to large objects or variables that are no longer needed, preventing them from being garbage collected.

437.Explain the concept of `bind, call`, and `apply` methods in JavaScript?

`bind`,`call`, and `apply` are methods used to manipulate the `this` keyword in JavaScript functions. `bind` creates a new function with a specified `this` value, `call` calls a function with a specified `this` value and arguments passed individually, and `apply` calls a function with a specified `this` value and arguments passed as an array.

438.What is event bubbling and event capturing in JavaScript?

Event bubbling is a mechanism where an event occurring in a child element is propagated up to its parent elements, while event capturing is the reverse process where the event is captured from the top of the DOM hierarchy down to the target element.

439.What is the difference between synchronous and asynchronous code?

• Synchronous: Code is executed sequentially, blocking further execution until the current operation completes.

• Asynchronous: Code is executed non-blockingly, allowing other operations to continue while waiting for the current one to complete.

440.How do you handle errors in JavaScript?

Errors in JavaScript can be handled using try-catch blocks to catch exceptions and handle them gracefully, preventing the program from crashing.

441.What are JavaScript modules and how do they improve code organization?

JavaScript modules are reusable pieces of code that encapsulate related functionality and are designed to promote modularity, encapsulation, and code reuse, leading to better code organization and maintainability.

442.What are generators in JavaScript?

Generators are functions that can be paused and resumed, allowing for more flexible control flow.

443.Explain the concept of currying in JavaScript?

Currying is the technique for converting a function that takes multiple arguments into a sequence of functions that each take a single argument.

444.What is memoization and how is it useful in JavaScript?

Memoization is an optimization technique used to speed up function execution by caching the results of expensive function calls and returning the cached result when the same inputs occur again.

445.What is the `new` keyword in JavaScript?

The `new` keyword creates an instance of a user-defined object type or one of the built-in object types with a constructor function.

- Create a new empty object.
- Sets the prototype of the new object to the constructor`s prototype.
- Binds `this` to the new object.
- Returns the new object unless the constructor returns an object.

446.What is “Promise.all”?

“Promise.all” takes an iterable of promises and returns a single promise that resolves when all of the promises in the iterable have resolved or rejects with the reason of the first promise that rejects.

447.What is “Promise.race”?

“Promise.race” takes an iterable of promises and returns a promise that resolves or rejects as soon as one of the promises in the iterable resolves or rejects.

448.What is the purpose of the `Array.prototype.map()` method?

`Array.prototype.map()` creates a new array populated with the results of calling a provided function on every element in the calling array.

449.What is the purpose of the `Array.prototype.filter()` method?

`Array.prototype.filter()` creates a new array with all elements that pass the text implemented by the provided function.

450.What is the `Array.prototype.reduce()` method used for?

`Array.prototype.reduce()` executes a reducer function on each element of the array, resulting in a single output value.

451.What is JSON and its common operations?

JSON is a text-based data format following JavaScript object syntax, which was popularized by Douglas Crockford. It is useful when you want to transmit data across a network. It is basically a text file with an extension of .json, and a MIME type of application/json.

452.What is the purpose of the array slice method?

The slice() method returns the selected elements in an array as a new array object. It selects the elements starting at the given start argument, and ends at the given optional end argument without including the last element. If you omit the second argument then it selects till the end of the array. This method can also accept a negative index which counts back from the end of an array.

453.What is the purpose of the array splice method?

The splice() method adds/removes items to/from an array, and then returns the removed item. The first argument specifies the array position/index for insertion or deletion whereas the optional second argument indicates the number of elements to be deleted. Each additional argument is added to the array.

454.What is a pure function?

A Pure function is a function where the return value is only determined by its arguments without any side effects.

455.What is a service worker?

A service worker is basically a script that runs in the background, separate from a web page and provides features that don't need a web page or user interaction. Some of the major features of service workers are Rich offline experiences, periodic background syncs, push

notifications, intercept and handle network requests and programmatically managing a cache of responses.

456.How do you manipulate DOM using a service worker?

Service worker cannot access the DOM directly, But it can communicate with the pages it controls by responding to messages sent via the postMessage interface, and those pages can manipulate the DOM.

457.What is web storage?

Web storage is an API that provides a mechanism by which browsers can store key/value pairs locally within the user's browser, in a much more intuitive fashion than using cookies. The web storage provides two mechanisms for storing data on the client.

• Local storage: It stores data for current origin with no expiration date.

• Session storage: It stores data for one session and data is lost when the browser tab is closed.

458.What is a Cookie?

A Cookie is a piece of data that is stored on your computer to be accessed by your browser. Cookies are saved as key/value pairs.

459.What is the main difference between localStorage and sessionStorage?

LocalStorage is the same as SessionStorage but it persists the data even when the browser is closed and reopened whereas in sessionStorage data gets cleared when the page session ends.

460.What are the three states of promise?

Promises have three states:

• Pending: This is an initial state of the Promise before an operation begins.

• Fulfilled: This state indicates that the specified operation was completed.

•Rejected: This state indicates that the operation did not complete. In this case an error value will be thrown.

461.Why do you need strict mode?

Strict mode is useful to write “secure” JavaScript by notifying “bad syntax” into real errors. For example, it eliminates accidentally creating a global variable by throwing an error and also throws an error for assignment to a non-writable property, a getter-only property, a non-existing property, a non-existing variable, or a non existing object.

