

Java Interview Questions

By

Praveen Oruganti

Blog: <https://praveenorugantitech.blogspot.com>

Facebook Group: <https://www.facebook.com/groups/268426377837151>

Github repo: <https://github.com/praveenoruganti>

Email: praveenorugantitech@gmail.com

Java Principal Architect interview @ Aristocrat

1.What is Apache Kafka? When it is used? What is the use of Producer, Consumer, Partition, Topic, Broker and Zookeeper while using Kafka?

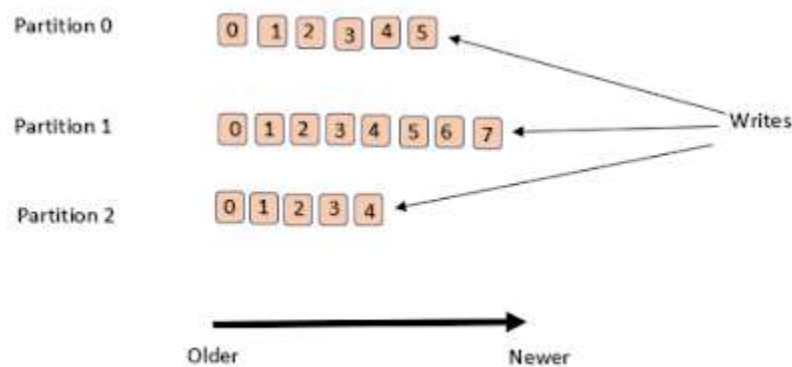
Apache Kafka is a distributed publish-subscribe messaging system.

It is used for real-time streams of data and used to collect big data for real-time analysis.

Topic: A topic is a category or feed or named stream to which records are published. Kafka stores topic in logs file.

Broker: A kafka cluster is a set of servers , each of which is called a broker.

Partition: Topics are broken up into ordered commit logs called partitions. Kafka spreads those log's partitions across multiple servers or disks.



Producer: A producer can be any application who can publish messages to a topic. The producer does not care what partition a specific message is written to and will balance messages over every partition of a topic evenly.

Directing messages to a partition is done using message key and a partitioner. Partitioner will generate a hash of the key and map it to a partition.

Producer publishes a message in the form of key-value pair.

Consumer: A consumer can be any application that subscribes to a topic and consume the messages.

A consumer can subscribe to one or more topics and reads the messages sequentially.

The consumer keeps track of the messages it has consumed by keeping track on the offset of messages.

Zookeeper: This is used for managing and coordinating kafka broker.

2.What are the Kafka features?

Kafka features are listed below:

2 | Praveen Oruganti

Blog: <https://praveenorugantitech.blogspot.com>

Facebook Group: <https://www.facebook.com/groups/268426377837151>

Github repo: <https://github.com/praveenoruganti>

Email : praveenorugantitech@gmail.com

- **High throughput** : Provides support for hundreds of thousands of messages with modest hardware.
- **Scalability** : Highly scalable distributed systems with no downtime.
- **Data loss** : Kafka ensures no data loss once configured properly.
- **Stream Processing** : Kafka can be used along with real time streaming applications like Spark and Storm.
- **Durability** : Provides support for persisting messages to disk.
- **Replication** : Messages can be replicated across clusters, which supports multiple subscribers.

3.What are Kafka components?

Kafka components are:

- **Topic**
- **Partition**
- **Producer**
- **Consumer**
- **Messages**

A topic is a named category or feed name to which records are published. Topics are broken up into ordered commit logs called partitions.

Each message in a partition is assigned a sequential id called an offset.

Data in a topic is retained for a configurable period of time.

Writes to a partition is generally sequential , thereby reducing the number of hard disk seeks.

Reading messages can either be from beginning and also can rewind or skip to any point in partition by giving an offset value.

A partition is an actual storage unit of kafka messages which can be assumed as a kafka message queue. The number of partitions per topic are configurable while creating it.

Messages in a partition are segregated into multiple segments to ease finding a message by its offset.

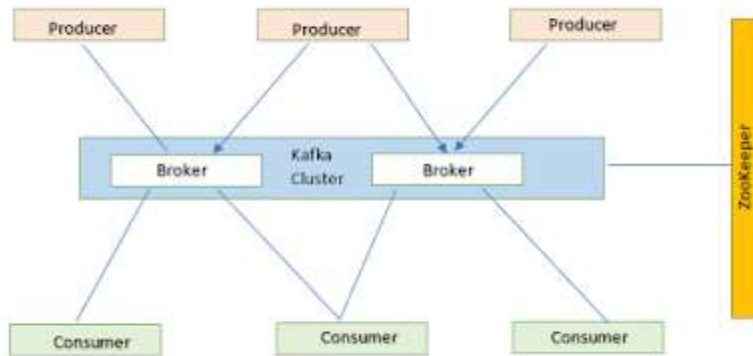
The default size of a segment is very high, i.e. 1GB, which can be configured.

Each segment is composed of the following files:

- **Log**: Messages are stored in this file.
- **Index**: stores message offset and its starting position in the log file.
- **TimeIndex**

4.What is a Kafka cluster?

Kafka cluster is a set of servers and each server is called a broker.

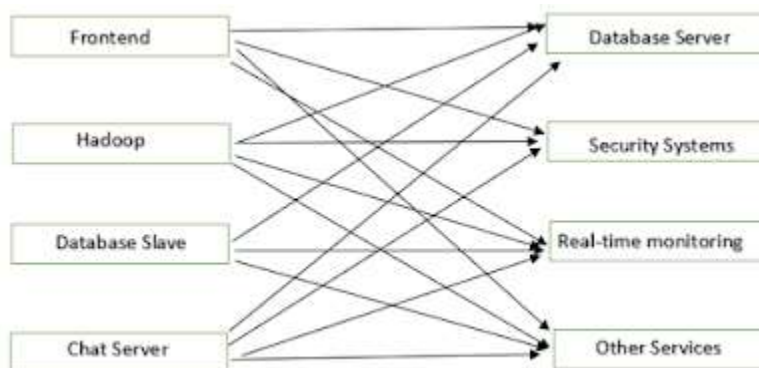


In the above diagram, Kafka cluster is a set of servers which are shown with Broker name. Producers publish messages to topics in these brokers and Consumers subscribe to these topics and consume messages.

Zookeeper is used to manage the coordination among kafka servers.

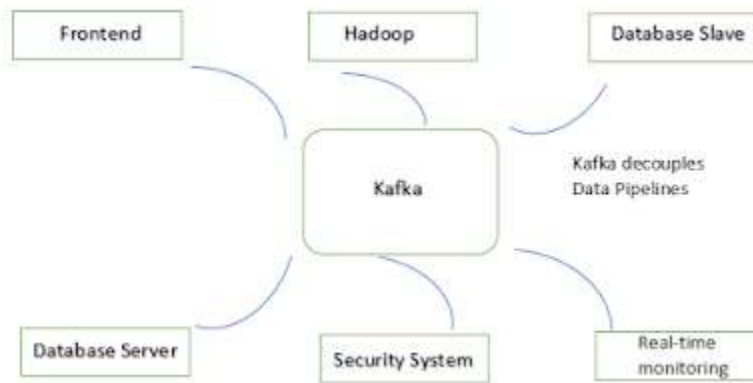
5.What problem does Kafka resolve?

