==========

JAVA QUESTIONS & ANSWERS1

==========

1. **What is constructor chaining?**
   Constructor chaining is the process of calling one constructor from another constructor with respect to the current object.
   Constructor chaining can be done in two ways:
   - **Within same class: It can be done using this() keyword for constructors in same class.**
   - **From base class: by using super() keyword to call constructor from the base class.**
     REF :- https://www.geeksforgeeks.org/constructor-chaining-java-examples

2. **Explain this and super keywords.**

```
// Java program to illustrate Constructor Chaining
// within same class Using this() keyword
// and changing order of constructors
class Temp
{
        // default constructor 1
        Temp()
        {
                System.out.println("default");
        }

        // parameterized constructor 2
        Temp(int x)
        {
                // invokes default constructor
                this();
                System.out.println(x);
        }

        // parameterized constructor 3
        Temp(int x, int y)
        {
                // invokes parameterized constructor 2
                this(5);
                System.out.println(x * y);
        }
```

```
        public static void main(String args[])
        {
                // invokes parameterized constructor 3
                new Temp(8, 10);
        }
}
```

3. **What is the need of the Default method inside the interface?**
   It is given for backward compatibility. Suppose you have one interface and ten classes implementing it. Later if you want to enhance the functionality of the interface by adding some method inside that, due to this all ten implementation classes will get disturb. To overcome this problem JAVA8 introduced this feature.

4. **What is the need of the Static method inside the interface?**
   To provide the utility functionality.

5. **Where have you used Stream in your project?**
   https://javaconceptoftheday.com/solving-real-time-queries-using-java-8-features-employee-management-system/

6. **What is the contract between equals and hashCode method? What will happen if we do not follow the contract?**
   REF : https://www.youtube.com/watch?v=QixVKPBVy38
   you must override hashCode() in every class that overrides equals() method
   Failure to do so will result in a violation of the general contract for Object.hashCode(), which will prevent your class from functioning properly in conjunction with all hash-based collections, including HashMap, HashSet and HashTable
   If two objects are having the same hashcode it doesn't mean that those objects are equal. But if two objects are equal using equals method then they must have the same hashcode.

7. **Please show me how to override hashcode and equals methods.**

```
package com.jit.hashcodeequals;

//https://www.youtube.com/watch?v=QixVKPBVy38
public class Employee {
        private int empId;
        private String empName;
        private String cmpnyName;

        public Employee(int empId, String empName, String cmpnyName) {
                this.empId = empId;
```

```
		this.empName = empName;
		this.cmpnyName = cmpnyName;
	}
g
	public int getEmpId() {
		return empId;
	}

	public void setEmpId(int empId) {
		this.empId = empId;
	}

	public String getEmpName() {
		return empName;
	}

	public void setEmpName(String empName) {
		this.empName = empName;
	}

	public String getCmpnyName() {
		return cmpnyName;
	}

	public void setCmpnyName(String cmpnyName) {
		this.cmpnyName = cmpnyName;
	}

	@Override
	public String toString() {
		return "" + "empId= " + empId + ", empName=" + empName + ",
cmpnyName=" + cmpnyName + "";
	}

	@Override
	public int hashCode() {
		final int prime = 31;
		int result = 1;
		result = prime * result + ((cmpnyName == null) ? 0 :
cmpnyName.hashCode());
		result = prime * result + empId;
		result = prime * result + ((empName == null) ? 0 :
empName.hashCode());
		return result;
```

```
	}

	@Override
	public boolean equals(Object obj) {
		if (this == obj)
			return true;
		try {
			Employee e = (Employee) obj;
			if (this.empId == e.empId &&
this.empName.equals(e.empName) && this.cmpnyName.equals(e.cmpnyName)) {
				return true;
			} else {
				return false;
			}
		} catch (ClassCastException e) {
			return false;
		} catch (NullPointerException e) {
			return false;
		}
	}

}
```

8. **Explain Singleton design pattern.**

```
public class Printer {

	private static volatile Printer INSTANCE;

	// private constructor
	private Printer() {
		if (INSTANCE != null) {//to stop object creation using reflection API
			throw new RuntimeException("object is already created");
		}
		System.out.println("Printer::0-param constructor");
	}

	// static factory method
	public static Printer getInstance() {
		if (INSTANCE == null) {// first null check
			synchronized (Printer.class) {
				if (INSTANCE == null)
					INSTANCE = new Printer();
			}
```

```
                }
                return INSTANCE;
        }

        //to stop cloning
        @Override
        protected Object clone() throws CloneNotSupportedException {
                throw new CloneNotSupportedException("Cloning not allowed in
singleton Printer class");
        }

        //to stop Deserialization
        public Object readResolve() {
                return INSTANCE;
        }

        //business method
        public void print(String msg) {
                System.out.println(msg);
        }
}
```

9. What is the difference between StringBuffer and StringBuilder?
   - StringBuffer is synchronized i.e. thread safe. It means two threads can't call the methods of StringBuffer simultaneously.
   - StringBuilder is non-synchronized i.e. not thread safe. It means two threads can call the methods of StringBuilder simultaneously.

10. Explain internal working of HashMap. What enhancement was done in Java8?
    REF :- https://www.youtube.com/watch?v=c3RVW3KGIIE
    In Java8 , when we have too many unequal keys which gives same hashcode(index) -
    when the number of items in a hash bucket grows beyond certain threshold (TREEIFY_THRESHOLD =8), content of that bucket switches from using a linked list of Entry objects to a balanced tree. This theoretically improves the worst-case performance from O(n) to O(log n)

11. What is the prerequisite if we want to store data in sorting order?
    Class must be comparable.
12. How to create our own immutable class?
    1. Declare the class as final so it can't be extended.
    2. Make all fields private so that direct access is not allowed.
    3. Don't provide setter methods for variables

4. Make all fields final so that it's value can be assigned only once.
5. Initialize all the fields via a constructor performing deep copy.
6. Perform cloning of objects in the getter methods to return a copy rather than returning the actual object reference.

Age.java

```
---------
public class Age {

	private int day;
	private int month;
	private int year;

	public int getDay() {
		return day;
	}

	public void setDay(int day) {
		this.day = day;
	}

	public int getMonth() {
		return month;
	}

	public void setMonth(int month) {
		this.month = month;
	}

	public int getYear() {
		return year;
	}

	public void setYear(int year) {
		this.year = year;
	}

}
```

ImmutableStudent.java

```
--------------------------------
package com.jit.immutable;

public final class ImmutableStudent {
```

```
        private final int id;
        private final String name;
        private final Age age;

        public ImmutableStudent(int id, String name, Age age) {
                this.name = name;
                this.id = id;
                Age cloneAge = new Age();
           cloneAge.setDay(age.getDay());
           cloneAge.setMonth(age.getMonth());
           cloneAge.setYear(age.getYear());
           this.age = cloneAge;
        }

        public int getId() {
                return id;
        }

        public String getName() {
                return name;
        }

        public Age getAge() {
                Age cloneAge = new Age();
                cloneAge.setDay(this.age.getDay());
                cloneAge.setMonth(this.age.getMonth());
                cloneAge.setYear(this.age.getYear());
                return cloneAge;
        }
}
```

13. What is the difference between Comparable and Comparator?

Comparable	Comparator
It is meant for default natural sorting order.	It is meant for customized sorting order.
Present in java.lang package.	Present in java.util package.
It defines only one method compareTo(Object obj)	It defines two methods compare(Object ob1, Object ob2) and equals().

String and all Wrapper classes implements Comparable interface.	The only implemented classes of Comparator are Collator, RuleBasedCollator

14. **Why is the String class immutable?**

    String is immutable for several reasons, here is a summary:-

    Security: parameters are typically represented as String in network connections, database connection urls, usernames/passwords etc. If it were mutable, these parameters could be easily changed.

    Synchronization and concurrency: making String immutable automatically makes them thread safe thereby solving the synchronization issues.

    Caching: when compiler optimizes your String objects, it sees that if two objects have same value (a="test", and b="test") and thus you need only one string object (for both a and b, these two will point to the same object).

    Class loading: String is used as arguments for class loading. If mutable, it could result in wrong class being loaded (because mutable objects change their state).

15. **What is the use of volatile keyword?**

    The volatile variable in Java is a special variable that is used to signal threads, a compiler that the value of this particular variable is going to be updated by multiple threads inside the Java application.By making a variable volatile using the volatile keyword in Java, the application programmer ensures that its value should always be read from the main memory and the thread should not use the cached value of that variable from their own stack.

16. **Explain Serialization and Deserialization and Externalization**

    REF :- https://www.geeksforgeeks.org/serialization-in-java/

    https://www.geeksforgeeks.org/difference-between-serializable-and-externalizable-in-java-serialization/

17. **What is Optional in Java8?**

    https://www.tutorialspoint.com/java8/java8_optional_class.htm#:~:text=Optional%20is%20a%20container%20object%20used%20to%20contain%20not%2Dnull%20objects.&text=This%20class%20has%20various%20utility,what%20Optional%20is%20in%20Guava.

    Java 8 has introduced a new class Optional in java.util package. It can help in writing a neat code without using too many null checks. By using Optional, we can specify alternate values to return or alternate code to run. This makes the code

more readable because the facts which were hidden are now visible to the developer.

```
Optional<Foo> possibleFoo = doSomething();
if (possibleFoo.isPresent()) {
   Foo foo = possibleFoo.get();
   // ...use the foo object...
}

else {
   // ...handle case of missing Foo...
}
```

Method	Description
equals(Object obj)	Indicates whether some other objects is "equal to" this Optional.
filter(Predicate<? super T>predicate)	If a value is present, and this value matches the given predicate, return an **Optional** describing the value, otherwise return an empty **Optional**.
flatMap(Function<? super T, Optional<U> mapper)	If a value is present, apply the provided Optional-bearing mapping function to it, return that result, otherwise return an empty Optional.
get()	If a value is present in this Optional, returns the value, otherwise throws **NoSuchElementException**.
hashCode()	Returns the hash code value of the present value, if any, or 0 (zero) if no value is present.
ifPresent(Consumer<? Super T>consumer)	If a value is present, invoke the specified consumer with the value, otherwise do nothing.

Method	Description
isPresent()	Returns true if there is a value present, otherwise false.
map(Function<? super T, ? extends U> mapper)	If a value is present, apply the provided mapping function to it, and if the result is non-null, return an **Optional** describing the result.
orElse(T other)	Return the value if present, otherwise return other.
orElseGet(Supplier<? extends T> other)	Return the value if present, or invoke other and return the result of that invocation.
orElseThrow(Supplier<? extends X> exceptionSupplier)	Return the contained value, if present, otherwise throw an exception to be created by the provided supplier.
toString()	Returns a non-empty string representation of this optional suitable for debugging.

18. Explain cloning and what are the types of cloning?

https://www.geeksforgeeks.org/deep-shallow-lazy-copy-java-examples/

The process of creating exactly duplicate object is called cloning. We can perform cloning by using clone() method of object class. We can perform cloning only for cloneable objects. An object is said to be cloneable if and only if the corresponding class implements cloneable interface.

Two types of cloning :-

1. **Shallow Cloning :-The process of creating bitwise copy of an object is called shallow cloning. If the main object contain primitive variables then exactly duplicate object will be created in the cloned object. If the main object contain any reference variable then corresponding object won't be created just duplicate reference variable will be create pointing to old content object. Object class clone() method meant for shallow cloning.**

Cat.java

```
public class Cat {
        int j;

        public Cat(int j) {
                this.j = j;
        }
}
```

Dog.java

```
public class Dog implements Cloneable{
	Cat c;
	int i;
	public Dog(Cat c, int i) {
		this.c = c;
		this.i = i;
	}
	@Override
	protected Object clone() throws CloneNotSupportedException {
		return super.clone();
	}
}
```

ShallowCloning.java

```
public class ShallowCloning {

	public static void main(String[] args)throws
CloneNotSupportedException {
		Cat c =new Cat(20);
		Dog d1 = new Dog(c,10);
		System.out.println(d1.i+" -------- "+d1.c.j);

		Dog d2 = (Dog) d1.clone();
		d2.i=888;
		d2.c.j=999;
		System.out.println(d1.i+" ------ "+d1.c.j);

	}

}
```

2. Deep Cloning :- The process of creating exactly duplicate independent copy including contained object is called Deep cloning. In deep cloning if the main object contain only primitive variables then in the cloned object duplicate copies will be created. If the main object contains any reference variable then the corresponding contained object also will be created in the cloned copy.

Cat.java

```
public class Cat {
	int j;

	public Cat(int j) {
		this.j = j;
```

```
	}
}
```

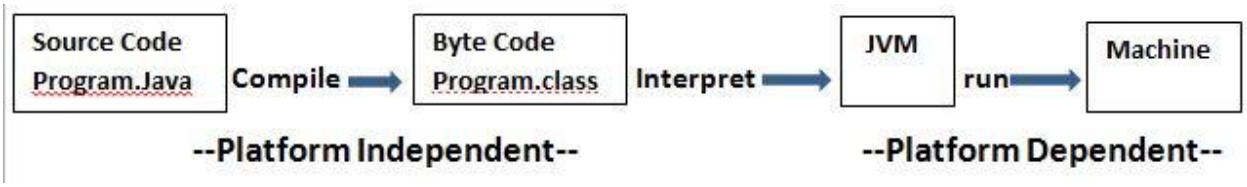
Dog.java

```
-------------
public class Dog implements Cloneable {
	Cat c;
	int i;

	public Dog(Cat c, int i) {
		this.c = c;
		this.i = i;
	}

	@Override
	protected Object clone() throws CloneNotSupportedException {
		Cat c1 = new Cat(c.j);
		Dog d = new Dog(c1, i);
		return d;
	}
}
```

DeepCloning.java

```
------------------------
public class DeepCloning {

	public static void main(String[] args)throws
CloneNotSupportedException {
		Cat c =new Cat(20);
		Dog d1 = new Dog(c,10);
		System.out.println(d1.i+" -------- "+d1.c.j);

		Dog d2 = (Dog) d1.clone();
		d2.i=888;
		d2.c.j=999;
		System.out.println(d1.i+" ------ "+d1.c.j);

	}

}
```

19. Why Java is platform independent?



Using the Java Virtual Machine we can make the byte code understandable to any platform. That is why the byte code is known as platform-independent. But on the other hand the Java Virtual Machine is different for each platform; that is why it is known as platform-dependent. Java is platform-independent because it does not depend on any type of platform. Hence, Java is platform-independent language.

Important Points :-

*)In Java, programs are compiled into byte code and that byte code is platform-independent.

*)The byte code is executed by the Java Virtual Machine and the Java Virtual Machine is platform dependent.

*)Java is platform-independent.

*)Any machine to execute the byte code needs the Java Virtual Machine.

20. Why Java is not pure object oriented language?

Java is not fully object oriented because it supports primitive data type like it, byte,long etc.,which are not objects. That is why JAVA is not 100% object oriented.

21. What is the difference between Runnable and Callable?

Runnable	Callable
If a thread is not required to return anything after completing the job then we should go for Runnable.	If a thread is required to return something after completing the job then we should go for Callable.
Runnable interface contains only one method run().	Callable interface contains only one method call().
Runnable job not required to return anything and hence return type of run() method is void.	Callable job is required to return something and hence return type of call() method is object.
Within the run() method if there is any chance of rising checked exception compulsory we should handle by using try/catch because we can't use throws keyword for run method.	Within the call() method if there is any chance of rising checked exception we are not required to handle by using try/catch because call() method already throws exception.
Runnable interface present in java.lang package.	Callable interface present in java.util.concurrent package.
Introduced in 1.0 version.	Introduced in 1.5 version.

