1. **Why use Java?**
    - ***Object-Oriented Programming Language*** - It is based on the concept of class and objects.
    - ***Distributed*** - It is network aware and hence, ***java programs can easily be run on any platform***.
    - ***Secure*** - Java provides ***one with the most secure environment***. Even though it is running on a network, one can be sure about the ***online security with java runtime environment***.
    - ***Dynamic*** - Java is ***one of the most dynamic programming language***.

2. **What is the most important feature of Java?** Java is a platform independent language.

3. **What is a JVM?** JVM is Java Virtual Machine which is a runtime environment for the compiled java class files.

4. **Are JVM's platform independent?** JVM's are not platform independent. JVM's are platform specific run time implementation.

5. **What is the difference between a JDK and a JVM?** JDK is Java Development Kit which is for development purpose and it includes execution environment also. But JVM is purely a run time environment and hence you will not be able to compile your source files using a JVM.

8. **Aggregation** vs **Composition**. **Aggregation** is a relationship between two classes that is best described as a "has-a" and "whole/part" relationship. The aggregate class contains a reference to another class and is said to have ownership of that class. A child can be **present** independently while in composition it **cannot**. **Composition** is the design technique to implement has-a relationship in classes. We can use java inheritance or Object composition in java for code reuse. **Aggregation is weak** Association while **composition** being **a strong one.**

What is Aggregation? Aggregation is a specialized form of Association where all objects have their own lifecycle but there is ownership and child object can not belongs to another parent object. Let's take an example of Department and teacher. A single teacher cannot belong to multiple departments, but if we delete the department teacher object will not destroy. We can think about "has-a" relationship.

What is Composition? Composition is specialized form of Aggregation and we can call this as a "death" relationship. It is a strong type of Aggregation. Child object does not have their lifecycle and if parent object deletes all child object will also be deleted. Let's take again an example of a relationship between House and rooms. House can contain multiple rooms there is no independent life of room and any room can not belongs to two different house if we delete the house room will automatically delete.

What is association? Association is a relationship where all object have their own lifecycle and there is no owner.

9. **How to define a constant variable in Java?** Should be declared as static and final. Example: static final int MAX_LENGTH = 50

11. **Why is the main() method declared static?** This is necessary because main() is called by the JVM before any objects are made.

13. **Can a main() method be overloaded?** Yes. You can have any number of main() methods with different method signature and implementation in the class.

14. **Can a main() method be declared final?** Yes. Any inheriting class will not be able to have it's own default main() method.

15. **Does the order of public and static declaration matter in main() method?** No. It doesn't matter but void always come before main().

16. **Can a source file contain more than one class declaration?** Yes a single source file can contain any number of Class declarations but only one of the class can be declared as public.

17. **What is a package?** Package is a collection of related classes and interfaces.

18. **Can a class be declared as protected?** The protected access modifier cannot be applied to class and interfaces. Methods, fields can be declared protected, however methods and fields in a interface cannot be declared protected.

19. **I don't want my class to be inherited by any other class. What should i do?** You should declared your class as final. But you can't define your class as final, if it is an abstract class. A class declared as final can't be extended by any other class.

What purpose does the keywords final, finally, and finalize fulfill?
Final is used to apply restrictions on class, method and variable. Final class can't be inherited, final method can't be overridden and final variable value can't be changed.
Finally is used to place important code, it will be executed whether exception is handled or not.
Finalize is used to perform clean up processing just before object is garbage collected.

21. **static** keyword in java which identifies it is class based i.e. it can be accessed without creating the instance of a Class.

34. Can a class be declared as static? We can not declare top level class as static, but only inner class can be declared static.

```
public class Test {
static class InnerClass {
public static void InnerMethod() { System.out.println("Static Inner Class!"); }
}
        public static void main(String args[]) { Test.InnerClass.InnerMethod(); }
}
//output: Static Inner Class!
```

35. When will you define a method as static? When a method needs to be accessed even before the creation of the object of the class then we should declare the method as static.

37. I want to print "Hello" even before main() is executed. How will you achieve that? Print the statement inside a static block of code. Static blocks get executed when the class gets loaded into the memory and even before the creation of an object. Hence it will be executed before the main() method. And it will be executed only once.

38. What is the importance of static variable? static variables are class level variables where all objects of the class refer to the same variable. If one object changes the value then the change gets reflected in all the objects.

39. Can we declare a static variable inside a method? Static variables are class level variables and they ***can't be declared inside a method***. If declared, the class will not compile.