Without any messaging queue implementation, what the communication between client nodes and server nodes look alike is shown below:



There is a large numbers of data pipelines which are used for communication. It is very difficult to update this system or add another node.

If we use Kafka, then the entire system will look like something:



So, all the client servers will send messages to topics in Kafka and all backend servers will consume messages from kafka topics.

6.What is the purpose of using @ServletComponentScan?

We add @ServletComponentScan to enable scanning for @WebFilter, @WebServlet and @WebListener.

It is used on the main SpringBootApplication.java class.

Embedded containers do not support @WebServlet, @WebFilter and @WebListener. That's why spring has introduced @ServletComponentScan annotation to support some dependent jars which use these 3 annotations.

To use @ServletComponentScan, we need to use spring boot with version 1.3.0 or above.

And we also need to add spring-boot-starter-parent and spring-boot-starter-web dependencies.

pom.xml file:

```

<parent>
  <groupId> org.springframework.boot </groupId>
  <artifactId> spring-boot-starter-parent</artifactId>
  <version> 1.5.1.RELEASE </version>
</parent>

<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
    <version>1.5.1.RELEASE</version>
  
```

5 | Praveen Oruganti

Blog: <https://praveenorugantitech.blogspot.com>

Facebook Group: <https://www.facebook.com/groups/268426377837151>

Github repo: <https://github.com/praveenoruganti>

Email : praveenorugantitech@gmail.com

```
</dependency>
</dependencies>
```

7. How does Servlet work and what are the lifecycle methods?

A servlet is a class that handles requests, processes them and reply back with a response.

e.g. we can use a servlet to collect input from user through an HTML form, , query records from a database and create web pages dynamically.

Servlets are under the control of another java application called servlet container. When an application running in a web server receives a request , the server hands the request to the servlet container - which in turn passes it to the target servlet.

Maven dependency for using servlet is given below:

```
<dependency>
  <groupId> javax.servlet</groupId>
  <artifactId> javax.servlet-api</artifactId>
  <version> 3.1.0</version>
</dependency>
```

Lifecycle methods of servlet are described below:

init(): The init method is designed to be called only once. If an instance of servlet does not exist, the web container does the following:

- Loads the servlet class
- Create an instance of the servlet class
- Initializes it by calling the init() method.

The init() method must be completed successfully before the servlet can receive any requests.

The servlet container cannot place the servlet in service if the init() method either throws a ServletException or does not return within a time period defined by the web server.

```
public void init throws ServletException{
    // code here
}
```

service(): This method is only called after the servlet's init() method has completed successfully.

The container calls the service method to handle the requests coming from the client, interprets the HTTP request type (GET, PUT, POST, DELETE etc.) and calls doGet(), doPut(), doPost() and doDelete() methods.

```
public void service(ServletRequest req, ServletResponse response) throws ServletException{
}
```

6 | Praveen Oruganti

Blog: <https://praveenorugantitech.blogspot.com>

Facebook Group: <https://www.facebook.com/groups/268426377837151>

Github repo: <https://github.com/praveenoruganti>

Email : praveenorugantitech@gmail.com

destroy(): It is called by the servlet container to take the servlet out of the service.

This method is only called after all the threads in the service have exited or a time period has passed.

After the container calls this method, it will not call the service method again on the servlet.

```
public void destroy(){  
}
```

8.What application server have you used and what are the benefits of that?

I have used Weblogic application server.

Weblogic server provides various functionalities:

- Weblogic server provides support for access protocols like HTTP, SOAP etc.
- It also provides data access and persistence from database server.
- It also supports SQL transactions for data integrity.
- It also provides security.

So means, when we use Weblogic server, we do not have to care about protocol, security, database transactions, data integrity etc. All these are handled by Weblogic server. We can focus on business logic.

Java Interview @ Polaris

1.

```
public class Quiz23{  
  
    public static void main(String[] args){  
  
        int x = 0;  
        int[] nums = {1,2,3,5};  
  
        for(int i : nums){  
  
            switch(i){  
  
                case 1:  
                    x += i;  
                case 5:  
                    x += i;  
                default:  
                    x += i;  
                case 2:  
                    x += i;  
  
            }  
            System.out.println(x);  
        }  
    }  
}
```

7 | Praveen Oruganti

Blog: <https://praveenorugantitech.blogspot.com>

Facebook Group: <https://www.facebook.com/groups/268426377837151>

Github repo: <https://github.com/praveenoruganti>

Email : praveenorugantitech@gmail.com

What will be the output of above code?

27

Explanation:

For case i = 1: All the cases will run.

For case i = 2: only case 2 will run.

For case i = 3: Default and case 2 will run.

For case i = 5: case , default and case 2 will run.

2.What is the difference between Map and FlatMap?

Map: Transforms the elements into something else. It accepts a function to apply to each element and returns a new Stream of values returned by the passed function.

It takes function object as the parameter e.g.: Function, ToIntFunction, ToIntDoubleFunction, ToLongFunction.

FlatMap: Combination of Map and Flat operations. So, we first apply map operation on each element and then flattens the result.

So, if function used by Map is returning a single value, then map is ok. But, if function used by Map operation is returning a stream of list or stream of stream, then we need to use flatmap to get stream of values.

For example:

If we have a stream of String containing {"12", "34"} and a method getPermutations() which returns a list of permutations of given string.

When we apply getPermutation() into each string of Stream using map , we get something like [{"12","21"}, {"34","43"}], but if we use flatMap, we get a stream of strings e.g.: ["12","21","34","43"].

Another example:

```
List evens = Arrays.asList(2,4,6);
```

```
List odds = Arrays.asList(3,5,7);
```

```
List primes = Arrays.asList(2,3,5,7,11);
```

```
List numbers = Stream.of(evens, odds, primes).flatMap(list -> list.stream()).collect(Collectors.toList());
```

```
System.out.println("flattened list : "+numbers);
```

Output: flattened list : [2,4,6,3,5,7,2,3,5,7,11]

3.What is the difference between passing int array and String array to Stream.of()?

Stream.of(int[]) gives Stream<int[]>

Stream.of(String[]) gives Stream<String>

So, when using Stream.of() with int[] , we get Stream<int[]> and then for getting ints from Stream, we use flatMapToInt(i -> Arrays.stream(i)) to get IntStream and then we can either use map() or forEach().

8 | Praveen Oruganti

Blog: <https://praveenorugantitech.blogspot.com>

Facebook Group: <https://www.facebook.com/groups/268426377837151>

Github repo: <https://github.com/praveenoruganti>

Email : praveenorugantitech@gmail.com

e.g.:

```
int[] arr = {1,2,3,4};  
Stream<int[]> streamArr = Stream.of(arr);  
IntStream intStream = streamArr.flatMapToInt(i -> Arrays.Stream(i));  
intStream.forEach(System.out :: println);
```

4.What is a Boxed Stream?