22. How to increase Memory of Java program?

Start the program with -Xms=[size] -Xmx -XX:MaxPermSize=[size] -XX:MaxNewSize=[size]
For example -
-Xms512m -Xmx1152m -XX:MaxPermSize=256m -XX:MaxNewSize=256m

23. What are the types of Garbage Collector?

JVM has five types of GC implementations:
1)Serial Garbage Collector
2)Parallel Garbage Collector
3)CMS Garbage Collector
4)G1 Garbage Collector
5)Z Garbage Collector
Reference :- https://www.baeldung.com/jvm-garbage-collectors

24. What is atomic variable in thread?

This Java Concurrency tutorial helps you understand the concept of *Atomic Variables* provided by the Java Concurrency API. Look at the `java.util.concurrent.atomic` package you will see the following classes:

```
AtomicBoolean
AtomicInteger
AtomicLong
```

You can think of these are wrapper of primitive types boolean, integer and long, with the difference: they are designed to be safely used in multi-threaded context.
They are called atomic variables because they provide some operations that cannot be interfered by multiple threads. Here's

https://www.codejava.net/java-core/concurrency/understanding-atomic-variables-in-java

25. Use of skip() and limit() in java8?

26. How to create your own custom queue?

```
*Insert from rear end
*Delete from front end
// A class to represent a queue
class Queue
{
        private int[] arr;      // array to store queue elements
        private int front;      // front points to the front element in the queue
        private int rear;       // rear points to the last element in the queue
        private int capacity;   // maximum capacity of the queue
```

```
private int count;      // current size of the queue

// Constructor to initialize a queue
Queue(int size)
{
        arr = new int[size];
        capacity = size;
        front = 0;
        rear = -1;
        count = 0;
}

// Utility function to dequeue the front element
public void dequeue()
{
        // check for queue underflow
        if (isEmpty())
        {
                System.out.println("Underflow\nProgram Terminated");
                System.exit(1);
        }

        System.out.println("Removing " + arr[front]);

        front = (front + 1) % capacity;
        count--;
}

// Utility function to add an item to the queue
public void enqueue(int item)
{
        // check for queue overflow
        if (isFull())
        {
                System.out.println("Overflow\nProgram Terminated");
                System.exit(1);
        }

        System.out.println("Inserting " + item);

        rear = (rear + 1) % capacity;
        arr[rear] = item;
        count++;
}
```

```
	// Utility function to return the front element of the queue
	public int peek()
	{
		if (isEmpty())
		{
			System.out.println("Underflow\nProgram Terminated");
			System.exit(1);
		}
		return arr[front];
	}

	// Utility function to return the size of the queue
	public int size() {
		return count;
	}

	// Utility function to check if the queue is empty or not
	public Boolean isEmpty() {
		return (size() == 0);
	}

	// Utility function to check if the queue is full or not
	public Boolean isFull() {
		return (size() == capacity);
	}
}

class Main
{
	public static void main (String[] args)
	{
		// create a queue of capacity 5
		Queue q = new Queue(5);

		q.enqueue(1);
		q.enqueue(2);
		q.enqueue(3);

		System.out.println("The front element is " + q.peek());
		q.dequeue();
		System.out.println("The front element is " + q.peek());

		System.out.println("The queue size is " + q.size());
```

```
                q.dequeue();
                q.dequeue();

                if (q.isEmpty()) {
                        System.out.println("The queue is empty");
                }
                else {
                        System.out.println("The queue is not empty");
                }
        }
}
```

27. How to create your own custom stack?

```
import java.util.EmptyStackException;

public class Stack {

   private int arr[];
   private int size;
   private int index = 0;

   public Stack(int size) {
      this.size = size;
      arr = new int[size];
   }

   public void push(int element) {

      if (isFull()) {
         throw new StackOverflowError("Stack is full");
      }

      arr[index] = element;
      index++;
   }

   public int pop() {

      if (isEmpty()) {
         throw new EmptyStackException();
      }
      return arr[--index];
   }
```

```
    public boolean isEmpty() {
        if (index == 0) {
            return true;
        }
        return false;
    }

    public boolean isFull() {
        if (index == size) {
            return true;
        }
        return false;
    }

    public int size() {
        return index;
    }
}
================================================

public class StackClient {

    public static void main(String[] args) {

        Stack stack = new Stack(5);
        stack.push(5);
        stack.push(4);
        stack.push(3);
        stack.push(2);
        stack.push(1);

        System.out.println("1. Size of stack after push operations: " + stack.size());

        System.out.printf("2. Pop elements from stack : ");
        while (!stack.isEmpty()) {
            System.out.printf(" %d", stack.pop());
        }

        System.out.println("\n3. Size of stack after pop operations : " + stack.size());
    }
}
```

28. How many objects are eligible for garbage collection?

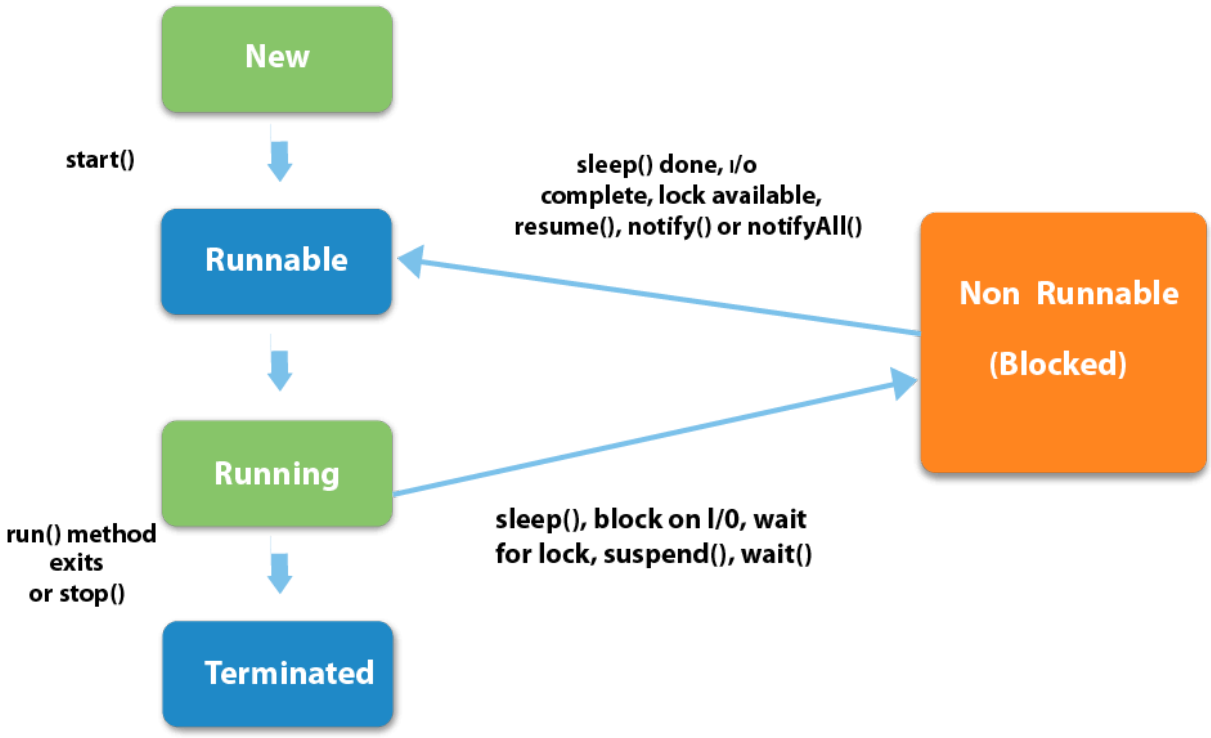
```
class Beta { }

class Alpha {
        Beta b1;
        Beta b2;
}

public class Tester {
        public static void main(String[] args) {
        Beta b1 = new beta();
        Beta b2 = new beta();
        Alpha a1 = new Alpha();
        Alpha a2 = new Alpha();
        a1.b1 = b1;
        a1.b2 = b1;
        a2.b2 = b2;
        a1 = null; b1 = null; b2 = null; // do stuff } }
        When line above line  is reched , how many objects will be eligible for
        garbage collection?
}
```

29. What is the result?

```
class Mixer {
        Mixer() { }
        Mixer(Mixer m)
                {
                        ml = m;
                }
        Mixer m1;
        public static void main(String[] args) {
                Mixer m2 = new Mixer();
                Mixer m3 = new Mixer(m2);
                m3.go();
                Mixer m4 = m3.m1;
                m4.go();
                Mixer m5 = m2.m1;
                m5.go();
}
void go() {
        System.out.print("hi ");
        }
}
```

29. Do you know about balanced Tree?
30. Write a program in such way one thread will print 1 to 10, 21-30,41-50,61-70,81-90 and other thread will print 11 to 20, 31-40,51-60,71-80,90-100.

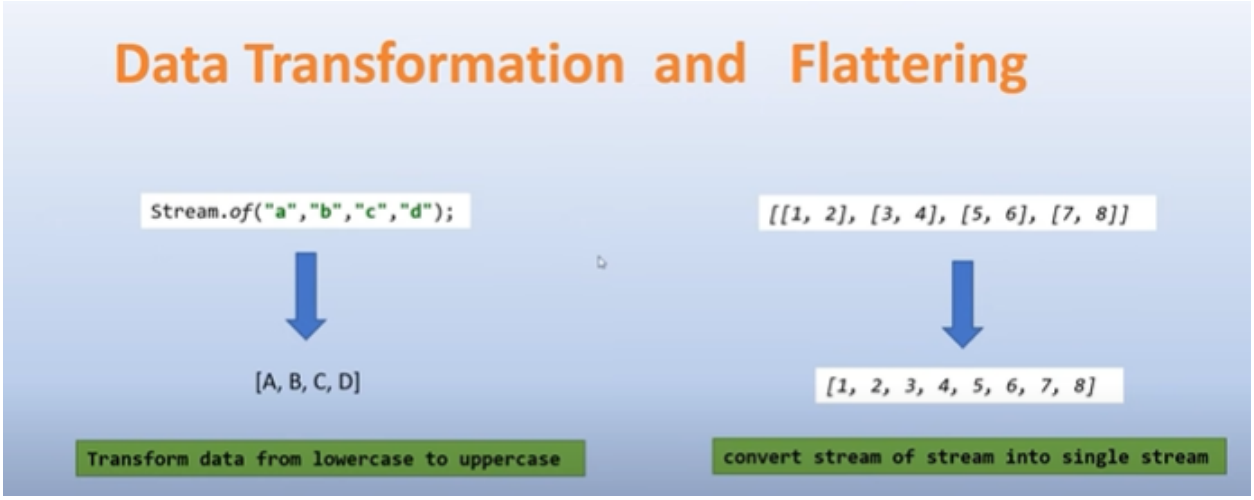
31. Write a program to sort an array without using two loops.
32. Explain the lifecycle of the Thread.



Reference :- https://www.javatpoint.com/life-cycle-of-a-thread

33. What is the difference between map() and flatMap()?
Reference :- https://www.youtube.com/watch?v=CwvIS3ViGFQ
https://github.com/Java-Techie-jt/java8

*map() method is used for transformation.
*flatMap() is used for transformation and flattering.
*flatMap() = map() + flattering.
*map() takes Stream<T> as input and return Stream<R>
*map():- It is a mapper function produces single value for each input value. Hence it is also called as One-To-One mapping.
*flatMap() takes Stream<Stream<T>> as input and return Stream<R>
*flatMap():- It is a mapper function produces multiple value for each input value. Hence it is also called as One-To-Many mapping.



34. What key you have used to configure log4j?

```
appender.file.type = File
appender.file.name = LOGFILE
appender.file.fileName=${filename}/propertieslogs.log
appender.file.layout.type=PatternLayout
appender.file.layout.pattern=[%-5level] %d{yyyy-MM-dd HH:mm:ss.SSS} [%t]
%c{1} - %msg%n
```

35. How to use external server in case of spring boot project?

Reference :- https://codezup.com/deploy-spring-boot-war-to-external-tomcat/

Steps :-

1. Package as war in pom.xml
2. Exclude the tomcat jar using exclusion tag in pom.xml
3. Extend Spring Boot Application with SpringBootServletInitializer

```
@SpringBootApplication
public class YourSpringBootApplication extends
SpringBootServletInitializer {

  public static void main(String[] args) {
    SpringApplication.run(YourSpringBootApplication.class, args);
  }

  @Override
  protected SpringApplicationBuilder configure(SpringApplicationBuilder
builder) {
    return builder.sources(YourSpringBootApplication.class);
  }
```

}

36. What is diff b/w ArrayList and LinkedList?
37. Can you explain the internal flow of HashMap?
38. what is the diff b/w HashMap and Hashtable?

39. **HashMap is non-synchronized. It is not thread-safe and can't be shared between many threads without proper synchronization code whereas Hashtable is synchronized. It is thread-safe and can be shared with many threads.**
40. **HashMap allows one null key and multiple null values whereas Hashtable doesn't allow any null key or value.**
41. **HashMap is generally preferred over HashTable if thread synchronization is not needed**

42. Diff b/w Array and ArrayList?
43. Diff b/w ArrayList and Vector?
44. In your project where you used cuncurrent hashmap?

https://www.geeksforgeeks.org/concurrenthashmap-in-java/
Note:- Hashtable does not allow null keys but HashMap allows one null key and any number of null values
The ConcurrentHashMap class is introduced in JDK 1.5 belongs to java.util.concurrent package, which implements ConcurrentMap as well as to Serializable interface also.

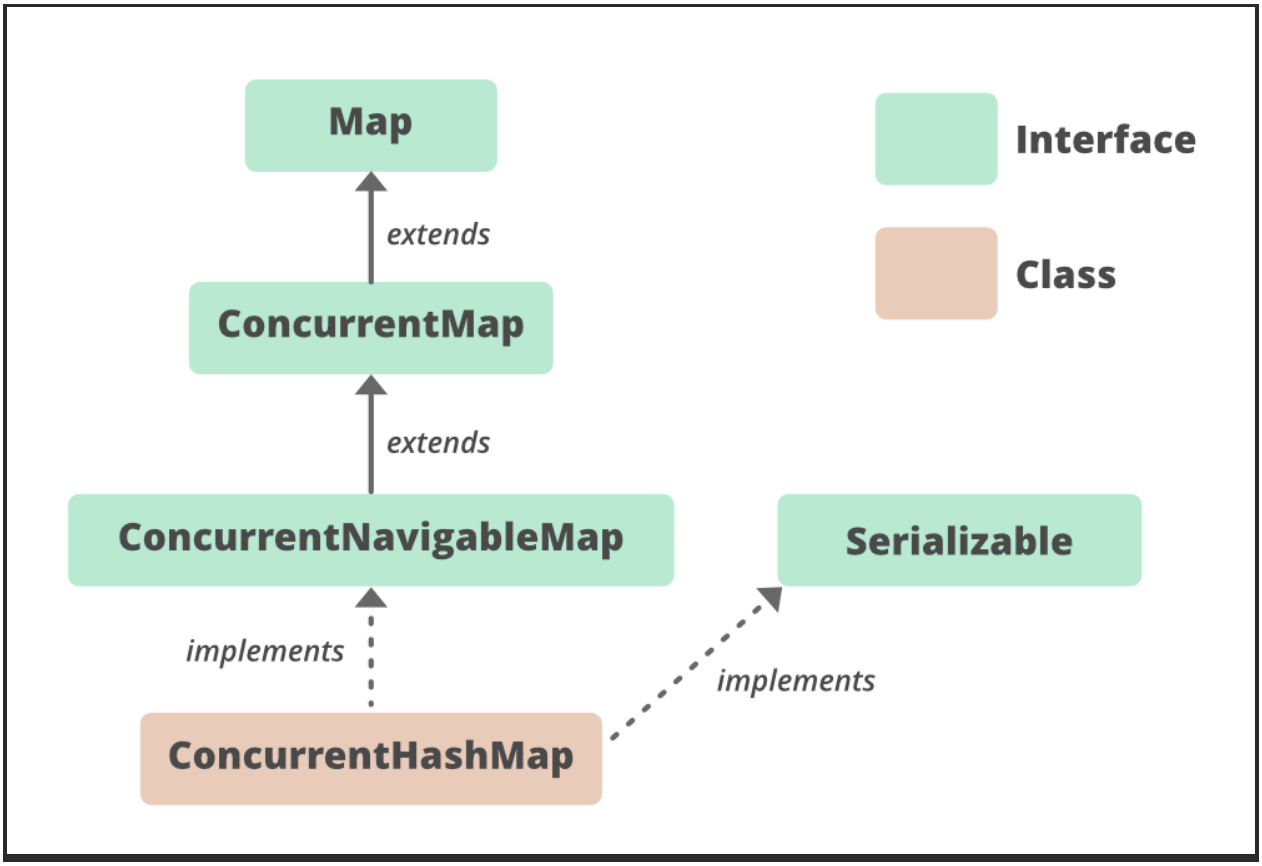
Key points of ConcurrentHashMap:

- **The underlined data structure for ConcurrentHashMap is Hashtable.**
- **ConcurrentHashMap class is thread-safe i.e. multiple threads can operate on a single object without any complications.**

- **At a time any number of threads are applicable for a read operation without locking the ConcurrentHashMap object which is not there in HashMap.**
- **In ConcurrentHashMap, the Object is divided into a number of segments according to the concurrency level.**
- **The default concurrency-level of ConcurrentHashMap is 16.**
- **In ConcurrentHashMap, at a time any number of threads can perform retrieval operation but for updated in the object, the thread must lock the particular segment in which the thread wants to operate. This type of locking mechanism is known as Segment locking or bucket locking. Hence at a time, 16 update operations can be performed by threads.**
- **Inserting null objects is not possible in ConcurrentHashMap as a key or value.**

public class ConcurrentHashMap<K,V> extends AbstractMap<K,V> implements ConcurrentMap<K,V>, Serializable

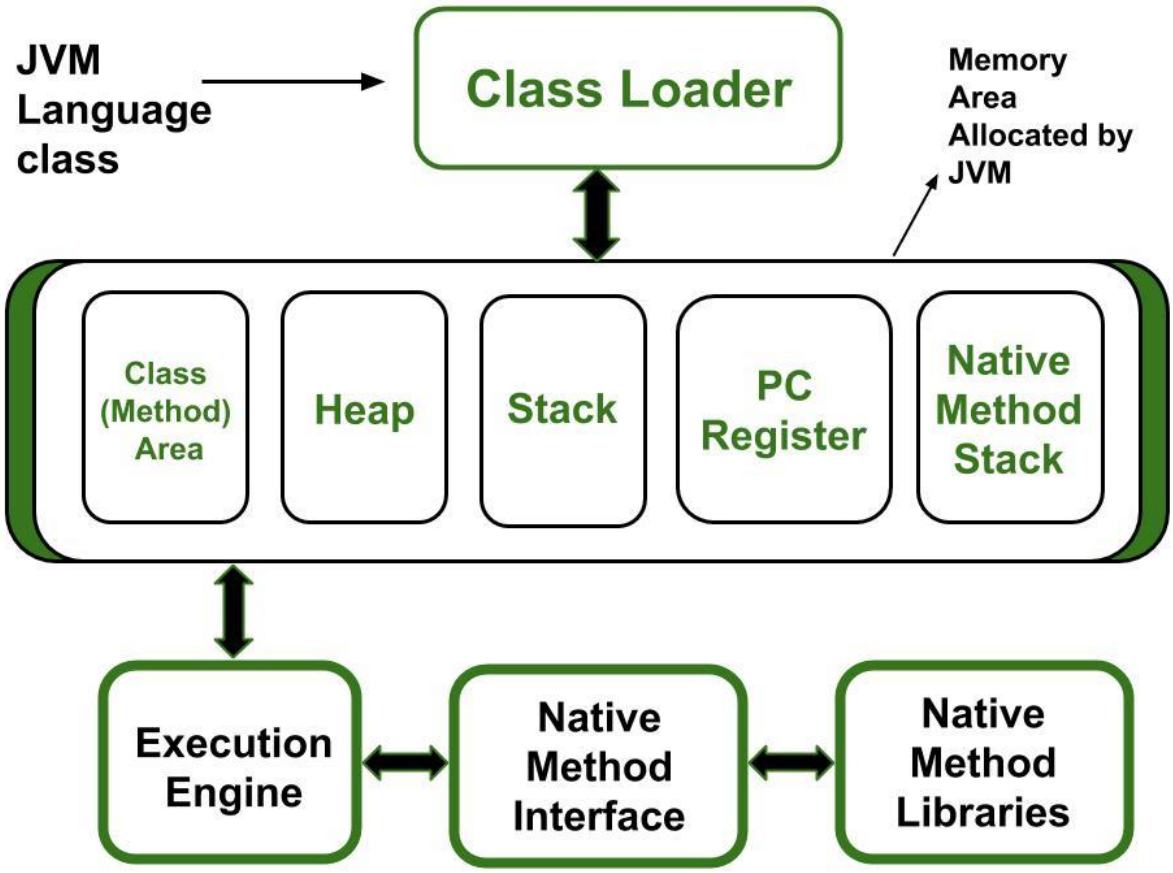
- ***Concurrency-Level: It is the number of threads concurrently updating the map. The implementation performs internal sizing to try to accommodate this many threads.***
- ***Load-Factor: It's a threshold, used to control resizing.***
- ***Initial Capacity: Accommodation of a certain number of elements initially provided by the implementation. if the capacity of this map is 10. It means that it can store 10 entries.***



45. What is java annoying?
46. Diff b/w callable interface and future interface in concurrent package?
    https://www.geeksforgeeks.org/callable-future-java/

    Callable and Runnable both are interfaces whose instances are executed by a thread. But Callable is capable of returning some values whereas runnable is not. Runnable can be executed by Thread class as well as ExecutorService where as callable can be executed by ExecutorService only. The value returned after executorService.submit(new callable<Integer>) is Future<Integer>.
    Callable class method throws Checked Exception.

    To create the thread, a Runnable is required. To obtain the result, a Future is required.
47. What is class loaders?
48. How can you take List into Map?
49. What types of memories are available in java?



50. How can you take Map into List?

List<Map<integer,string>>

51. when you will get ClassNotFoundException and NoClassDefFoundError?

ClassNotFoundException	NoClassDefFoundError
It is an exception. It is of type java.lang.Exception.	It is an error. It is of type java.lang.Error.
It occurs when an application tries to load a class at run time which is not updated in the classpath.	It occurs when java runtime system doesn't find a class definition, which is present at compile time, but missing at run time.
It is thrown by the application itself. It is thrown by the methods like Class.forName(), loadClass() and findSystemClass().	It is thrown by the Java Runtime System.
It occurs when classpath is not updated with required JAR files.	It occurs when required class definition is missing at runtime.

52. How you implement exception handling in your project?

53. where you implement multi-threading in your project?
    Lakhi will show the code
54. Find the output of below program.
    String a="a";
    String b=new String("a");
    System.out.println((a==b));// false
    String c=b.intern();
    System.out.println((a==c)); true

The Java String class intern() method returns the interned string. It returns the canonical representation of string.

It can be used to return string from memory if it is created by a new keyword. It creates an exact copy of the heap string object in the String Constant Pool.

55. what is diff b/w String str="xyz"; and String str2= new String("xyz");
56. What are the java coding standard you follow?
57. Expalin about java architectue?
58. Difference between Collection and Collections?
    Collection is root interface of collections framework.
    Collection represents the group of individual objects as a single unit.
    Collection is root interface for data structures like List, Set, Queue.
    Whereas Collections is utility class which work for collection.
    All the methods in Collections are Static. They are used as utitlity for collection like if we want to find maximum and minimum elements in collection
    or we want to sort the unsorted collection then we can use Collections methods
59. Explain about jvm architecture?
    The main function of JVM is to load and execute the application.

60. How many types of memory available in java?

61. what is diff b/w throw and throws?

    https://www.geeksforgeeks.org/throw-throws-java/

    The throw keyword in Java is used to explicitly throw an exception from a method or any block of code. We can throw either checked or unchecked exception. The throw keyword is mainly used to throw custom exceptions.

Example:

```
throw new ArithmeticException("/ by zero");
```

```
// Java program that demonstrates the use of throw
class ThrowExcep
{
    static void fun()
    {
        try
        {
            throw new NullPointerException("demo");
        }
        catch(NullPointerException e)
        {
            System.out.println("Caught inside fun().");
            throw e; // rethrowing the exception
        }
    }

    public static void main(String args[])
    {
        try
        {
            fun();
        }
        catch(NullPointerException e)
        {
```

```
                System.out.println("Caught in main.");
            }
        }
}
```

throws is a keyword in Java which is used in the signature of method to indicate that this method might throw one of the listed type exceptions. The caller to these methods has to handle the exception using a try-catch block.

Syntax:

```
type method_name(parameters) throws exception_list
```

exception_list is a comma separated list of all the exceptions which a method might throw.

We can use throws keyword to delegate the responsibility of exception handling to the caller (It may be a method or JVM) then caller method is responsible to handle that exception.

```
// Java program to illustrate throws
class tst
{
    public static void main(String[] args)throws InterruptedException
    {
        Thread.sleep(10000);
        System.out.println("Hello Geeks");
    }
}
```

- **throws** keyword is required only for **checked exception** and usage of throws keyword for unchecked exception is meaningless.
- throws keyword is required only to convince compiler and usage of throws keyword does not prevent abnormal termination of program.
- By the help of throws keyword we can provide information to the caller of the method about the exception.

62. can you tell me java8 features?

Lambda expressions,

Method references,

Functional interfaces,

Stream API- **Introduced in Java 8, the Stream API is used to process collections of objects. A stream is a sequence of objects that supports various methods which can be pipelined to produce the desired result.**

Default methods,

Static methods in interface,

Optional class,

Collectors class,

ForEach() method,

63. can i add elements to list , if it is defined as final?
ex:final List<String> list= new ArrayList<>();

64. if you pass duplicate key to map what will happen?
If the key is duplicate then the value would be updated and the same value will be returned as the output.

65. Diff b/w abstract class and interface?

66. Why is null stored at zero index in hashmap?
From the source code of HashMap, if the key is null it is handled differently. There is no hashcode generated for null, but it is uniquely stored at index 0 in an internal array with hash value 0. Also note that hash value of an empty string also is 0(in case keys are strings), but the index where it is stored in the internal array ensures that they are not mixed up.

We can store only one null key in Hashmap.

67. How to use JAXON API and Object mapper class.?

68. What is wrapper class?
The wrapper class in Java provides the mechanism *to convert primitive into object and object into primitive*.
AutoBoxing

```
int a=20;

Integer i=Integer.valueOf(a);//converting int into Integer explicitly
```

Unboxing

The automatic conversion of wrapper type into its corresponding primitive type is known as unboxing.

```
Integer a=new Integer(3);
int i=a.intValue();//converting Integer to int explicitly
int j=a;//unboxing, now compiler will write a.intValue() internally
```

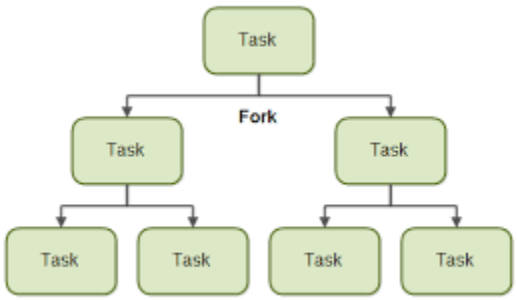
69. What is Interface?

https://docs.oracle.com/javase/tutorial/java/concepts/interface.html

an interface is a group of related methods with empty bodies. A bicycle's behavior, if specified as an interface, might appear as follows:

```
interface Bicycle {

    //  wheel revolutions per minute
    void changeCadence(int newValue);

    void changeGear(int newValue);

    void speedUp(int increment);

    void applyBrakes(int decrement);
    }
```

70. Do you know about Fork Join?



The fork-join framework allows to break a certain task on several workers and then wait for the result to combine them. It leverages multi-processor machine's capacity to great extent

Fork is a process in which a task splits itself into smaller and independent sub-tasks which can be executed concurrently.

```
Sum left  = new Sum(array, low, mid);
left.fork();
```

Join is a process in which a task join all the results of sub-tasks once the subtasks have finished executing, otherwise it keeps waiting.

```
left.join();
```

ForkJoinPool

it is a special thread pool designed to work with fork-and-join task splitting.

Syntax

```
ForkJoinPool forkJoinPool = new ForkJoinPool(4);
```

https://www.tutorialspoint.com/java_concurrency/concurrency_fork_join.htm

71. **Write a program to show dead lock case.**

A.Java

======

```
public class A {

	public synchronized void d1(B b) {
		System.out.println("Thread1 starts execution of d1() method");
		try {
			Thread.sleep(6000);
		}catch (InterruptedException e) {
			e.printStackTrace();
		}
		System.out.println("Threa1 trying to call B's last()");
		b.last();
	}

	public synchronized void last() {
		System.out.println("Inside A this is the last method");
	}

}
```

B.java
======

```
public class B {

	public synchronized void d2(A a) {
		System.out.println("Thread2 starts execution of d2() method");
		try {
			Thread.sleep(6000);
		}catch (InterruptedException e) {
			e.printStackTrace();
		}
		System.out.println("Threa2 trying to call A's last()");
		a.last();
	}

	public synchronized void last() {
		System.out.println("Inside B this is the last method");
	}

}
```

DeadLockDemo.java
=================

```
public class DeadLockDemo extends Thread {

	A a = new A();
	B b = new B();

	public void m1() {
		this.start();
		a.d1(b);// This is executed by main thread
	}

	public void run() {
		b.d2(a);// THis is executed by child thread
	}

	public static void main(String[] args) {
		DeadLockDemo d =new DeadLockDemo();
		d.m1();
	}
}
```

72. **Write a program to show producer consumer case.**

Producer.java

==========

```
import java.util.List;
public class Producer implements Runnable {

	List<Integer> sharedList = null;
	final int MAX_SIZE = 5;
	private int i = 0;

	public Producer(List<Integer> sharedList) {
		this.sharedList = sharedList;
	}

	@Override
	public void run() {
		while (true) {
			try {
				produce(i++);
			} catch (InterruptedException e) {
				e.printStackTrace();
			}
		}

	}

	public void produce(int i) throws InterruptedException {
		synchronized (sharedList) {
			while (sharedList.size() == MAX_SIZE) {
				System.out.println("Shared List is full....waiting for the
consumer to consume");
				sharedList.wait();
			}
		}

		synchronized (sharedList) {
			System.out.println("Producer produced elemnt : "+i);
			sharedList.add(i);
			Thread.sleep(100);
			sharedList.notify();
		}
	}
}
```

Consumer.java
==========

```
import java.util.List;

public class Consumer implements Runnable {

	List<Integer> sharedList = null;


	public Consumer(List<Integer> sharedList) {
		this.sharedList = sharedList;
	}

	@Override
	public void run() {
		while (true) {
			try {
				consume();
			} catch (InterruptedException e) {
				e.printStackTrace();
			}
		}

	}

	public void consume() throws InterruptedException {
		synchronized (sharedList) {
			while (sharedList.isEmpty()) {
				System.out.println("Shared List is empty....waiting for
the producer to produce");
				sharedList.wait();
			}
		}

		synchronized (sharedList) {
			Thread.sleep(1000);
			System.out.println("Consumed the element :"+
sharedList.remove(0));
			sharedList.notify();
		}
	}
}
```