462.How do you declare strict mode

The strict mode is declared by adding “use strict”; to the beginning of a script or a function. If declared at the beginning of a script, it has global scope.

463.What is the purpose of the delete operator?

The delete operator is used to delete the property as well as its value.

Example:

var user= { firstName: “john”, lastName: ”Doe”, age: 20};

delete user.age;

console.log(user); // {firstName: “john”, lastName: ”Doe”}

464.is JavaScript a compiled or interpreted language?

JavaScript is an interpreted language, not a compiled language. An interpreter in the browser reads over the JavaScript code, interprets each line, and runs it. Nowadays modern browsers use a technology known as Just-In-Time (JIT) compilation, which compiles JavaScript to executable bytecode just as it is about to run.

465.What is the use of setTimeout?

The setTimeout() method is used to call a function or evaluate an expression after a specified number of milliseconds.

467.What is the use of setInterval?

The setInterval() method is used to call a function or evaluate an expression at specified intervals.

468.Why is JavaScript treated as Single threaded?

JavaScript is a single-threaded language. Because the language specification does not allow programmers to write code so that the interpreter can run parts of it in parallel in multiple threads or processes. Whereas languages like java, go, C++ can make multi-threaded and multi-process programs.

469.What are break and continue statements?

The break statement is used to “jump out” of a loop. i.e, It breaks the loop and continues executing the code after the loop.

The continue statement is used to “jump over” one iteration in the loop. i.e, It breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

470.What is a spread operator?

Spread operator allows iterables to be expanded into single arguments/elements.

Example:

```
function calculateSum(x,y,z) {

return x+y+z;

}

const numbers= [1,2,3];

console.log(calculateSum(...numbers)); //6
```

471.What is react?

React is a JavaScript library for rendering user interfaces (UI). UI is built from small units like buttons, text, and images. React lets you combine them into reusable, nestable components.

472.What are components in React?

Components are the building blocks of a React application. They are reusable, independent pieces of code that represent parts of the user interface.

473.What is JSX?

JSX stands for JavaScript XML. It is a syntax extension for JavaScript that allows you to write HTML-like code within JavaScript.

474.What is the difference between a class component and a functional component?

Class components are ES6 classes that extend `React.Component` and have a render method. Functional components are simple functions that return JSX.

475.What is state in React?

State is an object that holds the dynamic data of a component. It determines how that component renders and behaves.

476.What are props in React?

Props are read-only attributes that are passed from a parent component to a child component.

477.What is the virtual DOM?

The virtual DOM is an in-memory representation of the real DOM elements generated by React components. React uses it to optimize updates and rendering.

478.How does the virtual DOM work?

React creates a virtual DOM, updates it when the state changes and then compares it with the real DOM to apply only the necessary changes.

479.What is the purpose of `key` in React?

Keys are used to identify elements in a list. They help React identify which items have changed, are added, or are removed, enhancing performance.

480.What is lifting state up in React?

Lifting state up is the process of moving state from a child component to a common parent component to share state between multiple child components.

481.What are hooks in React?

Hooks are functions that let you use state and other React features in functional components. Examples include `useState` and `useEffect`.

482.What is `useState` hook?

`useState` is a hook that lets you add state to functional components.

483.What is `useEffect` hook?

`useEffect` is a hook that lets you perform side effects in functional components, such as fetching data, directly interacting with the DOM, and more.

484.What is context in React?

Context provides a way to pass data through the component tree without having to pass props down manually at every level.

485.What is the difference between `componentDidMount` and `useEffect`?

`componentDidMount` is a lifecycle method used in class components that runs once after the component mounts. `useEffect` can be used in functional component to achieve the same behavior.

486.What is a higher-order component (HOC)?

A higher-order component is a function that takes a component and returns a new component with additional props.

487.What are React fragments?

Fragments let you group a list of children without adding extra nodes to the DOM.

488.How do you handle forms in React?

Forms can be handled using controlled components where form data is handled by the component state.

489.What is the difference between controlled and uncontrolled components?

Controlled components have their form data controlled by the state, whereas uncontrolled components maintain their own state using refs.

490.What is React Router?

React router is a standard library for routing in React. It enables navigation among views of various components in a React application.

491.How do you pass data between components in React?

Data can be passed between components using props for parent-to-child communication, and state or context for more complex data sharing.

492.What is Lazy loading in React?

Lazy loading is a technique to load components only when they are needed, improving the performance of the application

493.What is the difference between `React.createElement` and JSX?

`React.createElement` is a function that returns an object representing a React element. JSX is a syntax sugar for `React.createElement`.

494.How do you optimize performance in a React application?

Performance can be optimized using techniques like code-splitting, memoization, and using PureComponent or React.memo to prevent unnecessary re-renders.

495.What is Redux?

Redux is a state management library for JavaScript applications, often used with React for predictable state management.

496.What are the three principles of Redux?

The three principles are: a single source of truth (the state is stored in a single object), state is read-only (immutable), and changes are made with pure functions called reducers.

497.What is `React.memo`?

`React.memo` is a higher-order component that prevents a functional component from re-rendering if its props have not changed.

498.What is `PropTypes` in React?

`PropTypes` is a type-checking library used to validate the props passed to a component, ensuring they are of the expected type.

499. What is the use of `shouldComponentUpdate`?

`shouldComponentupdate` is a lifecycle method that determines whether the component should re-render or not, based on changes in props or state.

500.What are portals in React?

Portals provide a way to render children into a DOM node that exists outside the hierarchy of the parent component.