If we want to convert stream of objects to collection:

```
List<String> strings = Stream.of("how", "to", "do", "in", "java").collect(Collectors.toList());
```

The same process doesn't work on streams of primitives, however.

//Compilation Error:

```
IntStream.of(1,2,3,4,5).collect(Collectors.toList());
```

To convert a stream of primitives, we must first box the elements in their wrapper class and then collect them. This type of stream is called boxed stream.

Example of IntStream to List of Integers:

```
List<Integer> ints = IntStream.of(1,2,3,4,5).boxed().collect(Collectors.toList());
```

```
System.out.println(ints);
```

Output: [1,2,3,4,5]

5:List Spring Core and Stereotype annotations.

Spring Core Annotations:

- @Qualifier
- @Autowired
- @Configuration
- @ComponentScan
- @Required
- @Bean
- @Lazy
- @Value

Spring framework Stereotype annotations:

- @Component
- @Controller
- @Service
- @Repository

6.What is difference between @Controller and @RestController?

@Controller creates a Map of model object and finds a view for that.

@RestController simply returns the object and object data is directly written into HTTP response as JSON or XML.

@RestController = @Controller + @ResponseBody

@RestController was added in Spring 4.

7. Why Spring introduced @RestController when this job could be done by using @Controller and @ResponseBody?

The functioning or output of @Controller and @ResponseBody is the default behavior of RESTFUL web services, that's why Spring introduced @RestController which combined the behavior of @Controller and @ResponseBody.

8. Tell something about Swagger tools which you have used.

Swagger Editor: Can edit OpenAPI specifications.

Swagger UI : A collection of HTML/CSS/Javascript assets that dynamically generate beautiful documentation.

Swagger Codegen: Allows generation of API client libraries.

Swagger Parser: Library for parsing OpenAPI definitions from java.

Swagger Core: Java related libraries for creating, consuming and working with OpenAPI definitions.

Swagger Inspector: API testing tool

SwaggerHub : Built for teams working with OpenAPI.

Java Technical Interview @ IBM

1. How locking mechanism is implemented by JVM?

The implementation of locking mechanism in java is specific to the instruction set of the java platform.

For example with x86, it might use the CMPXCHG - atomic compare and exchange - at the lowest level to implement the fast path of the lock.

The CMPXCHG instruction is a compare-and-swap instruction that guarantees atomic memory access at the hardware level.

If the thread cannot acquire the lock immediately, then it could spinlock or it could perform a syscall to schedule a different thread.

Different strategies are used depending on the platform, JVM switches.

2. Which Event bus is used by Saga pattern?

When we use event based communication, a microservice publishes an event when something notable happens such as when it updates a business entity. Other microservices subscribe to those events.

When a microservice receives an event, it can update its own business entities which might lead to more events being published.

This publish/subscribe system is usually performed by using an implementation of an event bus.

The event bus is designed as an interface with the API needed to subscribe and unsubscribe to events and to publish events.

The implementation of this event bus can be a messaging queue like RabbitMQ or service bus like Azure service bus.

3.What is the use of Amazon EC2? What are the steps to deploy on EC2?

Amazon EC2: Amazon Elastic Compute Cloud

It offers ability to run applications on the public cloud.

It eliminates investment for hardware. There is no need to maintain the hardware. We can use EC2 to launch as many servers as we need.

Steps to deploy on EC2:

- Launch an EC2 instance and SSH into it. Note: This instance needs to be created first on Amazon console [console.aws.amazon.com]. And we should also have certificate to connect to EC2 instance.
- Install Node on EC2 instance, if our app is in Angular.
- Copy paste code on EC2 instance and install all dependencies.
- Start server to run.

OR:

- Build Spring boot app in the local machine. Make .jar file.
- Upload this .jar on S3.
- Create EC2 instance.
- SSH into it from the local computer. Now, we are in EC2 instance.
- We can install JDK now.
- And using java - .jar file path, we can run our application.

4.Why should we use ThreadPoolExecutor, when we have Executor Framework?

Source code of Executors.newFixedThreadPool() is:

```
public static ExecutorService newFixedThreadPool(int nThreads){  
  
    return new ThreadPoolExecutor(nThreads, nThreads, oL,    TimeUnit.MILLISECONDS, new  
    LinkedBlockingQueue<Runnable>());  
  
}
```

This method uses ThreadPoolExecutor class which uses default configuration as is seen in above code. Now, there are scenarios where default configuration is not suitable, say instead of LinkedBlockingQueue, a PriorityQueue needs to be used etc.

In such cases, caller can directly work on underlying ThreadPoolExecutor by instantiating it and passing desired configuration to it.

Note: One advantage of using `ThreadPoolExecutor` is that we can handle `RejectedExecutionException` using `ThreadPoolExecutor.discardPolicy()`.

5.What is the difference between Spring 2 and Spring 5?

Below are the differences between Spring 2 and Spring 5:

- JDK baseline update
- Core framework revision
- Reactive programming model
- Core Container updates
- Testing improvements

6.What is the difference between @SpringBootApplication and @EnableAutoConfiguration?

Following are the differences between `@SpringBootApplication` and `@EnableAutoConfiguration`:

Availability: `@SpringBootApplication` was introduced in version 1.2, while `@EnableAutoConfiguration` was introduced in version 1.0.

Purpose: `@EnableAutoConfiguration` enables auto configuration feature of Spring Boot application which automatically configures things if certain classes are present in classpath e.g. it can configure Thymeleaf `TemplateResolver` and `ViewResolver` if Thymeleaf is present in the classpath.

On the other hand, `@SpringBootApplication` does three things:

- It allows us to run the main class as a jar with an embedded container [Web server Tomcat].
- It enables java configuration.
- It enables component scanning.

7.What happens when we call SpringApplication.run() method in main class of SpringBoot application?

Syntax of the class containing main method looks like code below:

```
@SpringBootApplication
public class StudentApplication{
    public static void main(String[] args){

        return SpringApplication.run(StudentApplication.class, args);
    }
}
```

When we run this class as a java application, then our application gets started.

`SpringApplication.run()` is a static method and it returns an object of `ConfigurableApplicationContext`.

```
ConfigurableApplicationContext ctx = SpringApplication.run(StudentApplication.class, args);
```

Thus, Spring container gets started once `run()` method gets called.

Spring container once started is responsible for:

- Creating all objects: This is done by `@ComponentScan`. Remember `@SpringBootApplication` is a combination of `@ComponentScan` + `@Configuration` + `@EnableAutoConfiguration`

- Dependency Injection
- Managing the lifecycles of all beans.

Java Interview @ Indie Games

1.How do you kill a thread in java?

There are two ways to kill a thread:

- Stop the thread [Deprecated now]
- Use a volatile variable and running thread will keep checking it's value and return from it's run method in an orderly fashion.

2.Why is Thread.stop() deprecated?