40. What value does read() return when it has reached the end of a file? The read() returns -1 when it has reached the end of a file.

41. Can a Byte object be cast to a double value? No, an object cannot be cast to a primitive value.

42. Which class is extended by all other classes? The Object class is extended by all other classes.

43. What restrictions are placed on method overloading? Two methods may not have the same name and argument list but ***different return types.***

44. What is *casting*? There are two types of casting, casting ***between primitive numeric types*** is used to convert larger values, such as double values, to smaller values, such as byte values. Casting ***between object references*** is used to refer to an object by a compatible class, interface, or array type reference.

45. What is the return type of a program's main() method? void.

46. What is *Downcasting*? Downcasting is the casting from a general to a more specific type, i.e. casting down the hierarchy

47. What is a native method? A native method is a method that is implemented in a language other than Java.

49. Does a class inherit the constructors of its superclass? A class ***does not*** inherit constructors from any of its superclasses.

50. Name the eight primitive Java types. The eight primitive types are byte, char, short, int, long, float, double, and boolean.

51. What happens when you add a double value to a String? The result is a String object.

52. What is the difference between inner class and nested class? When a class is defined within a scope of another class, then it becomes inner class. If the access modifier of the ***inner class is static***, then it ***becomes nested class***.

57. What happens to a static variable that is defined within a method of a class ? Can't do it. You'll get a compilation error.

60. Can a for statement loop indefinitely? Yes, a for statement can loop indefinitely. For example, consider the following: for(;;);

61. To what value is a variable of the String type automatically initialized? The default value of an String type is ***null***.

62. How are this() and super() used with constructors? this() is used to invoke a constructor of the same class. super() is used to invoke a superclass constructor.

63. What is a transient variable? Transient variable is a ***variable that may not be serialized***.

64. What is the difference between the >> and >>> operators? The **>>** operator carries the sign bit when shifting right. The **>>>** zero-fills bits that have been shifted out.

66. What is the purpose of garbage collection in Java, and when is it used? The purpose of garbage collection is to identify and discard objects that are no longer needed by a program so that their resources can be reclaimed and reused. A Java object is subject to garbage collection when it becomes unreachable to the program in which it is used

What does Thread mean? A thread is independent execution path.

What is the difference between Thread and Process in Java? (answer)

The **thread** is a ***subset of Process***, in other words, one process can contain multiple threads. Two process runs on different memory space, but all threads share same memory space. Don't confuse this with stack memory, which is different for the different thread and used to store local data to that thread. For more detail see the answer.

Explain different way of using thread? Extending *Thread class* or **implementing *Runnable Interface***.

Threads good for*: to maintain responsiveness of an application during a long running task &* to enable cancellation of separable tasks.

JVM creates a Thread object whose task is **defined by the main() method**. It starts the thread.

Multithreading refers to two or more tasks executing concurrently within a single program. A **thread is** an independent path of execution within a program. Many **threads can** run concurrently within a program. Every **thread** in **Java is** created and controlled by the **java**.lang.**Thread** class.

- Each thread has its private run-time stack
- if two thread execute the same method, each will have its own copy of the local variables the methods uses
- however, all threads see the same dynamic memory (heap)
- two different threads can act on the same object and same static fields at the same time (concurrently/simultaneously)

What is *volatile variable* in Java? Special modifier, which can only be used with instance variables.

What is thread-safety? is Vector a thread-safe class? (**Yes**, see details)

Thread-safety is a property of an object or code which guarantees that if executed or used by multiple threads in any manner e.g. read vs write it will behave as expected. For example, a thread-safe counter object will not miss any count if same instance of that counter is shared among multiple threads.

What are the differences between Heap(objects) and Stack(threads) Memory?

Features	***Stack***	***Heap***
Memory	Stack memory is used only by one thread of execution.	Heap memory is used by all the parts of the application.
Access	Stack memory can't be accessed by other threads.	Objects stored in the heap are globally accessible.
Memory Management	Follows LIFO manner to free memory.	Memory management is based on generation associated to each object.
Lifetime	Exists until the end of execution of the thread.	Heap memory lives from the start till the end of application execution.
Usage	Stack memory only contains local primitive and reference variables to objects in heap space.	Whenever an object is created, it's always stored in the Heap space.

70. *Primitive data types* are passed by reference or *pass by value*? Primitive data types are passed by ***value***.

71. *Objects are passed by* value or by *reference*? Java only supports pass by value. With objects, the object reference itself is passed by value and so both the original reference and parameter copy both refer to the same object.