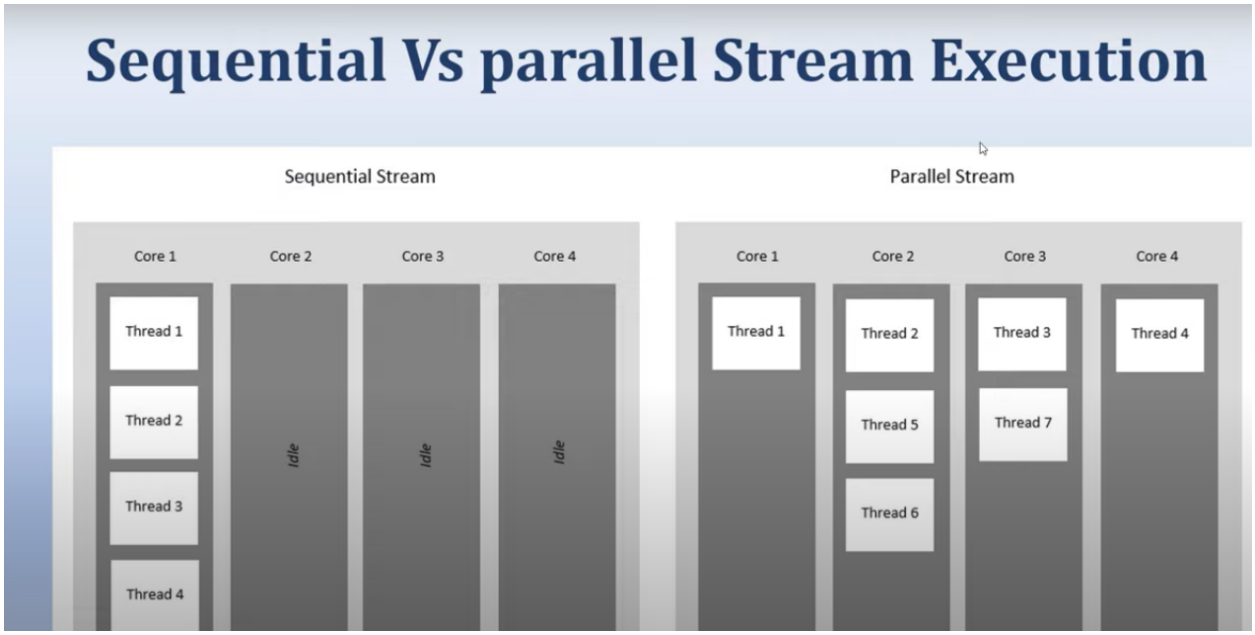
ProducerConsumerDemo.java

```
==========================
package com.jit.producerconsumer;
/*https://www.youtube.com/watch?v=EtJALCEIxDs*/
import java.util.ArrayList;
import java.util.List;

public class ProducerConsumerDemo {

	public static void main(String[] args) {
		List<Integer> sharedList = new ArrayList<Integer>();
		Thread thread1 = new Thread(new Producer(sharedList));
		Thread thread2 = new Thread(new Consumer(sharedList));
		thread1.start();
		thread2.start();
	}

}
```

73. **Explain the Parallel Stream.**

Reference :- https://www.youtube.com/watch?v=J7YqYlaev7g
https://github.com/Java-Techie-jt/java8

***Java Parallel Streams is a feature of Java8, It meant for utilizing multiple cores of the processor.**
***Normally any java code has one stream of processing, where it is executed sequentially. Whereas by using parallel streams, we can divide the code into multiple streams that are executed in parallel on separate cores and the final result is the combination of individual outcomes.**
***The order of execution, however, is not under our control.**



The parallel() method of the BaseStream interface returns an equivalent parallel stream.
Here we are getting text as a stream.

```
text.parallel().forEach(System.out::println);
```

The parallelStream() method of the Collection interface returns a possible parallel stream with the collection as the source
Here we are getting text as a list.

```
text.parallelStream().forEach(System.out::println);
```

74. Convert list to map using Stream.

```
public class Animal {
   private int id;
   private String name;

   //  constructor/getters/setters
}
======================
public Map<Integer, Animal> convertListAfterJava8(List<Animal> list) {
   Map<Integer, Animal> map = list.stream()
     .collect(Collectors.toMap(Animal::getId, Function.identity()));
   return map;
}
```

```
static <T> Function<T, T> identity() {

        return t -> t;

   }
```

Function<T, T> : First parameter is type of input of function and second parameter is type of output, identity function always returns its input arguments.

f(x)=x;

75. Predefined default method in Java8.

76. How to create your own checked & unchecked exception.

https://www.tutorialspoint.com/how-can-we-decide-that-custom-exception-should-be-checked-or-unchecked-in-java

All exceptions must be a child of Throwable.

If you want to write a checked exception that is automatically enforced by the Handle or Declare Rule, you need to extend the Exception class.

If you want to write a runtime(Unchecked) exception, you need to extend the RuntimeException class.

```
class NotProperNameException extends Exception {
   NotProperNameException(String msg){
       super(msg);
   }
}
```

```
class NotProperNameException extends RuntimeException {
   NotProperNameException(String msg){
       super(msg);
   }
```

```
}
```

77. **Find only first odd number in list using Stream API.**

```
List<Integer> list = new ArrayList<Integer>();
                    list.add(2);
                    list.add(8);
                    list.add(3);
                    list.add(7);
                    list.add(10);
                    list.add(99);
                    list.add(20);
                    Optional<Integer> in = list.stream().filter(i -> i % 2 != 0).findFirst();
                    System.out.println(in.get());
```

78. **What is Map-Reduce?**

Reference :- https://www.youtube.com/watch?v=w-iwyp_A7e8
https://github.com/Java-Techie-jt/java8

https://www.java67.com/2016/09/map-reduce-example-java8.html

What is Map-Reduce ?

- Map-Reduce is a functional programming model it serves our 2 purpose

  Map ⟶ Transforming data

  Reduce ⟶ Aggregating data

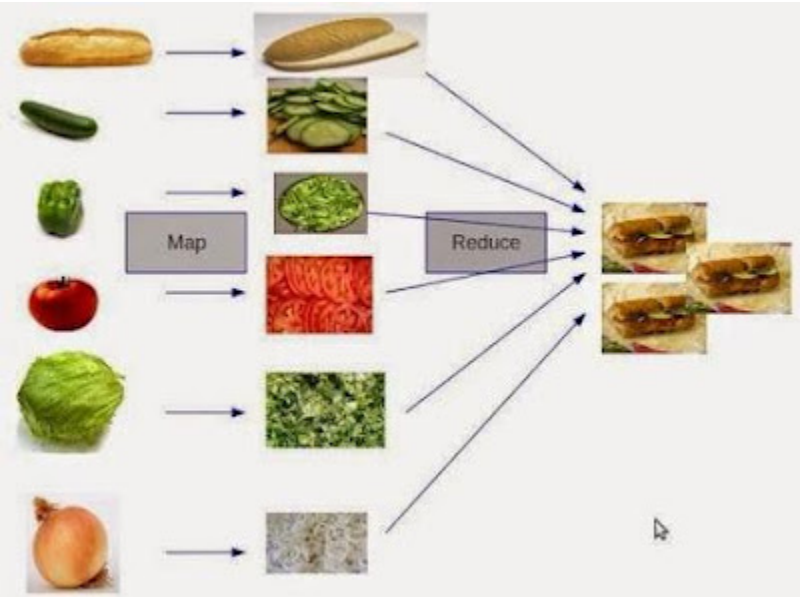
  (combine elements of a stream and produces a single value)

- Ex : Stream : [2,4,6,9,1,3,7] Sum of numbers present in stream?

- Map() → Transform Stream<Object> to Stream of int
- Reduce() → combine stream of int and produce the sum result

```
double average = peoples.stream().mapToInt(p-> p.getAge())
                        .average().getAsDouble();
```

This uses the concept of parallelism, where it creates a parallel stream out of the array, which can be processed by multiple cores and then finally joined back into to map the results together
The map function will create a stream containing only the values with meet the given criteria, then the average function will reduce this map into a single value.



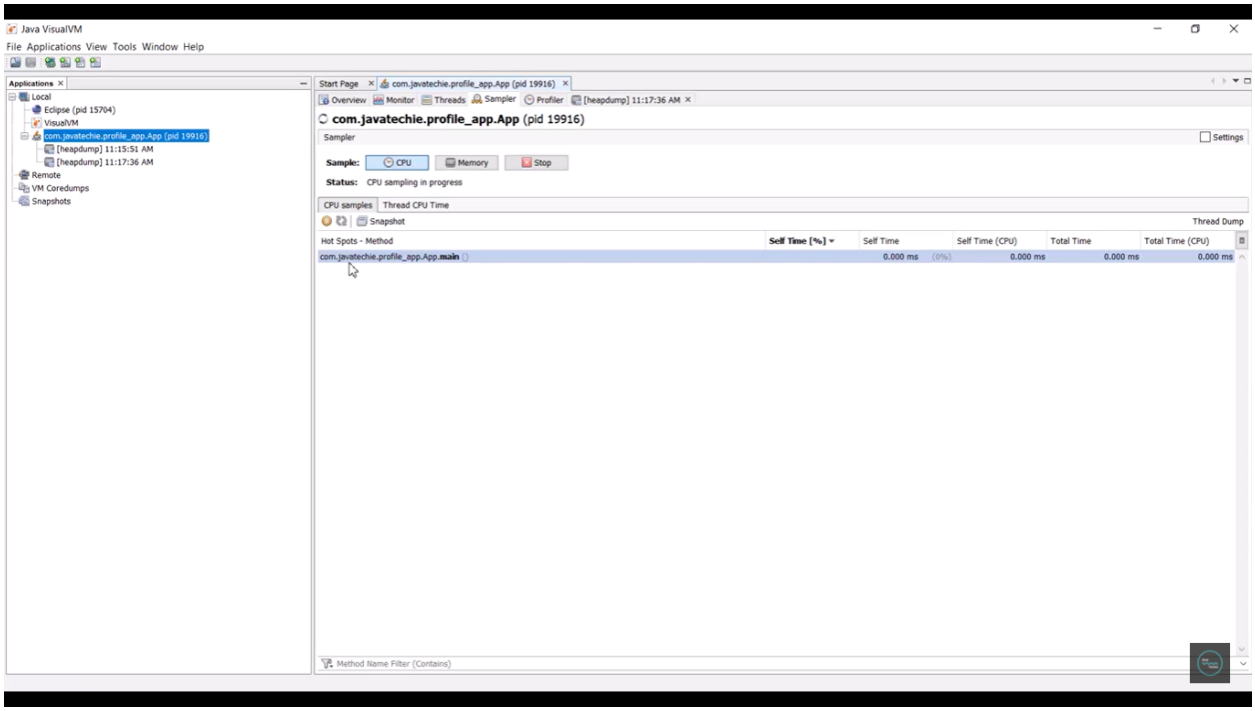
Read more:
https://www.java67.com/2016/09/map-reduce-example-java8.html#ixzz76vAHgzbl

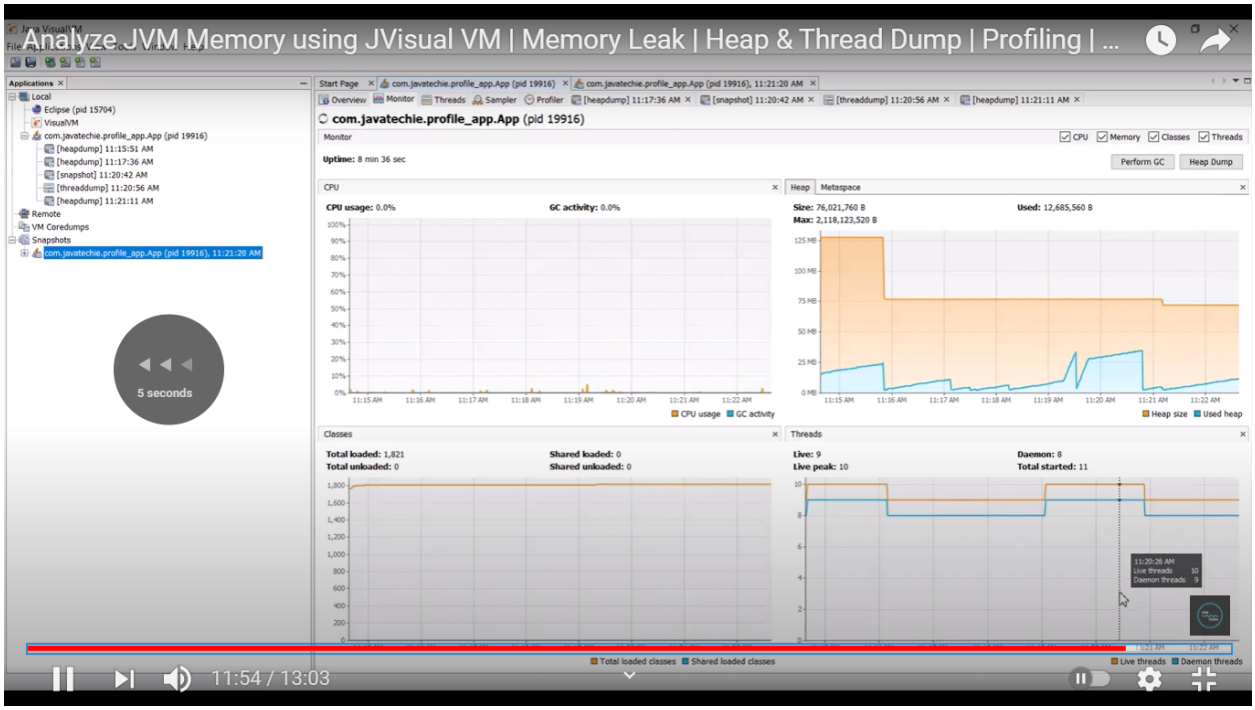
79. What is memory profiling?

https://www.dynatrace.com/support/help/how-to-use-dynatrace/diagnostics/memory-profiling/

https://www.youtube.com/watch?v=AHLkbqcVLvY

Memory profiling enables you to understand the memory allocation and garbage collection behavior of your applications over time.
Java Visual VM (Application)-- this application is useful for getting the memory related information which is running on top of JVM.
Java Visual VM extract the data from the JVM and show the information in the user interface.
Visual vm is by default available to our JDK under the /bin directory of jdk.
It helps to monitor memory leak , analyse the heap and thread dump and monitor the GC.





80. How to check in your system, which port is free.

    netstat -an

81. How to do remote debugging.

Starting the Application With Remote Debugging Enabled

java -Xdebug -Xrunjdwp:transport=dt_socket,address=8998,server=y -jar myapp.jar

The above command says: start myapp.jar + start a server socket at port 8998 and publish the debugging messages using the Java Debug Wire Protocol (jdwp) there.

Other ways is just configure debugger by using IDE remote java application debuggerr configuration.

https://dzone.com/articles/how-debug-remote-java-applicat

82. How to resolve outofmemory error.

    We can increase the value of heap memory. Or while setup server we need to do greatly by keeping the userbase and the behaviour and then decide the value for memory.

    ```
    -Xms2048m -Xmx2048m
    ```

    Xms - the minimum size of the heap

    Xmx - the maximum size of the heap

83. Have you worked in linux system.

    https://files.fosswire.com/2007/08/fwunixref.pdf

84. How to translate business requirementm into technical trem.
85. Thread Enhancement in Java8.

https://docs.oracle.com/javase/8/docs/technotes/guides/concurrency/changes8.html

86. Explain Executor Framework?

https://www.geeksforgeeks.org/what-is-java-executor-framework/
https://www.javatpoint.com/executor-framework-java\

Java executor framework (java.util.concurrent.Executor), released with the JDK 5 is used to run the Runnable objects without creating new threads every time and mostly re-using the already created threads

The java.util.concurrent.Executors provide factory methods that are being used to create ThreadPools of worker threads. Thread pools overcome this issue by keeping the threads alive and reusing the threads. Any excess tasks flowing in that the threads in the pool can handle are held in a Queue. Once any of the threads get free, they pick up the next task from this queue.