Because it is inherently unsafe. Stopping a thread causes it to unlock all the monitors that it has locked.(The monitors are unlocked as the ThreadDeath exception propagates up the stack.) If any of the objects previously protected by these monitors were in an inconsistent state, other threads may now view these objects in an inconsistent state. Such objects are said to be damaged.

When threads operate on damaged objects, arbitrary behavior can result. This behavior may be subtle and difficult to detect.

ThreadDeath exception kills thread silently, thus the user has no warning that this program may be corrupted.

3.How to check if a String is numeric in java?

We can use Apache Commons Lang 3.4 and 3.5 versions.
StringUtils.isNumeric() or NumberUtils.isParsable().

4.How can we timeout a thread? I want to run a thread for some fixed amount of time.If it is not completed within that time, I want to either kill it or throw some exception. How to do it?

We can use ExecutorService framework of Java 5.

```
public class Test{

    public static void main(String[] args){

        ExecutorService service = Executors.newSingleThreadExecutor();
        Future<String> future = service.submit(new Task());

        try{

            future.get(4, Timeout.SECONDS);
        }
        catch(Exception e){
            future.cancel(true);
        }

        executor.shutdownNow();
    }
}
```

```

    }
}

class Task implements Callable<String>{

    @Override
    public String call() throws Exception{

        Thread.sleep(5000);
        return "";
    }

}

```

In this code, thread sleeps for 5 seconds, but future will wait for 4 seconds and it will timeout and cancel it.

5.How to wait for all threads to finish using ExecutorService?

We can use shutdown() and awaitTermination() methods of ExecutorService.

```

ExecutorService service = Executors.newFixedThreadPool(4);

while(...){

    service.execute(new Task());

}
service.shutdown();
try{
    service.awaitTermination(Long.MAX_VALUE, TimeUnit.NANOSECONDS);
}
catch(InterruptedException e){

}

}

```

6.Can we create deadlock without thread in java?

It is not possible to run the code without atleast one thread. A single thread can block itself in some cases e.g. attempting to upgrade a read lock to a write lock. When a thread resource starves, it is called a livelock.

It is also possible to create a deadlock without creating an additional thread e.g. the finalizer thread and the main thread can deadlock each other.

Interview @ 1mg

1.What is System.out, System.in and System.err?

out, in and err all are fields in System class.

out: The standard output stream. This stream is already open and ready to accept output data. Typically this stream corresponds to display output or another output destination specified by the host environment or user.

Basically, it gives `PrintStream`. And all `print()` methods belong to class `PrintStream`.

```
public static final PrintStream out
```

in: The standard input Stream. This stream is already open and ready to supply input data. typically this stream corresponds to keyboard input or other input source specified by the host environment or user.

```
public static final InputStream in
```

err: The standard error output stream. This stream is already open and ready to accept output data. By convention, this output stream is used to display error messages.

```
public static final PrintStream err
```

2. What is the difference between `Class.forName()` and `Class.forName().newInstance()` methods?

Lets take an example to understand the difference better:

```
public class Demo{

    public Demo(){

    }

    public static void main(String[] args){

        Class clazz = Class.forName("Demo");
        Demo demo = (Demo)clazz.newInstance();
    }

}
```

`Class.forName()` returns the `Class` object associated with the class or interface with the given string name.

Then, calling `clazz.newInstance()` creates a new instance of the class represented by this `Class` object. This class is instantiated as if by a new expression with an empty argument list.

3. Why non-static variables are not allowed in static methods?

non-static variables means instance variables and they are only initialized when an instance is created. As static methods can be called without creating an instance, so accessing non-initialized instance variable is wrong as instance doesn't exist.

So, only way to access non-static variable in static method is that, just create instance of class in static method and access that variable.

```
public class StaticTest{
```

```

private int count = 0;

public static void main(String[] args){

    StaticTest test = new StaticTest();

    test.count++;

}

}

```

4.What is the difference between HTTP HEAD and GET verbs?

HTTP HEAD is almost identical to GET, but without the response body. Means in HTTP HEAD, we don't get any response body.

In other words, if GET /users returns a list of users, then HEAD /users will make the same request , but will not return the list of users.

HEAD requests are useful for checking what a GET request will return before actually making a GET request - like before downloading a large file or response body.

5.What is the difference between HTTP GET and POST methods?

There are multiple differences between GET and POST methods:

- GET is used to request data from a specified resource. POST is used to send data to server to create a resource.
- GET requests can be cached. POST requests cannot be cached.
- GET requests remain in the browser history. POST requests do not remain in the browser history.
- GET requests can be bookmarked. POST requests cannot be bookmarked.
- GET requests have length restrictions. POST requests have no restrictions on data length.

6.Is there any speed increase while indexing a table? And will indexing every column defeat the purpose of indexing?

Indexing any table, either memory or file system based, will speed up queries that select or sort results based on that column.

This is because the index works like a tree structure and the search distance depends upon the depth of the tree, which increases a lot slower than the row count of the column.

Indexing every column doesn't defeat the purpose of the index, but it will slow up inserts and updates because those changes will cause an update of every index of that table. Also, the indexes take up space on the database server.

7.What is covariant return type?

In covariant return type, parent's instances can be replaced with child's instances.

e.g.:


```

class WildAnimal{

    public String willYouBite(){
        return "Yes";
    }

}

class Lion extends WildAnimal{

    public String whoAreYou(){

        return "Lion";
    }

}

class BengalTiger extends WildAnimal{

    public String whoAreYou(){

        return "Tiger";
    }

}

class Zoo{

    public WildAnimal getWildAnimal(){

        return new WildAnimal();
    }

}

class AfricaZoo extends Zoo{

    @Override
    public Lion getWildAnimal(){

        return new Lion();
    }

}

class IndiaZoo extends Zoo{

    @Override
    public BengalTiger getWildAnimal(){

```

```

        return new BengalTiger();
    }
}

public class Covariant{

    public static void main(String[] args){

        AfricaZoo africaZoo = new AfricaZoo();
        System.out.println(africaZoo.getWildAnimal().whoAreYou());
    }

}

```

So, in class AfricaZoo, parent class WildAnimal is replaced by child class Lion while overriding method.

Java Technical Lead interview @ Accenture

1.What is the difference between SSL and TLS?

There are multiple differences between SSL and TLS:

Alert Messages:

SSL has "No Certificate" alert message. TLS protocol removes the alert message and replaces it with several other alert messages.

Record Protocol:

SSL uses MAC [Message Authentication Code] after encrypting each message while TLS on the other hand uses HMAC - a hash based message authentication code after each message encryption.

Handshake Process:

In SSL, the hash calculation also comprises the master secret and pad while in TLS, the hashes are calculated over handshake message.

2.What changes are introduced in Java 10?