Explain **Autoboxing** - is the automatic conversion that the Java compiler makes between the primitive types and their corresponding object wrapper classes.

Differences between ***Checked Exception*** and ***Unchecked Exception***?

Checked Exception The classes that extend Throwable class except RuntimeException and Error are known as checked exceptions. Checked exceptions are checked at compile-time. Example: IOException, SQLException etc.

Unchecked Exception The classes that extend RuntimeException are known as unchecked exceptions. Unchecked exceptions are not checked at compile-time. Example: ArithmeticException, NullPointerException etc.

Difference between ***Error*** and ***Exception***? An error is an irrecoverable condition occurring at runtime. Such as OutOfMemory error. These JVM errors you can not repair them at runtime.Though error can be caught in catch block but the execution of application will come to a halt and is not recoverable.

How can you handle Java exceptions? *Try, catch, finally; throw, throws*

78. If I write System.exit(0); at the end of the try block, will the finally block still execute? No. In this case the finally block will not execute because when you say System.exit(0); the control immediately goes out of the program, and thus finally never executes.

81. What is the difference between a *while* statement and a *do* statement?

a. A while statement checks at the beginning of a loop to see whether the next loop iteration should occur.
b. A do statement checks at the end of a loop to see whether the next iteration of a loop should occur. The do statement will always execute the body of a loop at least once.

82. What is the difference between a *while* statement and a *do while* statement? A while statement checks at the beginning of a loop to see whether the next loop iteration should occur. A do while statement checks at the end of a loop to see whether the next iteration of a loop should occur. The do while statement will always execute the body of a loop at least once.

85 Can main() method be declared final? Yes, the main() method ***can be*** declared final, ***in addition to being*** public static.

86. What is HashMap and Map? Map is an Interface and Hashmap is the class that implements Map.

87. Difference between HashMap and HashTable? The HashMap class is roughly equivalent to Hashtable, except that it is unsynchronized and permits nulls. (HashMap allows null values as key and value whereas Hashtable doesn't allow). HashMap does not guarantee that the order of the map will remain constant over time. HashMap is unsynchronized and Hashtable is synchronized.

88. Difference between Vector and ArrayList? Vector is synchronized whereas arraylist is not.

89. What is the difference between ArrayList and LinkedList?

1. ArrayList needs to be moved to a bigger space if new elements added to a filled array which is not required for LinkedList.
2. Removal and Insertion at specific place in ArrayList requires moving all elements and hence leads to O(n) insertions and removal whereas its constant O(1) for LinkedList.
3. Random access via index in ArrayList is faster than LinkedList which requires traversing the complete list through references.
4. For a set of integers you want to sort using quicksort, it's probably faster to use an array; for a set of large structures you want to sort using selection sort, a linked list will be faster.

List vs Set vs Map Interfecases of Collection

1) **Duplicity:**

List allows duplicate elements.
Set doesn't allow duplicates. Faster to add elements
Map stored the elements as key & value pair. Map doesn't allow duplicate keys while it allows duplicate values.

2) **Null values:**

List allows any number of null values.
Set allows single null value at most.
Map can have single null key at most and any number of null values.

3) **Order:**

List and all of its implementation classes maintains the insertion order.
Set doesn't maintain any order; still few of its classes sort the elements in an order such as LinkedHashSet maintains the elements in insertion order.
Map also doesn't stores the elements in an order, however few of its classes does the same. For e.g. TreeMap sorts the map in the ascending order of keys and LinkedHashMap sorts the elements in the insertion order, the order in which the elements got added to the LinkedHashMap.

OOPs or **Object Oriented Programming** is a Programming model which is organized around Objects instead of processes. We are using OOP because of the **following features**:

1. Abstraction - process of hiding implementation details and show only functionality to user.

2. Encapsulation is used for access restriction to a class members and methods. **Public, private, default, protected**.
- The variables of a class should only be accessed by that class
- Other classes should access these variables through methods such as getters and setters, in order to protect data integrity

3. Polymorphism is the concept where an object behaves differently in different situations. One function can have different forms. By using ***overloading*** **and** ***overriding*****.**
Overloading is **compile time** Polymorphism and **Overriding** are **Runtime(*Dynamic binding*)** Polymorphism.

4. **Inheritance** - where an object is based on another object. Inheritance is the mechanism of code reuse. The object that is getting inherited is called superclass and the object that inherits the superclass is called subclass. We use ***extends*** keyword in java to implement inheritance.