SingleThreadExecutor
ExecutorService executor = Executors.newSingleThreadExecutor();

FixedThreadPool(n)
ExecutorService fixedPool = Executors.newFixedThreadPool(2);

CachedThreadPool
ExecutorService executorService = Executors.newCachedThreadPool();

ScheduledExecutor
ScheduledExecutorService scheduledExecService = Executors.newScheduledThreadPool(1);

87. What will be the output?

```
class MyClass{
        public static void m1(int a ,long b) {
                System.out.println(1);
        }
        public st = newatic void m1(long b,int a ) {
                System.out.println(2);
        }
}

class Test{
        public static void main(String args[]){
                        MyClass c MyClass();
```

```
                c.m1(10,20);
            }
        }
```

85. What will be the output?

```
class A{
        display(){
            syso("displayA");
        }
}
class B extends A{
        display(){
            syso("displayB");
        }
main(){
        B b = new B();
        A a = new A();
        A a1 = new B();
        B b1 = new A();
        }
}
```

88. What will be the output?

- finally block overrides any return values from try and catch blocks.

https://javaconceptoftheday.com/return-value-from-try-catch-finally-blocks/

```
Try{
        return 1;
}catch(Exception e){
        return 2;
}
}finally{
        return 3;
}
```

89. What is Stream API?
90. WHat is a concurrent HashMap?
91. Difference between HashMap and hashtable?
92. Implement IOT using Java Design pattern.
93. Different ways to create object.
94. Will I get ConcurrentModificationException in below code :-

```
List<Integer> ll = new ArrtayList<>();
ll.add(1);
ll.add(2);
for(Integer i : ll) {
   if(i == 1) {
      ll.add(3);
   }
}
sop(ll);
```

O/P Exception in thread "main" java.util.ConcurrentModificationException
at java.util.ArrayList$Itr.checkForComodification(ArrayList.java:909)
at java.util.ArrayList$Itr.next(ArrayList.java:859)
at com.lucky.test.ps1.Test3.main(Test3.java:14)

95. Create the Immutable class for below :-

```
class Car{
        String carName;
        Date date;
}
```

96. What will be the output for below program :-

```
class B {
   public Integer run(String str) {
      System.out.println(str);
                return 1;
   }
   public Integer run(Object obj) {
                System.out.println(obj);
                return 1;
   }
   public static void main(String... ) {
      B b = new B()
      b.run(null);
   }
}
```

97. What will be the output for below program :-

```
class A {
   public Number run() {
      System.out.println("A");
   }
}
```

```
class B extends A{
   public Integer run() {
      System.out.println("B");
   }
   public static void main(String... ) {
      A a = new B()
      a.run();
   }
}
```

98. What will be output for below program:-

```
class A{
        public void display(){
        syso("displayA");
        }
}
class B extends A{
        public void display(){
                syso("displayB");
        }
        public static void main(String args[]){
                B b = new B();
                A a = new A();
                A a1 = new B();
                B b1 = new A();
        }
}
```

99. What will be output for below program:-

```
class B {

   public void main(String[] args) {
      sop("B");
   }
   public static void main(String a, String b) {
      sop("A");
   }
}
```

100. What will be the output for below program:-

```
class B {

   public static void main(String[] args) {
      sop("B");
   }
   public static void main(String a, String b) {
```

```
        sop("A");
    }
}
```

101. What will be the output for below program?

```
class A {
    public void run() {
        System.out.println("A");
    }
}



class B extends A{
    public static void main(String... ) {
        B b = new A();
                    A a=new B();
        a.run();
    }
}
```

102. What is the output for below program :-

```
class A {
    A() {
                    sop("A");
    }
}


class B extends A{
    B() {
        sop("B");
    }
    public static void main(String... ) {
        A a = new B();

    }
}
```

103. What will be the output for below program :-

```
class A {
    public void run() {
        System.out.println("A");
    }
}
class B extends A{
```

```
    public void run() throws NullPointer {
       System.out.println("B");
    }
    public static void main(String... ) {
       B b = new A();
       b.run();
                A a = new B();
                a.run();
    }
}
```

101. How to mock the private methods?
102. Have you used power mockito?
103. Which of the following options is the correct output of the program? Disregard new lines.

```
public static void main(String[] args){
        String s1 = "Hello";
        String s2 = new String(s1);
        String s3 = "Hello";

        System.out.println("s1 == s2:" + (s1 == s2));
        System.out.println("s1 == s3:" + (s1 == s3));
        System.out.println("s1.equals(s2):" + s1.equals(s2));
}
```

a. s1 == s2: true, s1 == s3: true, s1.equals(s2): true
b. s1 == s2: false, s1 == s3: false, s1.equals(s2): true
c. s1 == s2: true, s1 == s3: true, s1.equals(s2): false
d. s1 == s2: false, s1 == s3: true, s1.equals(s2): true

104. What is terminal operation in Java8.
105. Write no from 1 to 100 using Stream.

```
IntStream.range(1, 10).forEach(System.out::println);
```

106. How You can make HashMap thread safe?
You can make HashMap thread safe by using Collections.synchronizedMap(Map)
107.What is concurrent hashmap ?
108.How concurrent hashmap is better then thread safe hashmap in multi threading environment ?
109.How to find middle element of Linked List?
110.How to find a loop in LinkedList ?

```
public static boolean detectLoop(Node head)
        {
                Node slow=head;
                Node fast=head;
```

```
        while(fast!=null && fast.next!=null)
        {
                slow=slow.next;
                fast=fast.next.next;
                if(slow==fast)
                {
                        System.out.println("Loop detected");
                        return true;
                }
        }
        return false;

}
```

111.How to find the starting point of loop in LinkedList ?
112.How to find the length of the loop in LinkedList ?
113. How to run 5 threads sequentially.
114.. Print number 1 to 10 using two threads where thread 1 prints even number and thread 2 prints odd number.
115.. Producer consumer implementation using wait notify.
116. What is deadlock ? How to identify deadlock in java application ? How to prevent deadlock situations in application development ?
117. CountDownLatch vs CyclicBarrier ?
118. Future Object, ThreadLocal etc.
119. How to name a thread in executor service ?
120. How to interrupt a thread explicitly ?
121.Anti patterns in Java. What is god class ?
121.How classes are related to each other through association, aggregation and composition.

Association (bidirectional one to one, one to many, many to one or many to many association, represented by line with arrow in UML) for e.g. Teacher and Student. Multiple students can associate with a single teacher and a single student can associate with multiple teachers but there is no ownership between the objects and both have their own lifecycle.
Aggregation (Has-a relationship, unidirectional association, parent and child can survive individually, represented by line with diamond in UML) for e.g. Car and Wheel. Car can have multiple wheels but wheel can not belong to multiple cars and if we delete the car, wheel need not to be destroyed and used in another car.
Composition (Part-of relationship, unidirectional association, child can not survive without parent, represented by line with filled diamond in UML) for e.g. 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.

122. Explain each keyword in main method i.e. public static void main(String[] args).
123. Can you override a static method and private method ?
124. How many ways you can create an object in java ?
New Keyword, Cloning, Deserialization and Reflection.
125. Explain Java Memory Model with Heap structure. Explain here three parts of heap i.e. Young, Old and Permanent Generation. Also Explain Minor and Major GC.
126. How to do JVM performance tuning ? Explain here parameters.
a) Heap Memory: -Xms, -Xmx, -Xmn,
b) Permanent Generation Memory: -XX:PermSize, -XX:MaxPermSize
c) Garbage Collection i.e.
-XX:+UseSerialGC,
-XX:+UseParellelGC (-XX:ParellelGCThreads=<N>),
-XX:+UseParellelOldGC,
-XX:+UseConcMarkSweepGC (-XX:ParellelCMSThreads=<N>),
-XX:+UseG1GC
127. What is classloader in java ? How to write custom class Loader ? What is linkage error ?
128. Have you face any memory issue, how do you resolved that?
129. In production which server are you using?
130. How to keep your data on multiple servers?
131. How to connect to multiple databases?
132. Is there any purpose to provide setter method?
133. What is functional programming?
134. What is abstraction and encapsulation?
135. We are making field as private but we can modify it using setter method then what is the need of declaring field as private?

==========

SPRING INTERVIEW QUESTIONS

==========

1.What is the difference between Application context and bean factory?
BeanFactory :-
- **Find Beans**
- **Wire Dependencies**
- **Manage Lifecycle of Bean**
- **Creates bean only when factory.getBean(-) method are called, not when IOC container is created so we can say Bean Factory is performing lazy instantiations and injections.**
- **No direct support for properties file and placeholder (${})**
- **Does not support annotation it only support xml.**

- No automatic registration of BeanPostProcessor
- Use bit less memory
- No automatic registration of BeanFactoryPostProcessor.

ApplicationContext :-

- BeanFactory++
- Support Spring AOP features
- Support I18n capabilities
- WebApplicationContext for web applications etc
- Pre instantiation of the singleton scope beans while container is created. Its eager loading. If prototype scope bean is dependent to singleton then that prototype scope bean will also be pre instantiated to support pre instantiation. <bean lazy-init="true">
- Ability to work with placeholder and properties file
- Support for event handling & publishing
- Ability to stop and start container
- Automatic registration of BeanPostProcessors
- Automatic registration of BeanFactoryPostProcessors
- Support for annotation driven,100%code drivern and boot driven spring programming.

2. What is the dependency Injection?

Identify the dependencies and Inject the dependencies is called dependency

Injection.

Dependency Injection is a fundamental aspect of the Spring framework, through which the Spring container "injects" objects into other objects or "dependencies".

Simply put, this allows for loose coupling of components and moves the responsibility of managing components onto the container.

In DI the underlying server/container/freamework/JVM etc assign/injects dependent class object to target class object dynamically.

Spring IOC Container

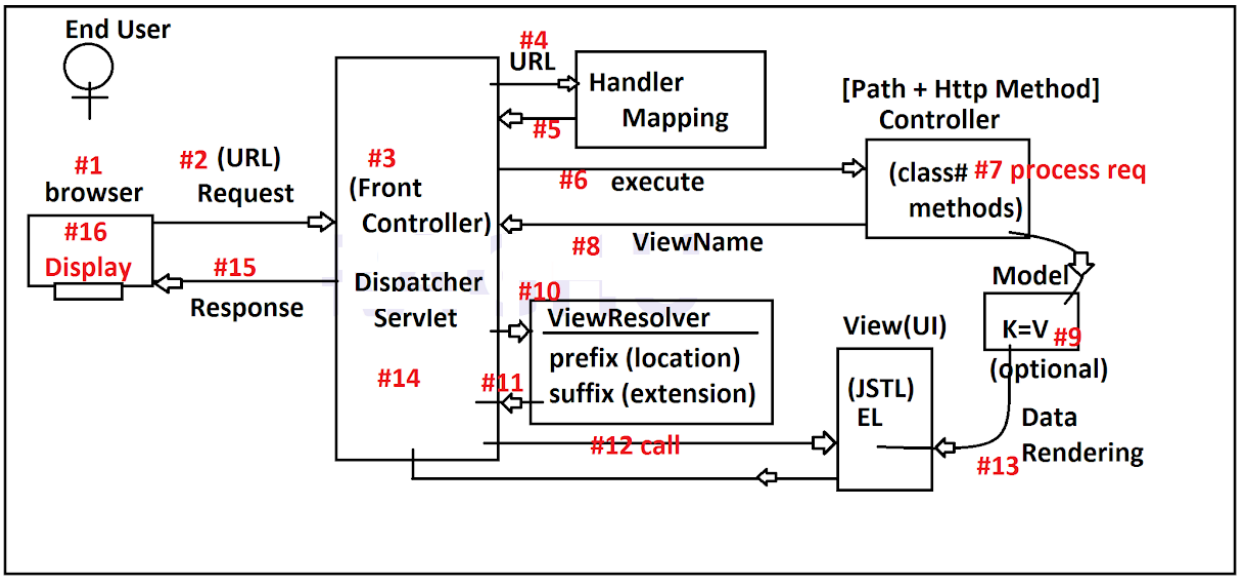
==========

An IoC container is a common characteristic of frameworks that implement IoC.

In the Spring framework, the interface *ApplicationContext* represents the IoC container. The Spring container is responsible for instantiating, configuring and assembling objects known as *beans*, as well as managing their life cycles.

https://www.baeldung.com/inversion-control-and-dependency-injection-in-spring

4.Explain MVC flow.



https://www.youtube.com/watch?v=g2b-NbR48Jo

5. What are the scopes of spring beans?

- **singleton :- This scopes the bean definition to a single instance per Spring IoC container (default).**
- **prototype :- This scopes a single bean definition to have any number of object instances.**
- **request :- This scopes a bean definition to an HTTP request. Only valid in the context of a web-aware Spring ApplicationContext.**
- **session :- This scopes a bean definition to an HTTP session. Only valid in the context of a web-aware Spring ApplicationContext.**
- **global-session :- This scopes a bean definition to a global HTTP session. Only valid in the context of a web-aware Spring ApplicationContext.**

https://www.baeldung.com/spring-bean-scopes

6. What are stereotype annotations? What is the difference between them?

@Component,@Controller,@Service,@Repository

@Component :- Create Object to our class

@Controller :- Create Object + HTTP Protocol (Web Application)

@Service :- Create Object + Calculations / Operations + Transaction Management ….etc

@Repository :- Create Object + DB Operations (Insert, Update, Delete)

Annotation	Meaning
@Component	generic stereotype for any Spring-managed component
@Repository	stereotype for persistence layer
@Service	stereotype for service layer
@Controller	stereotype for presentation layer (spring-mvc)

Enable component scanning

Spring by default does not scan means Spring container does create bean for those classes whose annotated with above for stereotype annotations. So we have to enable component scanning explicity by using "context:component-scan" tag in your applicationContext.xml file. So stereotype annotations will be scanned and configured only when they are scanned by DI container of spring framework.

we use the *@ComponentScan* annotation along with the *@Configuration* annotation to specify the packages that we want to be scanned. *@ComponentScan* without arguments tells Spring to scan the current package and all of its sub-packages.

3.2. *@ComponentScan* with Exclusions

Another way is to use a filter, specifying the pattern for the classes to exclude:

```
@ComponentScan(excludeFilters =
  @ComponentScan.Filter(type=FilterType.REGEX,

pattern="com\\.baeldung\\.componentscan\\.springapp\\.flowers\\..*"))
```

https://www.baeldung.com/spring-component-scanning

7. What are the types of Injection?

Setter Injection, Constructor Injection, Field Injection.

8. When should we go for Setter Injection, Constructor Injection, Field Injection?

9. Explain the wiring?

```
class Hello{
        A aobj;
        B bobj;
        Hello(A aobj){....}
        Void setBobj(...){..}
}
```

Wiring is the process of injecting the dependencies of the bean.

Wiring can be done in two ways :-

1.Explicit Wiring

```
<bean id="h" class="com.jit.spring.Hello">
        <constructor-arg ref="aobj"/>
        <property name="bobj" ref="bobj"/>
</bean>
```

2.Autowiring

```
<bean id="hello" class="com.jit.spring.Hello" autowire=".....">
```

Possible values for autowire :-

- byName
- byType
- constructor

10. Explain @Autowired.

It is used for Annotation based autowiring or field injection. By default it uses byType. When we use @Autowired and if one bean is found then it use byType, If more bean found then it uses byName.

@Autowired + @Qualifier("...") is equivalent to byName.

11. What is the difference between @Autowired , @Resource, @Inject?

Performance wise all are same, their is no difference.