There are multiple changes that have been done in java 10. Explaining all the changes below:

- var keyword
- Unmodifiable Collection enhancements
- G1GC performance improvement
- New JIT compiler [Created in pure java]
- Alternate memory devices for allocating heap
- Application class data sharing

var keyword:

It improves readability. It is not a reserved word, means it can be used as a class name or variable name. What the var keyword does is , turn local variable assignments:

```
HashMap<String, String> hm = new HashMap<>();
```

into

```
var hm = new HashMap<String,String>();
```

Unmodifiable Collection enhancements:

```
var vegetables = new ArrayList<>(Lists.of("Broccoli", "Celery", "Carrot"));
```

```
var unmodifiable = Collections.unmodifiableList(vegetables);  
vegetables.set(0, "Radish");  
var v = unmodifiable.get(0); // var v contains Radish.
```

It is so because, `unmodifiableList()` returns an unmodifiable view collection.

An unmodifiable view collection is a collection that is unmodifiable and that is also a view onto a backing collection.

Note that changes in the backing collection is still possible, and if they occur, they are visible through the unmodifiable view.

So, if we need completely unmodifiable collection then, java 10 has added two new API's.

```
var unmodifiable = List.copyOf(vegetables);
```

The **second API** adds three new methods to the `Collector` class in the `Stream` package.

We can now stream directly into an unmodifiable collection using `toUnmodifiableList`, `toUnmodifiableSet` and `toUnmodifiableMap`.

3.What is the improvement in G1GC in java 10?

Java 9 made the Garbage-First garbage collector (G1GC) by default, replacing the concurrent Mark-sweep garbage collector (CMS).

Java 10 introduces performance improvements to G1GC.

In java 10, G1GC is getting a performance boost with the introduction of full parallel processing during a full GC.

4.What is Application class-data sharing?

Java 5 introduced Class-Data sharing (CDS) to improve startup times of small java applications.

The general idea was that when the JVM first launched, anything loaded by the bootstrap classloader was serialized and stored in a file on disk that could be reloaded on future launches of the JVM.

This means that multiple instance of the JVM shared the class metadata so it wouldn't have to load them all every time.

5.How to write thread safe code in java?

Example of Non thread safe code in java:

```
public class Counter{
```

19 | Praveen Oruganti

Blog: <https://praveenorugantitech.blogspot.com>

Facebook Group: <https://www.facebook.com/groups/268426377837151>

Github repo: <https://github.com/praveenoruganti>

Email: praveenorugantitech@gmail.com

```

private int count;

// This method is not thread safe because ++ is not an atomic operation.
public int getCount(){

    return count++;
}
}

```

How to make code thread safe in java:

There are multiple ways to make this code thread safe:

- 1). Use synchronized keyword in java and lock the getCount() method so that only one thread can execute it at a time.
- 2). Use atomic integer , which makes this ++ operation atomic and since atomic operations are thread-safe and saves cost of external synchronization.

Below is the thread-safe version of Counter class in java:

```

public class Counter{

    private int count;

    AtomicInteger atomicCount = new AtomicInteger(0);

    //This method is thread safe now because of locking and synchronization
    public synchronized int getCount(){

        return count++;
    }

    //This method is thread safe because count is incremented atomically.
    public int getCountAtomically(){

        return atomicCount.incrementAndGet();
    }
}

```

6.How does AtomicInteger work?

Java 5 introduced java.util.concurrent.atomic package with a motive to provide a small kit of classes that support lock-free thread-safe programming on single variables.

AtomicInteger uses combination of volatile and CAS [Compare and Swap] to achieve thread safety for Integer Counter.

It is non blocking in nature.

20 | Praveen Oruganti

Blog: <https://praveenorugantitech.blogspot.com>

Facebook Group: <https://www.facebook.com/groups/268426377837151>

Github repo: <https://github.com/praveenoruganti>

Email : praveenorugantitech@gmail.com

Compare-and-Swap:

CAS is an atomic instruction used in multi-threading to achieve synchronization. It compares the contents of a memory location with a given value and only if they are same, modifies the contents of that memory location to a given new value. This is done as a single atomic operation.

The atomicity guarantees that the new value is calculated based on up-to-date information. If the value has been updated by another thread in the meantime, the write would fail.

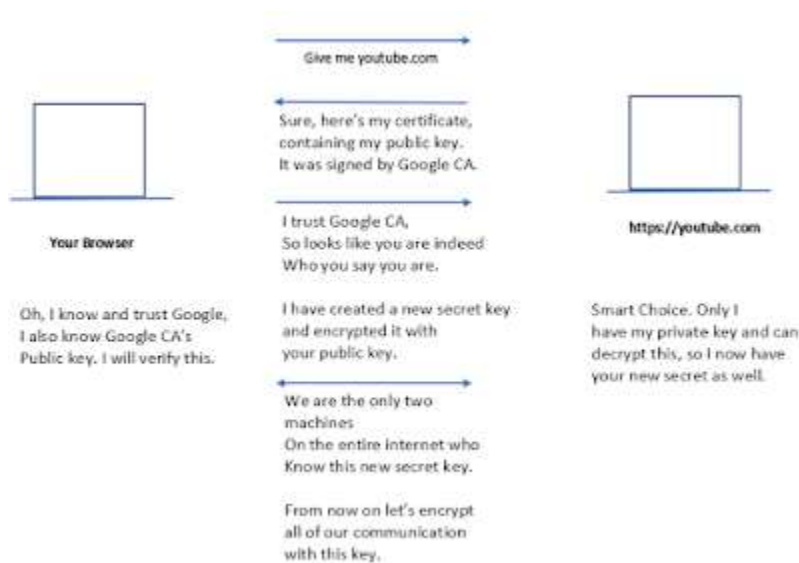
Java Interview @ Altran

1.How does HTTPS work?

In HTTPS, every message is encrypted or decrypted by use of public/private keys.

So, we need to trust that public key cryptography and signature works.

- Any message encrypted with Google's public key can only be decrypted with Google's private key.
- Anyone with access to Google's public key can verify that a message (signature) could only have been created by someone with access to Google's private key.



2.What is a Certificate Authority? How is a certificate signed?

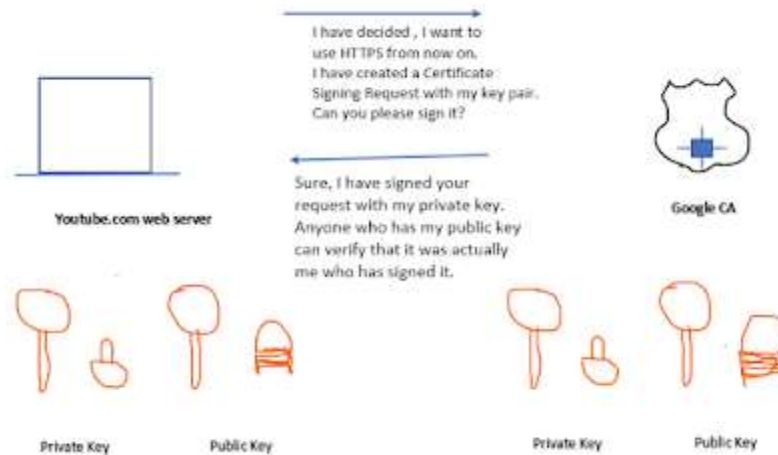
Suppose youtube.com web server uses HTTP currently. And now youtube wants to secure communication using HTTPS.