What is overloading in OOPs? *Overloading* is a process used to avoid redundant code where the ***same method name used multiple times but with a different set of parameters***. The actual method that gets called during runtime is resolved at compile time, thus avoiding runtime errors. ***Static, final, private*** methods you can use.

What is overriding? The logic of ***parent methods can be overridden by a child class***. Overriding is a process that allows a child class or subclass to provide a specific implementation of functions that is already provided by one of its super-classes or parent classes.

Can we overload a static method in Java? **Yes**, you can overload a static method in Java.

The difference between method overloading and overriding? Overloading is resolved at compile time. Overriding is resolved at runtime.

Can we prevent overriding a method without using the final modifier? Yes, you can prevent the method overriding in Java without using the final modifier. In fact, there are several ways to accomplish it e.g. you can mark the method private or static, those cannot be overridden.

Can we override a private method in Java? No, you cannot. Since the private method is only accessible and visible inside the class they are declared, it's not possible to override them in subclasses.

Can we change the return type of method to subclass while overriding? Yes, you can

Can we change the argument list of an overriding method? No, you cannot. The argument list is part of the method signature and both overriding and overridden method must have the same signature.

```
public class Superclass { public void printMethod() { System.out.println("Printed in Superclass."); } }
        // Here is a subclass, called Subclass, that overrides printMethod():
public class Subclass extends Superclass {
                // overrides printMethod in Superclass
        public void printMethod() {
                super.printMethod();
                System.out.println("Printed in Subclass");
        }

        public static void main(String[] args) {
                Subclass s = new Subclass();
                s.printMethod();  }
}
```

What is the difference between object and the class?

N	Object	Class
1)	Object is an **instance** of a class.	***Class*** is a **template** from which objects are created.
3)	Object is created through **new keyword** mainly e.g. Student s1 = new Student();	Class is declared using **class keyword** e.g. class Student{}
4)	Object is created **many times** as per requirement.	Class is declared **once**.
5)	Object **allocates memory when it is created**.	Class **doesn't allocated memory when it is created**.

Difference between class and interface java? An **interface** has fully abstract methods i.e. methods with nobody. An **interface** is syntactically similar to the **class** but there is a major **difference** between **class** and **interface** that is a **class** can be instantiated, but an **interface** can never be instantiated. ... **Class**: A **class** is instantiated to create objects.

Can a class extend more than one classes or does java support multiple inheritance? If not, why? **No**, a class in java **can not extend more than one classes** or **java does not support multiple inheritance**. A Class can implement any number of Interfaces.

Does java allow implementation of multiple interfaces having Default methods with Same name and Signature - No. Compilation error.

Can we modify variables defined inside Interface? Since, variables defined inside interface are ***final*** therefore we **cannot change the value of these** variables **anywhere** (simple OOPS concept). If we try to change the value, then compiler throws error.

The **Try-Catch** actually contains 3 pieces:

- ➢ The **try** block
  - ▸ Contains the code you'd like try and run
- ➢ The **catch** block
  - ▸ Contains the code you'd like to execute if an error occurs
- ➢ The **finally** block
  - ▸ Contains the code you'd like to execute regardless

Will code in a Finally statement fire if I return a value in a Try block? - YES

If a **return** statement is **present in Finally** block, then the return value in the finally block will **override** the **return** value **in** the **try or catch** blocks. If no return statement in Finally block, then the statements in finally block are executed and then either try or catch block return statements (where the control was last before entering the finally block) is executed.

What is a servlet? Java Servlet is server side technologies to extend the capability of web servers by providing support for dynamic response and data persistence.

What do you do in dab to prevent duplication - ***normalization*** *i*s process used to organize a database into tables and columns. The idea is that a table should be about a *specific* topic and that only those columns which support that topic are included.

The Normal Forms - certain rules in database management system design have been developed to better organize tables and minimize anomalies. The stage at which a table is organized is known as its normal form (or a stage of normalization).

There are 3 stages of normal forms:

- **first** normal form (or **1NF**)
  - *Rules*:
    1. Columns must have single values
    2. Columns must have unique names
    3. Values of a given attribute must be of the same data type
    4. No two records (or rows) can be identical
- **second** normal form (or **2NF**)
  - *Rules*: ***1.*** Be in 1NF; ***2.*** Single Column Primary Key
- **third** normal form (or **3NF**)
  - *Rules*: ***1.*** Be in 2NF; ***2.*** Has no transitive functional dependencies

Why is **String immutable** in Java?

String Pool - When a string is created and if it exists in the pool, the reference of the existing string will be returned instead of creating a new object. If string is not immutable, changing the string with one reference will lead to the wrong value for the other references.