@Autowired :- It is the part of spring framework (Spring 2.0)

```
Class Hello{

        @Autowired
        A aobj;

        @Autowired
        @Qualifier("bo2")
        B bobj;
}
```

@Resource :- It is provided by Java Vendor (javaee.jar) JAVAEE5

```
Class Hello{

        @Resource
        A aobj;

        @Resource(name="bo2")
        B bobj;
}
```

@Inject :- It is provided by Java Vendor (javax.inject-1.jar) JAVAEE6

```
Class Hello{

        @Inject
        A aobj;

        @Inject
        @Qualifier("bo2")
        B bobj;
}
```

https://www.baeldung.com/spring-annotations-resource-inject-autowire

12. Explain @PostConstruct and @Predestroy.

@PostConstruct :- After creating object you want to do something then use this.

Spring calls methods annotated with *@PostConstruct* only once, just after the initialization of bean properties. Keep in mind that these methods will run even if there is nothing to initialize.

The method annotated with *@PostConstruct* can have any access level but it can't be static.

One example usage of *@PostConstruct* is populating a database. During development, for instance, we might want to create some default users:

@PreDestroy :- Clean the resource before destroying the instance.

A method annotated with *@PreDestroy* runs only once, just before Spring removes our bean from the application context.

Same as with *@PostConstruct*, the methods annotated with *@PreDestroy* can have any access level but can't be static.

The purpose of this method should be to release resources or perform any other cleanup tasks before the bean gets destroyed, for example closing a database connection.

```
class Hello{
        A aobj;
        B bobj;

        @PostConstruct
        Void init1(){
                //Check whether dependencies are injected or not, if not injected
                Then inject here.
        }

        @PreDestroy
        Void cleanup(){
                //Resource released
        }
}
```

13. Explain Spring container.

There are two types of container:-

- BeanFactory
- ApplicationContext

BeanFactory :-

You can create the BeanFactory container as follows :-

a)Resource res = new ClassPathResource("jitendra.xml");
b)Resource res = new FileSystemResource("D:/D1/spring/labs/jitendra.xml");
BeanFactory factory = new XmlBeanFactory(res);

Life of Bean in the BeanFactory container :-

1. Container loads the Bean class into memory.
2. Container creates the Bean instance by using corresponding constructor (Constructor Injection)
3. Bean dependencies will be injected with the following ways
   a)XML based explicit wiring
   b)XML based autowiring
4. When bean class is implementing BeanNameAware interface then setBeanName() method will be called by constructor
5. When bean class is implementing BeanFactoryAware interface then

setBeanFactory() method will be called by container.
6.When bean class is implementing InitializingBean interface then afterPropertiesSet() method will be called by container.
7.When bean definition contains init-method attribute then that specified method will be called.
8.Fully initialized Bean instance will be ready to use in the BeanFactory container.

*At container shutdown time, it will destroy all the bean instances.
*When container is destroying one bean instance, it will do the following task..

- When bean class in implementing DisposableBean interface then destroy() Method will be called by the container.
- When bean definition contains destroy-method attribute then that specified Method will be called

Note :- BeanFactory container doesn't support annotation, BeanPostProcessor.
Note :- ApplicationContext internally uses BeanFactory.

ApplicationContext :-

ApplicationContext interface has three concrete implementations :-

1. ClassPathXmlApplicationContext
2. FileSystemXmlApplicationContext
3. XmlWebApplicationContext

You can create the ApplicationContext container as follows :-

1. ApplicationContext ctx = new ClassPathXmlApplicationContext("jitendra.xml");
2. ApplicationContext ctx = new FileSystemXmlApplicationContext("D:/D1/Spring/jitendra.xml");

Life of bean in the ApplicationContext container :-

1. Container loads the Bean class into memory.
2. Container creates the Bean instance by using corresponding constructor (Constructor Injection)
3. Bean dependencies will be injected with the following ways
   a)Annotation based autowiring(field injection)
   b)XML based explicit wiring (setter injection)
   c)XML based autowiring (setter injection)
4. When bean class is implementing BeanNameAware interface then setBeanName() method will be called by constructor
5. When bean class is implementing BeanFactoryAware interface then setBeanFactory() method will be called by container.
6. When bean class is implementing ApplicationContextAware interface then setApplicationContext() method will be called by the container.

7. **When BeanPostProcessor is registered then postProcessBeforeInitialization() method will be called by container.**
8. **When any method is found with @PostConstruct annotation then that method will be called.**
9. **When bean class is implementing InitializingBean interface then afterPropertiesSet() method will be called by the container.**
10. **When bean definition contains init-method attribute then that specified method will be called.**
11. **When BeanPostProcessor is registered then postProcessAfterInitialization() Method will be called by the container.**
12. **Fully initialized bean instance will be ready to use in the ApplicationContext container.**

***At container shutdown time, it will destroy all the bean instances.**
***When container is destroying one bean instance, it will do the following task..**

- **When any method found with @Predestroy annotation then that method will be called.**
- **When bean class is implementing DisposableBean interface then destroy() method will be called by container.**
- **When bean definition contains destroy-method attribute then that specified Method will be called**

14. Explain Bean definition.

```
<bean id="hello"
      name="hello"
      class="com.jlcindia.spring.Hello"
      lazy-init="false"
      scope="singleton"
      autowire="byName"
      init-method="myInit"
      destroy-method="myCleanup"
      abstract="true"
      parent="x"
      ....
/>
```

15. What is the difference between @Bean and @Component.
REF :-
https://springbootdev.com/2017/08/02/spring-framework-component-vs-bean-annotations

https://www.danvega.dev/blog/2017/05/17/spring-component-vs-bean/
/

16. What is the difference between @Qualifier and @Primary.

REF :-
https://newbedev.com/difference-between-primary-vs-autowired-with-qualifier-annotations

1. @Primary annotation in Spring

When there are multiple beans available of same type in Spring container, all of them are qualified to be autowired to single-valued dependency. That causes ambiguity and leads to throw an exception by framework. @Primary indicates that a bean should be given preference when multiple candidates are qualified to autowire a single-valued dependency.

There should be only one @Primary bean among same type of beans.

1. @Qualifier annotation in Spring

We use @Qualifier in Spring to autowire a specific bean among same type of beans, where as @Primary is used to give high preference to the specific bean among multiple beans of same type to inject to a bean.

17. What is the difference between @SpringBootApplication and @Configuration?

***@EnableAutoConfiguration* annotation enables Spring Boot to auto-configure the application context. Therefore, it automatically creates and registers beans based on both the included jar files in the classpath and the beans defined by us.**

The package of the class declaring the *@EnableAutoConfiguration* annotation is considered as the default. Therefore, we should always apply the *@EnableAutoConfiguration* annotation in the root package so that every sub-packages and class can be examined:

We can use *exclude* to disable a list of classes that we do not want to be auto-configured:

```
@Configuration
```

```
@EnableAutoConfiguration(exclude={JdbcTemplateAutoConfiguration.class})
public class EmployeeApplication {
    public static void main(String[] args) {
        ApplicationContext context =
SpringApplication.run(EmployeeApplication.class, args);
        // ...
    }
}
```

@SpringBootApplication = @SpringBootConfiguration + @EnableAutoConfiguration + @ComponentScan

https://www.baeldung.com/spring-componentscan-vs-enableautoconfiguration

18. What is spring-starter-parent?

The *spring-boot-starter-parent* project is a special starter project – that provides default configurations for our application and a complete dependency tree to quickly build our *Spring Boot* project.

For example, if we're building a web project, we can add *spring-boot-starter-web* directly, and we don't need to specify the version:

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

If we want to change the version of any dependency that we want to pull from the starter parent, we can add the dependency in the dependency tag and directly configure its property:

```
<properties>
    <junit.version>4.11</junit.version>
</properties>
```

https://www.baeldung.com/spring-boot-starter-parent

https://developer.okta.com/blog/2018/07/30/10-ways-to-secure-spring-boot

19. **How you implemented security in your project?**

https://developer.okta.com/blog/2018/07/30/10-ways-to-secure-spring-boot

To force HTTPS in your Spring Boot app, you can extend `WebSecurityConfigurerAdapter` and require a secure connection.

```
@Configuration
public class SecurityConfiguration extends
WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) throws Exception
{
        http.requiresChannel().anyRequest().requiresSecure();
    }

}
```

20. **Can you explain the Architecture of JWT token? What is the expiration time of JWT In your project.**

    Default 1200 seconds or 20 minutes.

21. **After validation if JWT is invalid/expired how do you handle this?**

22. **What is the difference between @Controller and @RestController?**
    https://www.javacodegeeks.com/2017/08/difference-restcontroller-controller-annotation-spring-mvc-rest.html
    ***@RequestBody* annotation maps the *HttpRequest* body to a transfer or domain object, enabling automatic deserialization of the inbound *HttpRequest* body onto a Java object.**

The *@ResponseBody* annotation tells a controller that the object returned is automatically serialized into JSON and passed back into the *HttpResponse* object.

https://www.baeldung.com/spring-controller-vs-restcontroller
https://www.baeldung.com/spring-request-response-body

23. What is cyclic dependency injection? How to resolve this?
24. How do you validate requests in spring boot?

https://www.baeldung.com/spring-mvc-custom-validator

https://www.javadevjournal.com/spring/spring-rest-api-validation/

25. When to use Constructor injection and when to use Setter dependency Injection in Spring.
We need the assurance from the Inversion of control (IoC) container that, before using any bean, the injection of necessary beans must be done.

In setter injection strategy, we trust the Inversion of control (IoC) container that it will first create the bean first but will do the injection right before using the bean using the setter methods. And the injection is done according to your configuration. If you somehow misses to specify any beans to inject in the configuration, the injection will not be done for those beans and your dependent bean will not function accordingly when it will be in use!

But in constructor injection strategy, container imposes (or must impose) to provide the dependencies properly while constructing the bean.
This was addressed as "container-agnostic manner", as we are required to provide dependencies while creating the bean, thus making the visibility of dependency, independent of any IoC container.
26. What are the Spring bean tag properties (attributes).
In spring the beans are managed by Spring IoC container, these are backbone of the application.
You can instantiate and manage them in your application using configurations.

In xml based spring bean configurations, using <bean> tag, you can manage them. Here we have given complete list of bean tag properties:

name / id:

This attribute specifies the bean unique identifier. In XML based configuration metadata, you use the id and/or name attributes to specify the bean identifier.

class:

This attribute is mandatory and specify the bean class to be used to create the bean. You should specify fully qualified class name. Include package name.

scope:

This attribute specifies the scope of the objects created from a particular bean definition. The scope values can be prototype, singleton, request, session, and global session.

constructor-arg:

This is used to inject the dependencies through bean constructor.

properties:

This attribute is used to inject the dependencies through setter method.

autowiring mode:

This is used to inject the dependencies.

lazy-init (lazy-initialization mode):

A lazy-initialized bean tells the IoC container to create a bean instance when it is first requested, rather than at startup.

init-method (initialization method):

A callback to be called just after all necessary properties on the bean have been set by the container. This is part of bean lifecycle.

destroy-method (destruction method):

A callback to be used when the container containing the bean is destroyed. This is part of bean lifecycle.

27. Difference between Spring boot and Spring.
28.Difference between Webservices and Microservices.

==========

HIBERNATE INTERVIEW QUESTIONS

==========

1. **What are the main objects of Hibernate?**
   - **Configuration**
   - **SessionFactory**
   - **Session**
   - **Transaction**
   - **Query**

2. **Why do we use Hibernate over JDBC?**
   1. In case of JDBC we need to do all resource management manually like close the connection, Statement, Resultset here in Hibernate Resource management will done by ORM itself.
   2. In case of JDBC we need to do Exception Mapping at the time of closing statement but in Hibernate it's not required.
   3. Caching mechanism is not available in JDBC, which is available in Hibernate.
   4. JDBC is Database Dependent where Hibernate is Database independent.
   5. Connection Reusability is not there in JDBC.

3. **What is the difference between Session ,SessionFactory and Configuration?**
   Configuration
   ==========
   - **By creating object for this concrete class, we activate the Hibernate framework by collecting Hibernate jars/libraries from the classpath or build paths.. We should add <hibernate_home>\lib\required folder jars**
   - **Takes hibernate configuration file name and location as input value and also takes hibernate mapping file names and location through hibernate configuration file name...**
   - **This class is given based on the Builder Design Pattern.**
   - **USING THIS OBJECT WE CAN CREATE Hibernate SessionFactory object .**
   - **If no hibernate cfg file name is supplied then it will take the hibernate.cfg.xml file from classpath (src folder of eclipse project) as default hibernate cfg file name.**

     ```
     // Activate Hibernate framework
     Configuration cfg= new Configuration();
     // supply HB cfg, mapping file name
     cfg.configure(); // Takes hibernate.cfg.xml of classpath as cfg file name
     cfg.configure("com/nt/cfgs/mycfg.xml"); // Takes given mycfg.xml
     ```

as hibernate cfg file name

SessionFactory

==========

- Designed based on Factory Design Pattern providing abstraction towards Session object creation.
- The configuration object internally readas the entries of configuration file, mapping files and creates different services like jdbc connection pooling, caches, dialect, Transaction Manager, O-R-Mapping metadata, and etc.. and uses them in the creation of SessionFactory object step by step by defining process. So we say configuration is given based on the builder design pattern.

4. What is first level cache and second level cache. How to implement Second Level Cache?

First Level Cache is :- Session Cache
Second Level Cache is :- SessionFactory Cache
Steps For Second Level Cache :-
1. Add the dependency :-

```
<dependency>
   <groupId>org.hibernate</groupId>
   <artifactId>hibernate-ehcache</artifactId>
   <version>5.2.2.Final</version>
</dependency>
```

2. Enable the cache

```
hibernate.cache.use_second_level_cache=true
hibernate.cache.region.factory_class=org.hibernate.cache.ehcache.EhCacheRegionFactory
```

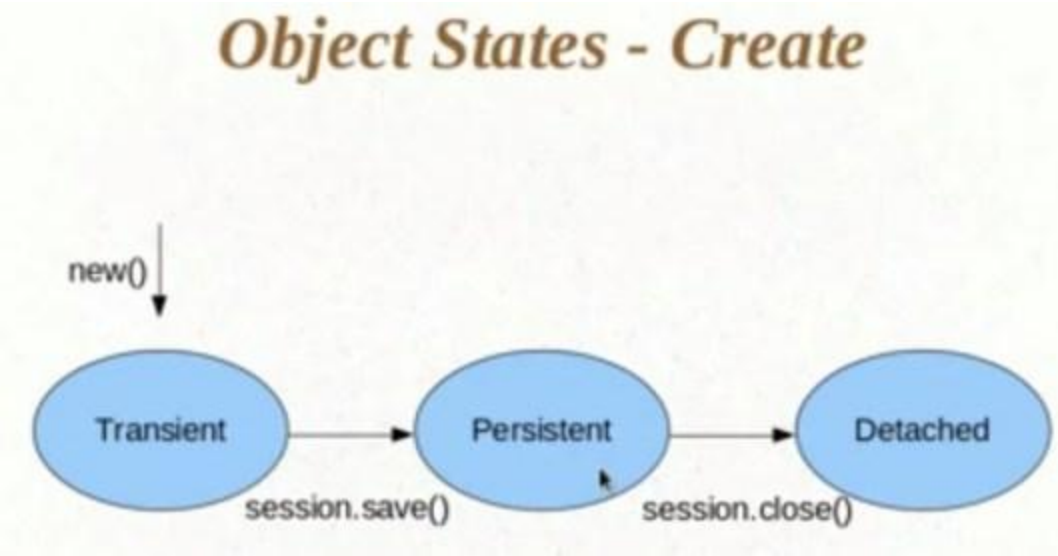
3.Make an Entity Cacheable

```
@Entity
@Cacheable
@org.hibernate.annotations.Cache(usage =
CacheConcurrencyStrategy.READ_WRITE)
public class Foo { }
```

5. Explain the Inheritance Mapping in Hibernate?
6. What is @Modifying?

The *@Modifying* annotation is used to enhance the *@Query* annotation to execute not only *SELECT* queries but also *INSERT*, *UPDATE*, *DELETE*, and even *DDL* queries.