Also there is Google CA which is considered as a trusted certificate authority.

As any party involved in public key cryptography, the Google certificate authority has a private key and a public key.

As youtube.com wants to communicate using HTTPS, they also need to create a new pair: public key and private key.

Now youtube.com web server creates a Certificate signing request with this key pair.



So most of the browsers [when they are delivered] has a list of these certificates issued by known Certificate Authorities [Google, Symantec, Thwate].

So, when we try to open youtube.com, then as youtube.com uses HTTPS, it sends certificate to the browser. As browser knows the Google's public key [as it is having that certificate with Google's public key], so it can use that key to know that this certificate is actually signed by trusted certificate authority [Google].

Means. now that public key can be used to create and encrypt a secret key which will then be used in symmetric encryption.

3.Explain the difference between HTTP, HTTPS, SSL and TLS?

HTTP: Hyper Text Transfer Protocol

Using this protocol, data is transferred in clear text.

HTTPS: Secure Hypertext Transfer Protocol

Means, HTTP with a security feature.

Encrypts the data that is being retrieved by HTTP. So, it uses encryption algorithms to encrypt the data.

HTTPS uses SSL [Secure Socket Layer] protocol to protect the data.

SSL: This protocol is used to ensure security on the internet. Uses public key encryption to secure data.

TLS: Transport Layer Security

It is the latest industry standard cryptographic protocol. It is a successor to SSL.

And it is based on same specifications.

And like SSL, it also authenticates the server, client and encrypts the data.

4.What are the collection types in hibernate?

Below is the list of collection types in hibernate:

- Bag
- List
- Map
- Array
- Set

5.What is hibernate proxy?

Mapping of classes can be made into a proxy instead of a table.

A proxy is returned when actually a load() is called on a session. The proxy contains actual method to load the data.

The proxy is created by default by hibernate for mapping a class to a file.

6.What is the difference between load() and get() methods in hibernate?

Hibernate's session interface provides several overloaded load() [It will not hit database] methods for loading entities. Each load() method requires the object's primary key as an identifier and it is mandatory to provide it.

In addition to the ID, hibernate also needs to know which class or entity name to use to find the object with that ID.

In case of get() method, we will get return value as NULL if identifier is absent.

But in case of load() method, we will get a runtime exception [ObjectNotFoundException].

When we call load() on hibernate's session object, then hibernate doesn't make a call to database. It creates and returns a proxy object. Now, when some state is fetched from the proxy object, then hibernate issues the appropriate SQL statement to database and builds the real persistent object.

So, when actual data is requested using proxy object and if no data exist for the identifier [id], then ObjectNotFoundException is thrown.

On the other hand, when we call `get()` method on hibernate's session object, then hibernate immediately issues a SQL statement to the database to fetch associated data [usually a row in database] to rebuild the requested persistent object.

So, if no data is found for the requested identifier [id], the method will return null.

Multithreading Interview in Ericsson

1.Can a thread acquire multiple locks at the same time and how?

Yes, a thread can acquire locks on multiple objects at the same time.

Also, a thread can acquire lock on the same object multiple times.

Locks obtained using `synchronized` are implicitly reentrant.

Whenever we use nested `synchronized` blocks with 2 different objects, then the same thread can acquire lock on both of these objects. But care needs to be taken in this case, as if we take two pairs of `synchronized` blocks with different order of similar objects, then it may cause deadlock.

2.What are the steps to avoid deadlock in java programs?

There are multiple ways in which we can avoid deadlocks:

- Avoid using nested `synchronized` blocks
- Always request and release locks in the same order.
- Use synchronization only whenever required.
- If two threads are waiting for each other to release shared resources, then this issue can be resolved by letting each waiting thread retry the operation at random interval until they successfully acquire the resource.
- Do not perform operations that can block while holding a lock. e.g. Do not perform file I/O over the network while holding a lock.

3.When we call Thread object's run() method, then in which thread that run() method gets executed?

Calling `run()` method on `Thread`'s object will cause execution of that method in current thread.

If a thread object was constructed by instantiating the subclass of `Thread` and fails to override `run()` method, then any calls to the `run()` method will invoke `Thread.run()`, which does nothing.

So, we should never call `run()` method on thread object.

4. When we can and should use ThreadGroup?

Each thread in java is assigned to one `ThreadGroup` upon it's creation. These groups are implemented by `java.lang.ThreadGroup` class.

When thread group name is not specified, the main default thread group is assigned by JVM.

`ThreadGroups` are useful for keeping threads organized.

There are few useful methods which can be used for specific use cases.

ThreadGroup.activeCount():

24 | Praveen Oruganti

Blog: <https://praveenorugantitech.blogspot.com>

Facebook Group: <https://www.facebook.com/groups/268426377837151>

Github repo: <https://github.com/praveenoruganti>

Email : praveenorugantitech@gmail.com

- Returns an estimated number of active threads in the current thread's thread group and its subgroups.

ThreadGroup.enumerate():

- It copies into the specified array every active thread in this thread group and its subgroups.

5.How to decide which method to call : notify() or notifyAll() ?

Whenever we call notify() or notifyAll() methods, it is guaranteed that only one thread will get the lock and resume execution.

Calling notify() is permitted under following conditions:

- All waiting threads have identical condition predicates.
- All threads perform the same set of operations after waking up. That is, any one thread can be selected to wake up and resume for a single invocation of notify.
- Only one thread is required to wake upon the notification.

6.Why to use Thread pool for handling multiple tasks?

Thread Pools allows a system to limit the number of simultaneous requests that it process to a number that it can comfortable serve.

If we don't use Thread Pool then, for every request, creating a new thread consumes a lot of time and resources and for task processing.

Describing here some of the benefits of using Thread Pool:

- Thread pool minimizes the overhead of thread lifecycle management because the threads in a thread pool can be reused and can be efficiently added or removed.
- It also reduces the time and resources required for thread creation.
- It also ensure graceful degradation of service when traffic bursts.

7.What is bounded thread pool and what happens when we execute interdependent tasks in bounded thread pool?

A bounded thread pool is a pool in which we specify an upper limit on the number of threads that can currently execute in a thread pool.

We should not use bounded thread pool to execute tasks that depend on the completion of other tasks in the pool.

A form of deadlock called thread-starvation deadlock arises when all the threads executing in the pool are blocked on tasks that are waiting on an internal queue for an available thread in which to execute.

Code Review Practices

While doing the code review, multiple factors are taken into account : Clean code, Security, performance, General points etc.