Example:

```
String str1 = "String1";
String str2 = "String1"; // It doesn't create a new String and rather reuses the string literal from pool
// Now both str1 and str2 pointing to same string object in pool, changing str1 will change it for str2 too
```

Security - String is widely used as parameter for many java classes, e.g. network connection, opening files, etc. *Making it mutable might possess threats due to interception by the other code segment.*

What is an Immutable Object - Object that can't be changed after instantiation.
What is an immutable class - Class using which only immutable (objects that cannot be changed after initialization) objects can be created.

String, StringBuilder and StringBuffer?

- If the Object value will ***not change***, use **String** Class because a String **object is immutable**.
- If the Object value can change and **will only be modified from a single thread**, use **StringBuilder** because StringBuilder is **unsynchronized**(**means faster**).
- If the Object value may change, and can be modified by **multiple threads**, use a **StringBuffer** because StringBuffer is **thread safe(synchronized)**.

Recursion - method calls itself to solve some problem.
One of the classic problems for introducing recursion is calculating the factorial of an integer.

```
private static long factorial(int n) {
        if (n == 1) return 1;
        else return n * factorial(n-1);
}
```

What is the constructor? A constructor is a **method which is used to initialize a newly created object** and is called just after when memory allocated to the object. It can be used to implement the objects to desired values or default values at the time of object creation.

What are the Rules in defining a constructor?

- Constructor name should be same as class name; It should not contain return type.
- It should not contain **Non Access Modifiers**: ***final*** ,***static***, ***abstract***, ***synchronized***
- In it logic return statement with value is not allowed.
- It **can have** all four accessibility modifiers: ***private*** , ***public***, ***protected***, ***default***
- It can have throws clause: we can throw exception from constructor.
- We can not place return in constructor.

***What is the difference between a constructor and a method*?**

a. Constructors create and initialize objects that don't exist yet, while methods perform operations on objects that already exist.
b. Constructors can't be called directly; they are called implicitly when the new keyword creates an object. Methods can be called directly on an object that has already been created with new keyword.
c. Constructors must be named with the same name as the class name. They can't return anything, even void (the object itself is the implicit return). Methods must be declared to return something, although it can be void.

Copy Constructor in java class is a special type of constructor that takes same class as argument. ***Copy constructor*** is used to provide a copy of the specified object.

It(**this**) works as a reference to a current object whose method or constructor is being invoked. this keyword can be used to refer any member of current object from within an instance method or a constructor.

How are this() and super() used with constructors? this() is used to invoke a constructor of the same class. super() is used to invoke a superclass constructor. You ***can use one at the time***.

Java access modifiers

public: visible to **all** other **classes**
default: If a class has no modifier, it is visible only within its own package (packages are groups of related classes)
private: visible only to the **current class**, its methods, and every instance (object) of its class
protected: visible to the **current class**, and **all** of **its child classes**

An ***abstract method*** is a method whose implementation is deferred (hold over) to a subclass.

Abstract class - a Class which ***doesn't provide complete implementation*** is defined as an abstract class. Abstract classes enforce abstraction.

An **interface** can contain only method declarations. An **Interface** is 100% abstract **class**. An ***interface & abstract class can't be instantiated*** (cannot make its objects). **Final** var in interface by default. All ***methods*** in interface are public and abstract.

What is the difference between abstract classes and interfaces?

Abstract Class	Interfaces
An abstract class can provide complete, default code and/or just the details that have to be overridden.	An interface cannot provide any code at all, **just the signature**.
In case of abstract class, a class may **extend only one** abstract class.	A Class may implement **several interfaces.**
An abstract class can have non-abstract methods.	All methods of an Interface are abstract.
An abstract class can have instance variables.	An Interface cannot have instance variables
An abstract class can have any visibility: **public, private, protected.**	An Interface visibility must be **public** (or) none.
An abstract class can contain constructors	An Interface cannot contain constructors
Abstract classes are **fast**	Interfaces are **slow** as it requires extra indirection to find corresponding method in the actual class

Can we make a class abstract without an abstract method? Yes, just add abstract keyword on the class definition and your class will become abstract.
- Can a abstract class be declared final? ***Not*** possible. An abstract class without being inherited is of no use and hence will result in compile time error.
- Can an Interface be final? ***Not*** possible. Doing so so will result in compilation error.
- Can you create an object of an abstract class? ***Not*** possible. Abstract classes can't be instantiated.
- Can a abstract class be defined without any abstract methods? Yes it's possible.
- Can an Interface *implement* another Interface? ***Interface doesn't provide implementation***
- Can an Interface *extend* another Interface? ***Yes***. Interface ***can extend more than one Interface***.
- Can a class be defined inside an Interface? Yes it's possible.
- Can an Interface be defined inside a class? Yes it's possible.
- What is a *Marker* Interface? An ***Interface which doesn't have any declaration inside*** but still enforces a mechanism.