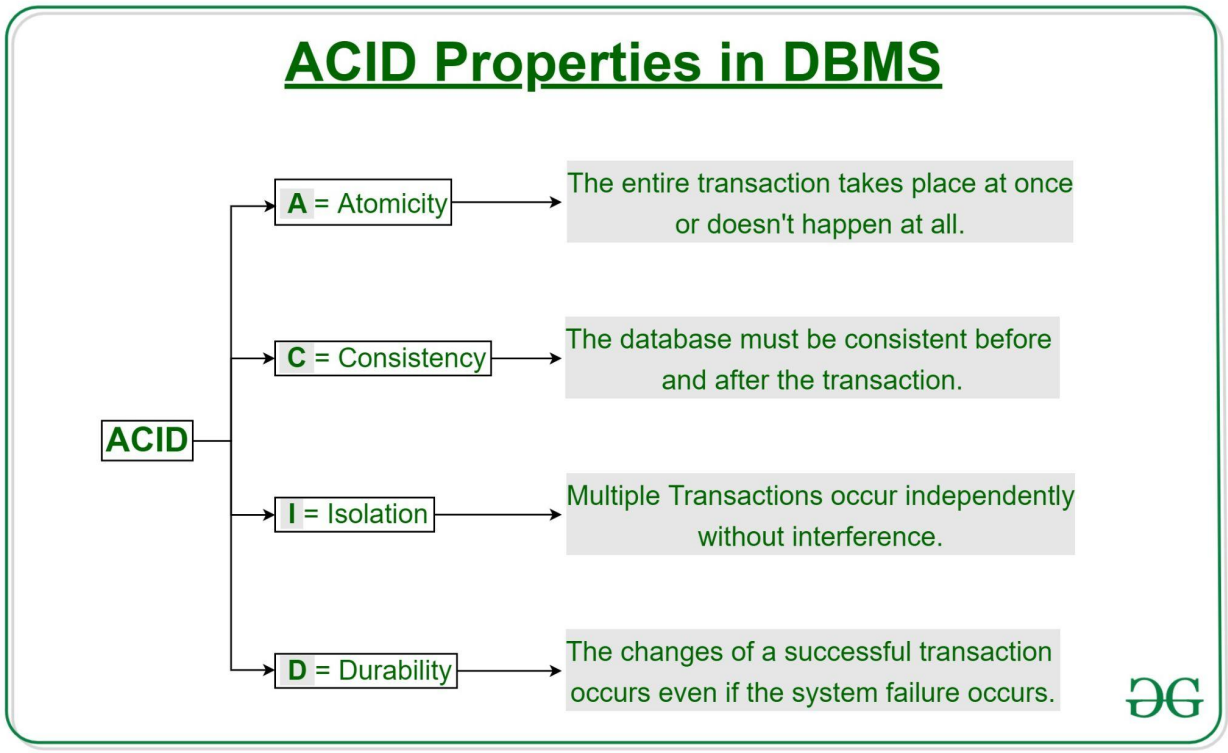
7. What are the different states of an object?



8. **Explain the Association mapping the Hibernate?**
9. **What is the difference between get() and load() method?**
    - **load() method throws exception when primary key is not found.**
    - **get() method return null value when primary key is not found.**

10. **Which DataSource you have used in your project?**
    HikariCp

11. **How to integrate Hibernate with Spring?**
    https://www.javatpoint.com/hibernate-and-spring-integration
12. **What are files required for Hibernate Configuration?**
13. **Explain ACID properties?**



14. **What is the difference between inner join and outer join in SQL?**

    The `INNER JOIN` keyword selects all rows from both tables as long as there is a match between the columns. If there are records in the "Orders" table that do not have matches in "Customers", these orders will not be shown!

    https://www.w3schools.com/sql/sql_join_inner.asp

15. **Query to delete duplicate rows?**

```
delete from TABLE where
Col1 in (select id from table group by id having (count (col1)>1))
```

16. **Find second height salary?**

```
Select * from employee
group by salary
Order by salary desc  limit 1,1;
```

    https://www.geeksforgeeks.org/sql-query-to-find-second-largest-salary/#:~:text=select%20*from%20employee%20group%20by,MAX(salary)%20FROM%20employee)%3B

17. **What is the difference between Left outer join and full outer join?**

In some databases LEFT JOIN is called LEFT OUTER JOIN.
The `LEFT JOIN` keyword returns all records from the left table (table1), and the matching records from the right table (table2). The result is 0 records from the right side, if there is no match.
Note: The `LEFT JOIN` keyword returns all records from the left table (Customers), even if there are no matches in the right table (Orders).

`FULL OUTER JOIN` and `FULL JOIN` are the same.
The `FULL OUTER JOIN` keyword returns all matching records from both tables whether the other table matches or not. So, if there are rows in "Customers" that do not have matches in "Orders", or if there are rows in "Orders" that do not have matches in "Customers", those rows will be listed as well.

https://www.w3schools.com/sql/sql_join_full.asp

18. What is cascade?
    https://www.baeldung.com/jpa-cascade-types

    Cascading is the way to achieve this. When we perform some action on the target entity, the same action will be applied to the associated entity.

19. How to mock the DB?

20. Show one query for Left outer join
    Select COLUMN_NAME from table1
    LEFT JOIN table2 on table1.COLUMN_NAME=table2.COLUMN_NAME
21. What is indexing?
    Indexes are special lookup tables that the database search engine can use to speed up data retrieval. Simply put, an index is a pointer to data in a table. An index in a database is very similar to an index in the back of a book.

    For example, if you want to reference all pages in a book that discusses a certain topic, you first refer to the index, which lists all the topics alphabetically and are then referred to one or more specific page numbers.

An index helps to speed up SELECT queries and WHERE clauses,
but it slows down data input, with the UPDATE and the INSERT statements.
Indexes can be created or dropped with no effect on the data.

Syntax :- CREATE UNIQUE INDEX index_name
on table_name (column_name);

Syntax :- CREATE INDEX index_name
on table_name (column1, column2);

Syntax :- DROP INDEX index_name;

22. Explain Different types of Joins .

There are different types of joins available in SQL −

INNER JOIN − returns rows when there is a match in both tables.

LEFT JOIN − returns all rows from the left table, even if there are no matches in the right table.

RIGHT JOIN − returns all rows from the right table, even if there are no matches in the left table.

FULL JOIN − returns rows when there is a match in one of the tables.

SELF JOIN − is used to join a table to itself as if the table were two tables, temporarily renaming at least one table in the SQL statement.

https://www.w3schools.com/sql/sql_join_self.asp

CARTESIAN JOIN − returns the Cartesian product of the sets of records from the two or more joined tables

23. What is SQL UNION?

The `UNION` operator is used to combine the result-set of two or more `SELECT` statements.

The SQL UNION clause/operator is used to combine the results of two or more SELECT statements without returning any duplicate rows.

Note: If some customers or suppliers have the same city, each city will only be listed once, because `UNION` selects only distinct values. Use `UNION ALL` to also select duplicate values!

To use this UNION clause, each SELECT statement must have

The same number of columns selected
The same number of column expressions
The same data type and
Have them in the same order
But they need not have to be in the same length.

Syntax
The basic syntax of a UNION clause is as follows −

```
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]

UNION

SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
```

****UNION ALL :- The UNION ALL operator is used to combine the results of two SELECT statements including duplicate rows.**

24. **What is the window function?**
    https://www.youtube.com/watch?v=TzsrO4zTQj8

Window functions applies aggregate and ranking functions over a particular window (set of rows). OVER clause is used with window functions to define that window. OVER clause does two things :

- **Partitions rows into form set of rows. (PARTITION BY clause is used)**
- **Orders rows within those partitions into a particular order. (ORDER BY clause is used)**

Aggregate Window Function :
Various aggregate functions such as SUM(), COUNT(), AVERAGE(), MAX(), MIN() applied over a particular window (set of rows) are called aggregate window functions.

https://www.geeksforgeeks.org/window-functions-in-sql/

Compute average salary and display it against every employee:-
SELECT Name, Gender, Salary,
AVG(Salary) OVER(ORDER BY Salary ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS Average
FROM Employees

25. **What is PIVOT and UNPIVOT.**
Pivot is a sql server operator that can be used to turn unique values from one column, into multiple columns in the output, there by effectively rotating a table.

https://www.youtube.com/watch?v=h3BtudZehuo

26. **Is Session and Session Factory is thread safe?**
27. **What is view and trigger?**
In SQL, a view is a virtual table based on the result-set of an SQL statement.
A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Trigger: A trigger is a stored procedure in database which automatically invokes whenever a special event in the database occurs. For example, a trigger can be invoked when a row is inserted into a specified table or when certain table columns are being updated.

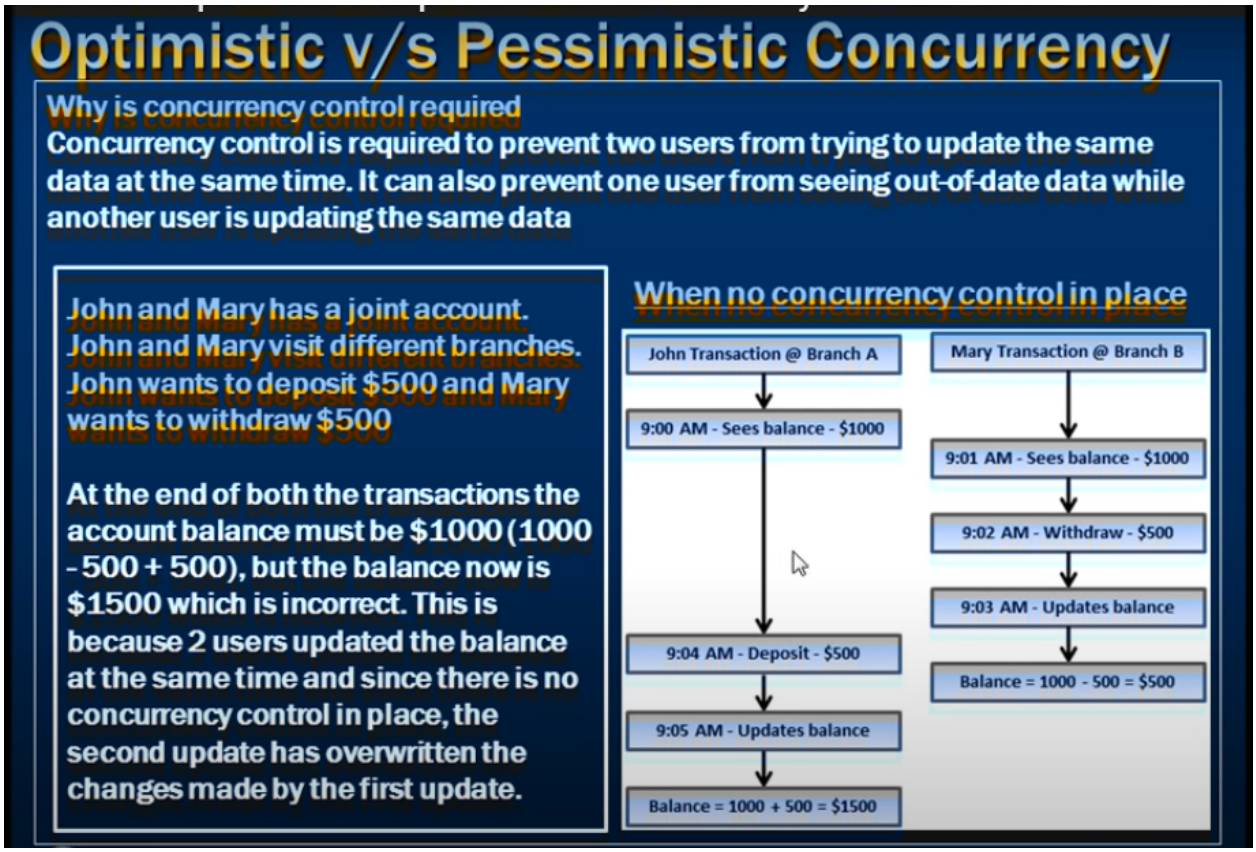
create trigger [trigger_name]

[before | after]
{insert | update | delete}
on [table_name]
[for each row]
[trigger_body]

https://www.geeksforgeeks.org/sql-trigger-student-database/

28. How to mock the DB?
29. Two table doesn't have any common key. Is there any possibility to join these two table in hibernate.
30. How to optimize the Query.
31. Have you written Stored procedure and functions?
32. Why is concurrency control required?

    Concurrency control is required to prevent two users from trying to update the same data at the same time. It can also prevent one user from seeing out-of-date data while other user is updating the same data.



33. What is the difference between optimistic and pessimistic locking.

    Reference :- https://www.youtube.com/watch?v=jQarp7R2hhU
    https://www.youtube.com/watch?v=79AGhjJiDt8

34. **What is difference between procedure and function and PLSQL?**

Stored Procedures

Stored Procedures are pre-compiled objects which are compiled for the first time and its compiled format is saved, which executes (compiled code) whenever it is called. For more about a stored procedure, please refer to the article Different types of Stored Procedure.

Functions

A function is compiled and executed every time whenever it is called. A function must return a value and cannot modify the data received as parameters. For more about functions, please refer to the article Different types of Functions.

- **The function must return a value but in Stored Procedure it is optional. Even a procedure can return zero or n values.**
- **Functions can have only input parameters for it whereas Procedures can have input or output parameters.**
- **Functions can be called from Procedure whereas Procedures cannot be called from a Function.**

35. **Difference between drop and truncate**

The DROP command is used to remove table definition and its contents. Whereas the TRUNCATE command is used to delete all the rows from the table. ... DROP is a DDL(Data Definition Language) command. Whereas the TRUNCATE is also a DDL(Data Definition Language) command

==========

RestAPI INTERVIEW QUESTIONS

==========

1. **Does the HTTP method return anything?**
2. **How do you handle exceptions in your project?**
3. **What are the http methods you are using in your project?**

4. **Can you list down some of the HTTP response status which you return as a part of response in your project?**
5. ***1xx informational response* – the request was received, continuing process**
6. ***2xx successful* – the request was successfully received, understood, and accepted**
7. ***3xx redirection* – further action needs to be taken in order to complete the request**
8. ***4xx client error* – the request contains bad syntax or cannot be fulfilled**
9. ***5xx server error* – the server failed to fulfil an apparently valid request**

10. **Explain Idempotent methods?**

    You can divide HTTP methods into two main categories safe and idempotent. Safe methods are HTTP methods that do not modify the resource like a GET request is safe because it doesn't modify the resource you are requesting like data of a Book.

    Another safe HTTP method is `HEAD`, which doesn't change the resource representation on the Server, but all other HTTP methods like `POST`, `PUT`, or `DELETE` are non-safe.

    Coming to idempotent methods, they are HTTP methods that can be called multiple times and they will produce the same result. They are considered the safe option to update a resource on the Server.

    Some examples of idempotent HTTP methods are GET, PUT, and PATCH. No matter how many times you call them, they will produce the same result with the same URI.