I'm explaining each one of them below:

25 | Praveen Oruganti

Blog: <https://praveenorugantitech.blogspot.com>

Facebook Group: <https://www.facebook.com/groups/268426377837151>

Github repo: <https://github.com/praveenoruganti>

Email : praveenorugantitech@gmail.com

Java Code Review Checklist:

Clean Code:

- **Use intention-revealing names**
 - Names should be such that, they reveal the purpose.
- **Use solution-problem domain names**
 - Names should be such that they tell about the actual solution or problem.
- **Classes should be small**
 - Keep the code in a class as less as possible and create other classes or subclasses for specific purpose.
- **Functions should be small**
 - Always break the functions in small.
- **Functions should do one thing**
 - Keep one separate function for each action.
- **Don't repeat yourself (Avoid duplication)**
 - Don't write duplicate codes. Check the entire code before writing the same code twice in the project.
- **Explain yourself in code :**
 - Write proper Class level and method level Comments.
- **Use exceptions rather than return codes**
- **Don't return null**
 - Never return null values from a function.

Security:

- Make class final if not being used for inheritance
- Avoid duplication of code
- Minimize the accessibility of classes and members
- Document security related information
- Input into a system should be checked for valid data size and range
- Release resources[Streams , Connections] in all cases.
- Purge sensitive information from exceptions
- Don't log highly sensitive information
- Avoid dynamic SQL, use prepared statement
- Limit the accessibility of packages, classes interfaces, methods and fields.
- Avoid exposing constructors of sensitive classes.
- Avoid serialization of sensitive classes
- Only use JNI when necessary

Performance:

- **Avoid excessive synchronization**
 - Don't use synchronize constructs unnecessarily
- **Keep synchronized sections small**

- **Beware the performance of String concatenations**
- Avoid joining strings as much as possible.
- **Avoid creating unnecessary objects.**
- Try to create only local objects and also create them based on actual need.

General:

- Don't ignore exceptions
- Return empty Arrays or Collections , not nulls
- In public classes, use accessor methods not public methods
- Avoid finalizers
- Refer to objects by their interfaces
- Always override toString()
- Document thread safety
- Use marker interfaces to define types

Static Code Analysis:

- Check static code analyzer report for the classes added/modified

Java Technical Architect interview

1.What is the difference in using Eureka and spring cloud consul?

Eureka and Spring cloud Consul are both Service Discovery tools.
Difference between them lie in multiple factors.

The architecture of Eureka is primarily client/server with a set of eureka servers per datacentre , usually one per availability zone.
Typically clients of eureka use an embedded SDK to register and discover services.

Eureka provides a weak consistent view of services using best effort replication. When a client registers with a server , that server will make an attempt to replicate to the other servers but provides no guarantee.
Service registrations have a short Time-To-Live, requiring clients to heartbeat with the servers. Unhealthy services or nodes will stop heartbeating, causing them to timeout and be removed from the registry.

Consul provides a super set of features, including richer health checking, key/value store and multi datacentre awareness. Consul requires a set of servers in each datacentre , along with an agent on each client, similar to using a sidecar like Ribbon. The Consul agent allows most applications to be consul unaware, performing the service registration via configuration files and discovery via DNS or load balancer sidecars.

Consul provides a strong consistency guarantee since servers replicate state using the Raft protocol. Consul supports a rich set of health checks.
Client nodes in Consul participate in a gossip based health check , which distributes the work of health checking, unlike centralized heartbeating which becomes a scalability challenge.

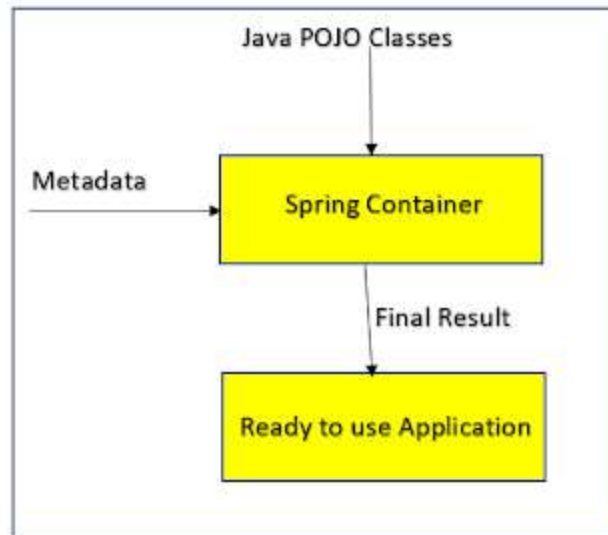
In Consul, discovery requests are directed to the elected consul leader which allows them to be strongly consistent by default.

The strongly consistent nature of Consul means it can be used as a locking service for cluster coordination. Eureka does not provide similar guarantee and typically requires running ZooKeeper for services that need to perform coordination or have stronger consistency needs.

2.How many types of containers are there in spring framework?

There are two types of Spring IOC containers:

- BeanFactory Container
- ApplicationContext Container



Spring BeanFactory container is the simplest container which provides basic support for DI. It is defined by `org.springframework.beans.factory.BeanFactory` interface. XMLBeanFactory is the main implementation of this interface. It reads configuration metadata from xml files for creating a fully configured application.

BeanFactory container is preferred where resources are limited to mobile devices or applet based applications.

Spring ApplicationContext Container:

It is defined by `org.springframework.context.ApplicationContext` interface. The ApplicationContext container has all the functionalities of BeanFactory container. It is generally recommended over BeanFactory container.

The most common implementations are :

- FileSystemXmlApplicationContext
- ClassPathXmlApplicationContext
- WebXmlApplicationContext

3.What is the difference between @Controller and @RestController?

@RestController = @Controller + @ResponseBody

@RestController was added in Spring 4.

@Controller : The job of @Controller is to create a map of model object and find a view.

@ResponseBody: It will just append the result object as JSON or XML in Http response.

@RestController : It simply return the object and object data is directly written into HTTP response as JSON or XML.

4.What is the difference between @RequestParam and @RequestAttribute?

@RequestParam:

This annotation is used to access the query parameters from the HTTP request URL. e.g.:

`http://example.com?param1=1¶m2=2`

In this URL, param1 and param2 are query parameters and they are accessed in methods like:

```
@RequestMapping("/")
public void method(@RequestParam("param1") String par){
}
}
```

@RequestAttribute:

It is used to access objects which have been populated on the server-side but during the same HTTP request.

e.g.:

We want to maintain a counter to know the number of visits to that page or number of requests made to a specific URL. So, in this case, we take an interceptor with AtomicInteger to increment the counter on every request.

e.g.:

The Interceptor:

```
public class Counter extends HandlerInterceptorAdapter{

    private AtomicInteger counter = new AtomicInteger(0);

    @Override
    public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler)
    throws Exception{

        request.setAttribute("visitorCounter", counter.incrementAndGet());
    }
}
```

```
}  
}
```

The Controller class:

```
public class CounterController{  
  
    @RequestMapping("/")  
    @ResponseBody  
    public String handle(@RequestAttribute("visitorCounter") Integer counter){  
  
        // code here.....  
    }  
}
```

Interview questions on SpringBoot

1.What is the use case of Spring Boot?

Spring Boot is used while creating microservices. As with increasing features in our applications, we need to create a separate microservice for each new feature and doing entire setup for this including adding all dependencies will take time. So, in this scenario, Spring Boot is required.

In a monolithic application, we just do setup and dependency addition only once, so in monolithic application, we don't need spring boot.