Abstract class Vs Interface

- ***Use*** an **Abstract** Class ***when***:
  - The logic for some methods will always be the same in child classes
  - You can create logic that will be useful for all child classes
  - You want to create variables that all child classes will use and set values for
  - Contains the methods you would always use anyway (dab connection, closing conn…)
- ***Use*** an **Interface** Class ***when***:
  - **Its blueprint of the class(security)**
  - provide a communication contract between two objects.
  - You just want to create a structure
  - The only variables you need in all child classes will have constant values

What is the difference between ***comparable*** and ***comparator***? **Comparable interface** is ***used for single sequence sorting*** i.e. sorting the objects based on single data member where as ***comparator interface is used to sort the object based on multiple data members.***

What is *synchronization* and why is it important? Synchronization refers to multi-threading. A synchronized block of code can be executed by only one thread at a time. Synchronization is a process which keeps all concurrent threads in execution to be in sync. Synchronization avoids memory consistency errors caused due to inconsistent view of shared memory.

Synchronized Methods are methods that are used to **control access to an object**. **Synchronized** blocks in **Java** are marked with the **synchronized** keyword. A **synchronized** block in **Java** is **synchronized** on some object. All **synchronized** blocks **synchronized** on the same object can only have **one thread executing inside them at a time**. ... This **synchronization** is implemented in **Java** with a concept called monitors.

What are synchronized methods and synchronized statements?

a. ***Synchronized methods*** are methods that are used to control access to an object. A thread only executes a synchronized method after it has acquired the lock for the method's object or class.

b. ***Synchronized statements*** are similar to synchronized methods. A synchronized statement can only be executed after a thread has acquired the lock for the object or class referenced in the synchronized statement.

What is the common usage of serialization? Whenever an object is to be sent over the network, objects need to be serialized. Moreover if the state of an object is to be saved, objects need to be serialized.

What is **Serialization**(**Storing**)? Serialization is a mechanism by which you can save the state of an object by converting it to a byte stream.

==========

Difference between WHERE and HAVING SQL

HAVING: is used to check conditions ***after*** the aggregation (collection) takes place.

WHERE: is used to check conditions ***before*** the aggregation(collection) takes place.

The main difference between WHERE and HAVING clause is, **WHERE** is used **for row operations** and **HAVING** is used **for column operations**.

A **primary key** is a field in a **table** which uniquely identifies each row/record in a database **table**. **Primary keys** must contain unique values. A **primary key** column cannot **have** NULL values. A **table can have only one primary key**, which **may** consist of **single** or **multiple** fields.

Foreign key is defined in a second table, but it refers to the primary **key** or a unique **key** in the first table.

A relational database management system (**RDBMS**).
A **database** is a collection of information that is organized so that it can be easily accessed, managed and updated.

Database normalization is used to organize a database.

Ans. **Inner join** is the intersection of two tables on the condition defined by the where clause i.e will get **records from both tables matched by a column**.
Outer join is the union of two tables i.e. will **get all records from both tables** and **will put null in the columns where related records are not present.**
Left Outer join is the left union of two tables i.e. all records from the table on the left and values from the right table for related records else null for the columns from right table.
Right Outer join is the right union of two tables i.e. all records from the table on the right and values from the left table for related records else null for the columns from left table.
What is **Self Join** and What is it's purpose - When a **Table Join itself**, it's a Self Join. For *example* - we like to know the pair of department names where first dept has lesser employees than the later.

Select D1.name , D2.name from Dept D1, Dept D2 where D1.employee_count < D2.employee_count

An **index** is used to speed up searching in the database. An index can be used to efficiently find all rows matching some column in your query and then walk through only that subset of the table to find exact matches. If you don't have indexes on any column in the *WHERE* clause, the SQL server have to walk through *the whole table* and check every row to see if it matches, which may be a slow operation on big tables.

The index can also be a UNIQUE index, which means that you cannot have duplicate values in that column, or a PRIMARY KEY which in some storage engines defines where in the database file the value is stored.