    PUT method is call when you have to modify a single resource, which is already a part of resource collection. POST method is call when you have to add a child resource under resources collection

    https://javarevisited.blogspot.com/2016/05/what-are-idempotent-and-safe-methods-of-HTTP-and-REST.html#axzz77JagUCAp

11. **Which method is safe and which method is unsafe?**
12. **How to implement cache in Rest API?**

    All communication done via REST API uses only HTTP request.
13. **Does RestAPi support only HTTP protocol?**
14. **If in the User object you are getting a profile picture also. How will you handle this at controller#method level. (Use MultipartFile concept)**

    Reference :- https://www.youtube.com/watch?v=dYfCxyA9Xy8
15. **What is the difference between REST and SOAP. Why do we choose REST over SOAP?**

No.	SOAP	REST
1)	SOAP is a **protocol**.	REST is an **architectural style**.
2)	SOAP stands for **Simple Object Access Protocol**.	REST stands for **REpresentational State Transfer**.
3)	SOAP **can't use REST** because it is a protocol.	REST **can use SOAP** web services because it is a concept and can use any protocol like HTTP, SOAP.
4)	SOAP **uses services interfaces to expose the business logic**.	REST **uses URI to expose business logic**.
5)	**JAX-WS** is the java API for SOAP web services.	**JAX-RS** is the java API for RESTful web services.
6)	SOAP **defines standards** to be strictly followed.	REST does not define too much standards like SOAP.
7)	SOAP **requires more bandwidth** and resource than REST.	REST **requires less bandwidth** and resource than SOAP.
8)	SOAP **defines its own security**.	RESTful web services **inherits security measures** from the underlying transport.
9)	SOAP **permits XML** data format only.	REST **permits different** data format such as Plain text, HTML, XML, JSON etc.
10)	SOAP is **less preferred** than REST.	REST **more preferred** than SOAP.

16. **How to handle file using post method in swagger?**
17. **What is 401 status code ?**

    Unauthorized
18. **What is difference between Webservice and MicroService?**

Microservices - also known as the microservice architecture - is an architectural style that structures an application as a collection of services that are

Highly maintainable and testable

Loosely coupled

Independently deployable

Organized around business capabilities

Owned by a small team

The microservice architecture enables the rapid, frequent and reliable delivery of large, complex applications. It also enables an organization to evolve its technology stack.

19. What is the difference between jar and war file?

The main difference between JAR and WAR Files is that the JAR files are the files that have Java class files, associated metadata and resources aggregated into a single file to execute a Java application while the WAR files are the files that contain Servlet, JSP, HTML, JavaScript and other files necessary for developing web applications.

WAR stands for Web Application Archive or Web Application Resource. These archive files have the . war extension and are used to package web applications that we can deploy on any Servlet/JSP container.

20. What is web services?

A web service is a collection of open protocols and standards used for exchanging data between applications or systems. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer. This interoperability (e.g., between Java and Python, or Windows and Linux applications) is due to the use of open standards.

to summarize, a complete web service is, therefore, any service that −

- Is available over the Internet or private (intranet) networks
- Uses a standardized XML messaging system
- Is not tied to any one operating system or programming language
- Is self-describing via a common XML grammar
- Is discoverable via a simple find mechanism

21. What is the difference between Http and Https?
22. What is the difference between Rest and Soap?

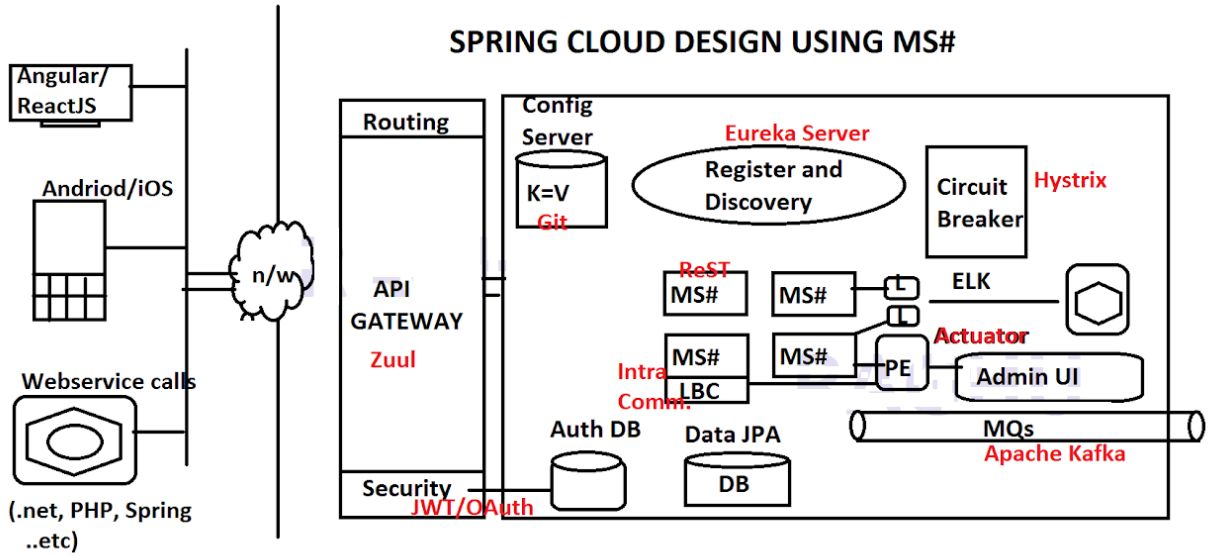
23. What is Option method in rest API

24. What is Purge method in rest API

==========

MICROSERVICES INTERVIEW QUESTIONS

==========



1. **What challenges did you face during migration of Monolithic to Microservices.**
2. **How in Monolithic you handle huge traffic / dynamic traffic.**
3. **How different applications communicate in Microservices.**
4. **Which tool do you use to do load testing?**

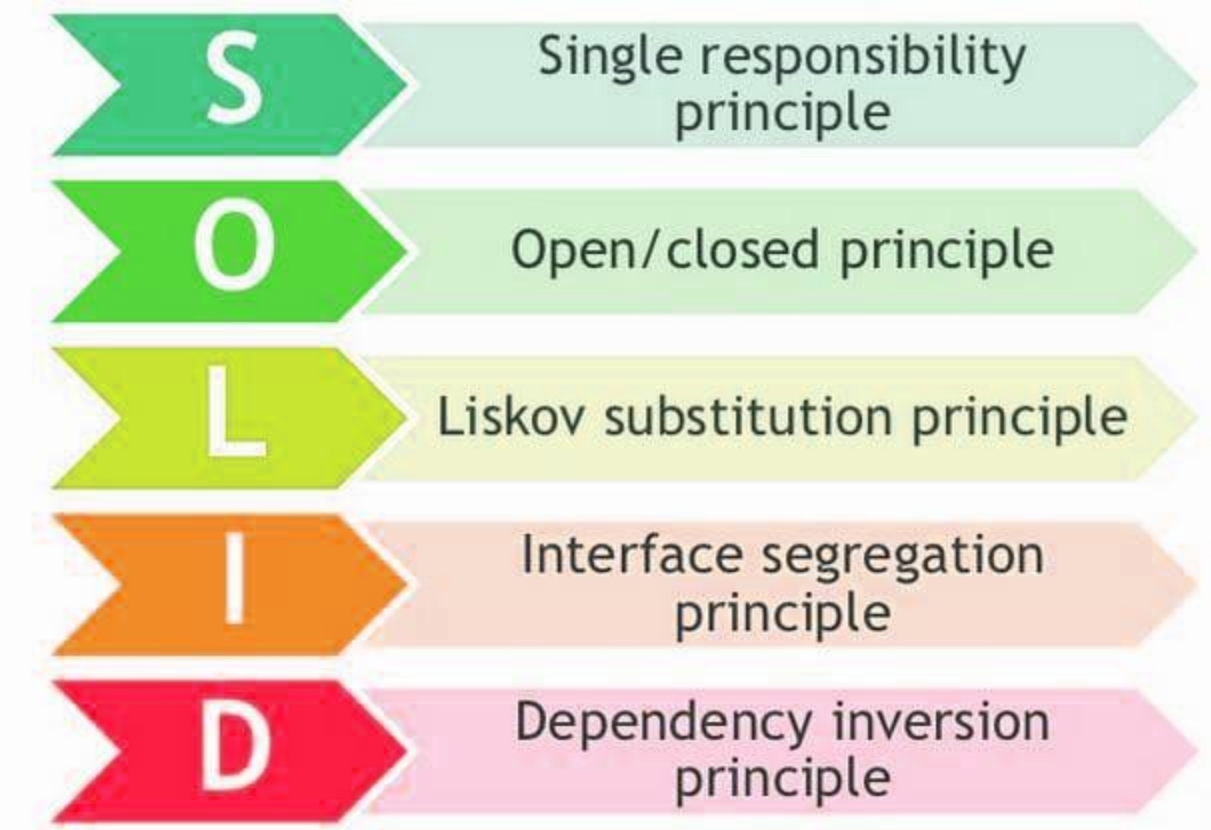
5. **How do you monitor robust applications?**
6. **How you deploy applications in kubernetes.**
7. **How you handle the cyclic dependency of databases in microservices.**
8. **What is offset in kafka?**
   Messages stored in topic and topic have partitions. The index of that partition is called offset.
9. **What are the main components of Spring cloud?**
   - **Register and Discovery Server**
     -Eureka Server
   - **Intra Communication Clients**
     -DiscoveryClient
     -LoadBalancerClient
     -FeignClient
   - **Fault Tolerance/Circuit Breaker**
     -Hystrix with Dashboard
   - **Log Trace and Visualize**
     -ELK (Elastic Search, Logstash, Kibana, Zipkin and Sleuth)
   - **Message Queues**
     -ActiveMQ, Apache kafka
   - **Production Support Endpoints**
     -Actuator with Admin UI
   - **Config Server(common properties)**
     -Git Accounts(GitHub/Gitlab)
   - **API Gateway**
     -Zuul
   - **Security**
     -JWT, OAuth(FB,GOOGLE,..)
10. **What are the clients used for Intra communication?**
    1. Discovery Client -> client.getInstances("----")
    2. Load Balancer Client ->client.choose("-----")
    3.Feign Client

==========

DESIGN PATTERN INTERVIEW QUESTIONS

==========

1. **SOLID Principle.**



Single Responsibility Principle - One class should have one and only one responsibility
Open Close Principle - Software components should be open for extension, but closed for modification
Liskov Substitution - Derived types must be completely substitutable for their base types
Interface Segregation - Clients should not be forced to implement unnecessary methods which they will not use
Dependency Inversion - Depend on abstractions, not on concretions

2. **Adapter Design Pattern.**
3. **Singleton Design Pattern.**
4. **Factory Design Pattern.**
5. **Builder Design Pattern.**
6. **Proxy Design Pattern.**
7. **Observable Design Pattern**
8. **Template Design Pattern**
9. **Give example of 5-6 design patterns being used in Java classes.**

Factory: java.util.Calendar#getInstance()
Abstract Factory: javax.xml.parsers.DocumentBuilderFactory#newInstance(), javax.xml.transform.TransformerFactory#newInstance(), javax.xml.xpath.XPathFactory#newInstance()
Builder: java.lang.StringBuilder#append(), java.lang.StringBuffer#append()
Decorator: All subclasses of Buy & Sell Domain Names.InputStream, OutputStream, Reader and Writer.
Chain of responsibility: javax.servlet.Filter#doFilter()
Iterator: java.util.Iterator
Observer: JMS message listener
Singleton: java.lang.Runtime#getRuntime()
Adapter: java.util.Arrays#asList(), java.io.InputStreamReader(InputStream) (returns a Reader), java.io.OutputStreamWriter(OutputStream) (returns a Writer)

==========

CONDITION BASED INTERVIEW QUESTIONS

==========

1.Have you ever faced memory leakage in your project? How did you overcome that?

2.How to work with multiple databases at a time?g

3.Tell me about yourself?

4.Explain your project?

5.What is the hardest challenge you have faced in your life?

6.What difficulties did you face in your project?

- Understanding the requirements
- Requirements changes in middle
- Some times test data is not available
- Code sometimes works fine in the dev/test environment. But same code fails in CERT / PROD environment

7.What are your top three weaknesses and strengths?

Strengths :-

- Adopt new technology very quickly
- Deliver neat and clean code
- Always punctual to learn something new

Weakness :-

- If I do something new , I get nervous

8.What do you know about the company?

9.Why are you looking for changes?

- To learn something new
- To see the diversity
- To grow financially

10. What is the most complex code you have written?
11.Have to written/used most complex algorithm in your project. If yes please explain scenario?
12. Have you worked in multithreaded environment. If yes please explain scenario?

==========

GIT INTERVIEW QUESTIONS

==========

1. **What is the difference between git fetch and git pull?**

==========

PROGRAMMING INTERVIEW QUESTIONS

==========

1. **Given an integer array 2,6,3,6,2,4,6,5,8,6,5,5,4,1**
   O/p 1:- 5->4->3….1
   O/p 2:- 4->5->3….1
   Using java 8 stream concept filter the element from above array whose count is more than 3.

2. **Write stream function to get employee's salary based on age if there age is more than 45 give increment in salary.**

```
List<Employee> list = EmployeeList.getAllEmployee();
list.stream().map((emp) -> {
if (emp.getAge()>30) {
emp.setSalary(emp.getSalary()+1000);
return emp;
} else {
return emp;
}
}).collect(Collectors.tgoList());
```

3. **Find the pair of number in an array which is nearest to given number.**
   https://www.geeksforgeeks.org/given-sorted-array-number-x-find-pair-array-whose-sum-closest-x/

   int arr[] = { 10, 22, 28, 29, 30, 40 };
   int sum = 54;
   Ex:- {30,20}
4. **Find the pair of number whose sum is equal to given number.**
   int arr[] = { 10, 24, 28, 29, 30, 40 };
   int sum = 54;

   Ex:- {30,24}
5. **Find the second highest number in an array.**
6. **Convert string to array of characters.**
   https://www.geeksforgeeks.org/convert-a-string-to-character-array-in-java/

7. **Sort an string array.**
8. **Find maximum repeating character/number/String literal in an array.**
9. **Sort the array without using two loops.**
10. **Reverse a String.**
11. **Write Stream function to find the string ending with "at" like retreat,habitat,preheat,acrobat,wildcat**
12. **Write program to check number is Anagram or Not?**
13. **WAP to check number is Palindrom or not.**
14. **Write program to check string is Palindrom or not**
15. **WAP to to sort Employee based on first name using Comparator.**

```
class  Employee
{
Private String firstName;
Private String lastName;
}
```

16. **Sort a String**
17. **Find prime number between given**
18. **Check if the given number is autobiological or not? For example 1210**
19. **https://www.geeksforgeeks.org/minimum-number-of-jumps-to-reach-end-of-a-given-array/?ref=leftbar-rightbar**
20. **Reverse order of element in arraylist using stream.**

```
public static <T> Collector<T, ?, Stream<T> > reverseStream()
    {
        return Collectors
            .collectingAndThen(Collectors.toList(),
                               list -> {
                                   Collections.reverse(list);
                                   return list.stream();
                               });
    }

    // Driver code
    public static void main(String[] args)
    {

        // Get the parallel stream
        List<Integer> lists = Arrays.asList(11, 22, 33, 44);
        Stream<Integer> stream = lists.parallelStream();

        // Reverse and print the elements
        stream.collect(reverseStream())
            .forEach(System.out::println);
    }
```

21. Sort employee map based on name using stream api.

https://www.java67.com/2017/07/how-to-sort-map-by-values-in-java-8.html

```
Map<String, Integer> sortedByValue = ItemToPrice.entrySet() .stream()
.sorted(Map.Entry.<String, Integer>comparingByValue()) .collect(toMap(Map.Entry::getKey,
Map.Entry::getValue, (e1, e2) -> e1, LinkedHashMap::new));
```

22. Use of joining Method in steam

https://www.geeksforgeeks.org/java-8-streams-collectors-joining-method-with-examples/

23. Longest Substring of given string with k distinct character

https://www.geeksforgeeks.org/find-the-longest-substring-with-k-unique-characters-in-a-given-string/

==========

Important Links

==========

http://lahotisolutions.blogspot.com/2018/01/top-java-interview-questions-2018-for.html