What is a **Queue data structure**? A Queue is a data structure which is linear and follows a fixed order in which the operations occur. The order is always **First In First Out** (FIFO). An excellent example of a queue is any queue of the customer for a resource where the customer that came first served first. The difference between stacks and queues is while deleting.

What is a **Stack data structure**? Basic features of Stack are an ordered list of the similar data type. The stack is LIFO (Last In First Out) data structure, or we can say FILO (**First In Last Out**) data structure. Push () method is used to insert/enter new elements into the Stack and pop() method is used to remove/delete an element from the stack.

What are new features introduced with Java 8 ?

- *Lambda Expressions*
- Interface Default and Static Methods
- Method Reference
- Parameters Name
- Optional
- *Streams*
- Concurrency

Why Spring? **Spring** is also another java based framework which makes developing a J2EE application very easy. It provides many capabilities like (a container which creates the objects you depend on rather than you create them of your own).

Why Spring MVC? A lot of implementations
1) Spring provides easy and **secure** way to handle Forms, Logins in itself (access control - **authentication** & **authorization**)
2) Spring takes care of all the **dependencies** you might want or need to build
3) Manage lifecycle of Java classes - called **beans https://www.tutorialspoint.com/spring/spring_bean_life_cycle.htm**
4) Spring handles **Autowiring** exceptionally well which can be a nightmare while you have to build complex web apps using Java.
Autowiring enables the programmer to inject the bean automatically. We don't need to write explicit injection logic. Let's see the code to inject bean using dependency injection.
5) Spring comes with various **design patterns** in its core principles such as **Factory** to create objects of beans using Application context reference. Spring uses **Proxy** pattern for AOP. **MVC.** By default, all the beans defined in the Spring are only created once and successive calls to getBean() returns the same object, hence effectively making use of **Singleton** design pattern.

Why Spring Boot? A suite of pre-configured frameworks and technologies (less code for small tasks)
- The shortest way to have a spring application up and running
- Spring boot improvement handle configuration and dependencies.
- Don't need to manually create beans – you can tell the spring boot this is the service and that all.
- Less configuration and coding.
- Instead of building the JSP pages All view will be in React or Angular.

What is Dependency Injection? How does dependency injection work? Dependency injection is one of the design pattern that help us to create complex systems in a simpler manner.

Dependency is another class that you need for your class to function.
Dependency Injection - when Dependency is push into the class from the outside. You shouldn't instantiate dependencies using a NEW operator from inside of the class. Instead you can take it as a constructor parameter or via setter.
Dependency Inversion Principle - code should depend upon abstractions we're decoupling our implementations from each other. Making our code cleaner, easier to modify and easier to reuse.

Web services is a software service used to communicate between two devices on a network. It does so over HTTP using technologies such as XML, SOAP, WSDL, and UDDI.

API - Application Programming Interface

(REST API) Rest Service (API) REpresentative State Transfer is a simple web-based interface that uses XML and HTTP	**SOAP-**Simple Object Access Protocol The 'Complex' method, which is a more abstract framework - with the benefit of offering more capability and flexibility.
1. using HTML and XML as the primary file type that carries the content. 2. A set of well-defined operations that apply to all pieces of information (called resources) - the main ones are GET, POST, PUT, and DELETE. 3. A universal syntax - basically means that each resource has its own unique URI (or URL). 4. stateless communication - all the information necessary to understand the request is contained in the HTTP message.	o Language, Platform & Transport Independent o Distributed Enterprise Environments o Standardized o Pre-build Extensibility o Build-in Error Handling o Automation

REST API methods:

GET - used to **retrieve data** from a server at the specified resource.

POST - used to **send data to the API server** to ***create*** or ***update*** a resource

PUT - Similar to POST, PUT requests are used to send data to the API to **create or update a resource**. The difference is that **PUT requests are idempotent**. That is, calling the same PUT request multiple times **will always produce the same result**. In contrast, calling a POST request repeatedly make have side effects of creating the same resource multiple times.

DELETE - **delete the resource at the specified URL**. If a new user is created with a POST request to /users, and it can be retrieved with a GET request to /users/{{userid}}, then making a DELETE request to /users/{{userid}} will **completely remove that user.**

MVC? ***Model*** – ***View*** – ***Controller*** - is the name of a methodology or ***design pattern*** for successfully and efficiently relating the user interface to underlying data models.

Model: Structures your data in a reliable form and prepares it based on controller's instructions

View: Displays data to user in easy-to-understand format, based on the user's actions

Controller: Takes in user commands, sends commands to the model for data updates, sends instructions to view to update interface.

Any other patterns you know of? you like? Why? This is the most used pattern. **Singleton** - You need an object that only needs to be ***instantiate once***, so, you can use a singleton. The class ***needs to declare a private constructor*** to prevent people to instantiate it from outside the class. Also, you need to ***declare a static field of the type of the singleton***. The method *getInstance*() assures that only one instance of this class is created at runtime.

```
public class SingletonSample {
  private static SingletonSample instance = null;

  private SingletonSample() { } //private constructor

  public static SingletonSample getInstance() {
        if(instance == null) { instance = new SingletonSample(); }
        return instance;
  }
}
```

Java ***Servlets*** are server-side Java program modules that process and answer client requests and implement the servlet interface.
A servlet acts as an intermediary between the client and the server.
As servlet modules run in the server, they can receive and respond to requests made by the client.

What is ***Request Dispatcher***?
RequestDispatcher interface is used to forward the request to another resource that can be HTML, JSP or another servlet in same application. We can also use this to include the content of another resource to the response.

There are two methods defined in this interface:

1. ***void forward()***
2. ***void include()***

What are the differences between *forward()* method and *sendRedirect()* methods?

Forward() method	SendRedirect() method
sends the same request to another resource.	sends new request always because it uses the URL bar of the browser.
works at server side.	works at client side.
works within the server only.	works within and outside the server.

What is the life-cycle of a servlet?
There are ***5 stages*** in the lifecycle of a servlet:

1. Servlet is *loaded*
2. Servlet is *instantiated*
3. Servlet is *initialized*
4. Service the *request*
5. Servlet is *destroyed*

How does *cookies* work in Servlets?
Cookies are text data sent by server to the client and it gets saved at the client local machine.

How to ***disable session*** in JSP? <%@ **page session="false"** %>

Session simply **means** a particular interval of time. **Session** Tracking is a way to maintain data of an user. It is also known as **session** management in servlet.
Http protocol is a stateless so we need to maintain state using session tracking techniques. Each time user requests to the server, server treats the request as the new request. So we need to maintain the state of an user to recognize to particular user.

Session Tracking Techniques:

- Cookies
- Hidden Form Field
- URL Rewriting
- HttpSession

Agile

Software Development Life Cycle (***SDLC***) is a process used by the software industry to design, develop and test high quality softwares. The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates

Business Analysis -> **Requirement Gathering** -> **Architecting & Design** -> **Development** -> **Testing & valuation** -> **Deployment** -> **Maintenance**

STLC - **Software Testing Life Cycle**

Business Analysis -> **Test Planning** -> **Test Case Development** -> **Environment Setup** -> **Test Execution** -> **Test Cycle Closure**

Entry Criteria gives the prerequisite items that must be completed before testing can begin.
Exit Criteria defines the items that must be completed before testing can be concluded.

Shift to the Left in Quality Engineering & Assurance is about moving activities from the current phase to the applicable prior phase, thereby detecting defects earlier in the game. This results in reduced cost, improved Quality and faster time to market.

Knowledge Transition (KT) is the process from where the project team gains information on the application, business process and testing processes from the client/SME at the beginning of a project.
Requirement Analysis

SMAC = Social Media, Mobile, Analytics, & Cloud Computing

Velocity is a metric that predicts how much work an Agile software development team can successfully complete within a two-week sprint (or similar time-boxed period). ***# of user stories you complete***.
Velocity is a useful planning tool for estimating how fast work can be completed and how long it will take to complete a project.

Sprint Backlog - A detailed list of tasks to be completed during a Sprint on a daily basis.

What is ***Hibernate*** Framework? Object-relational mapping or ORM is the programming technique to map application domain model objects to the relational database tables. Hibernate is java based ORM tool that provides framework for mapping application domain objects to the relational database tables and vice versa.

Spring Bean Scopes

1. singleton(default*)

Scopes a single bean definition to a single object instance per Spring IoC container.

2. prototype

Scopes a single bean definition to any number of object instances.

3. request

Scopes a single bean definition to the lifecycle of a single HTTP request; that is each and every HTTP request will have its own instance of a bean created off the back of a single bean definition. Only valid in the context of a web-aware Spring ApplicationContext.

4. session

Scopes a single bean definition to the lifecycle of a HTTP Session. Only valid in the context of a web-aware Spring ApplicationContext.

5. global session

Scopes a single bean definition to the lifecycle of a global HTTP Session. Typically only valid when used in a portlet context. Only valid in the context of a web-aware Spring ApplicationContext.