2.What are the steps to use Spring-boot-starter-caching ?

Steps for using spring-boot-starter-cache are following:

1). Add dependency:

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

2). Add @EnableCaching annotation on main Spring Boot Application class.

The @EnableCaching triggers a post-processor that inspects every spring bean for the presence of caching annotations on public methods. If such an annotation is found, a proxy is automatically created to intercept the method call and handle the caching behavior accordingly.

The post-processor handles the @Cacheable, @CachePut and @CacheEvict annotations.

Spring boot automatically configures a suitable CacheManager to serve as a provider for the relevant cache.

3). Put @Cacheable and other annotations on methods.

e.g.:

```
public interface BookRepository{
```

```

    Book getByIsbn(String isbn);
}

```

Use our own implementation if we are not using Spring Data.

```

public class SimpleBookRepository implements BookRepository{

    @Override
    @Cacheable("books")
    public Book getByIsbn(String isbn){

        return new Book(isbn);
    }

}

```

3.What happens when we call **SpringApplication.run()** method in main class of **SpringBoot** application?

Syntax of the class containing main method looks like code below:

```

@SpringBootApplication
public class StudentApplication{

    public static void main(String[] args){

        SpringApplication.run(StudentAppplication.class, args);
    }

}

```

When we run this class as a java application, then our application gets started.

SpringApplication.run() is a static method and it returns an object of **ConfigurableApplicationContext**.

ConfiguravleApplicationContext ctx = **SpringApplication.run**(StudentApplication.class, args);

Thus, Spring container gets started once **run()** method gets called.

Spring container once started is responsible for:

- Creating all objects: This is done by **@ComponentScan**. Remember **@SpringBootApplication** is a combination of **@ComponentScan** + **@Configuration** + **@EnableAutoConfiguration**
- Dependency injection
- Managing the lifecycle of all beans

Steps executed under run() method are as follows:

- Application Context is started.
- Using application context, auto discovery occurs: **@ComponentScan**
- All default configurations are setup.
- An embedded servlet container is started e.g. Tomcat. No need to setup a separate web server. **Note:** Embedded servlet container is launched only if the web is mentioned in a dependency.

4.What are Spring Boot Actuator endpoints?

Spring Boot Actuator is used to monitor and manage application usages in production environment without coding and configuration for any of them.

This monitoring and managing information is exposed via REST like endpoint URL's.

Actuator Endpoints:

/env : Returns list of properties in current environment

/health : Returns application health information.

/beans : Returns a complete list of all the Spring beans in the application.

/trace : Returns trace logs [By default the last 100 HTTP requests]

/dump : It performs a thread dump

/metrics : It shows several useful metrics information like JVM memory used, CPU usage , open files and much more.

Note: These endpoints can be explicitly enabled/disabled.

If we need to list all endpoints in the browser:

Just run the application and from browser , use localhost:8080/actuator. It will list 2 endpoints by default : info and health

For other endpoints, we need to expose them manually.

Now, if we want to explore health endpoint, then just open localhost:8080/actuator/health URL and it will just display "status" : "UP"

We can enable or disable the endpoint from application.properties file:

management.endpoint.shutdown.enabled = true

By default all the endpoints are enabled except shutdown endpoint.

5.By default on which port Spring Boot Actuator runs?

On port 8080

We can override that setting by adding application.properties file.

6.Difference between Spring 2 and Spring 5?

Below are the differences between Spring 2 and Spring 5 versions:

- JDK Baseline update
- Core framework revision

32 | Praveen Oruganti

Blog: <https://praveenorugantitech.blogspot.com>

Facebook Group: <https://www.facebook.com/groups/268426377837151>

Github repo: <https://github.com/praveenoruganti>

Email : praveenorugantitech@gmail.com

- Reactive programming model
- Core Container updates
- Testing improvements

7.What is the difference between @SpringBootApplication and @EnableAutoConfiguration?

Following are the differences between @SpringBootApplication and @EnableAutoConfiguration:

Availability : SpringBootApplication was introduced in version 1.2 , While EnableAutoConfiguration was introduced in version 1.0

Purpose : @EnableAutoConfiguration enables auto configuration feature of Spring Boot application which automatically configures things if certain classes are present in classpath e.g. : it can configure Thymeleaf TemplateResolver and ViewResolver if Thymeleaf is present in the classpath.

On the other hand , @SpringBootApplication does three things.

- It allows us to run the main class as a jar with an embedded container [Web server Tomcat].
- It enables java configuration.
- It enables component scanning.

Java Technical Architect interview @ Tech Mahindra

1.What are Spring boot actuator endpoints?

Spring boot actuator is used to monitor and manage application usage in production environment without coding and configuration for any of them.

This monitoring and managing information is exposed via REST like endpoint URL's.

Actuator endpoints:

/env : Returns list of properties of current environment.

/health : Returns application health information.

/beans : Returns a complete list of all the spring beans in the application.

/trace : Returns trace logs [By default last 100 HTTP requests]

/dump : It performs a thread dump.

/metrics : It shows several useful metrics information like JVM memory usage, CPU usage, open files and much more.

Note: These endpoints can be explicitly enabled/disabled.

If we need to list all endpoints in the browser:

Just run the application and from the browser, use localhost:8080/actuator. It will list 2 endpoints by default : info and health.

For other endpoints, we need to expose them manually.

Now, if we want to explore health endpoint, then just open localhost:8080/actuator/health URL and it will just display "status" : "UP"

We can enable or disable the endpoint from application.properties file:

```
management.endpoint.shutdown.enabled = true
```

By default , all the endpoints are enabled except shutdown endpoint.

2.What is the difference between map and flatmap in java 8?

map() method is used to map an object or entry from stream to some other value.

flatMap() method on the other hand applies map() on the entries and then flatten the result.

e.g.:

Suppose we have a string array with entries:

```
String[] strArray = {"12", "46"};
```

And we have to find all permutations of these strings.

So output with map() method will be: [12, 21] , [46, 64]

While the output with flatMap() method will be : [12, 21, 46, 64]

Lets take another example:

```
List<Integer> evens = Arrays.asList(2,4,6,8);
```

```
List<Integer> odds = Arrays.asList(1,3,5,7);
```

```
List<Integer> primes = Arrays.asList(5,7,11,13);
```

```
List numbers = Stream.of(evens, odds, primes).flatMap(list-> list.stream()).collect(Collectors.toList());
```

```
System.out.println("List = "+numbers);
```

Note: this output is possible only with flatMap() method.

3.Explain Spring Data JPA versus Hibernate?

JPA : Java Persistence API provides specification for creation, deletion, persistence and data management from java objects to relations [tables] in database.

Hibernate : There are various providers which implement JPA. Hibernate is one of them. So , we have other providers as well e.g.: Eclipse Link.

Spring Data JPA: This is another layer on top of JPA which spring provides to make coding easier.

4.When to use @Component and when to use @Bean?

@Component is a class level annotation while @Bean is a method level annotation.

@Component is preferable for component scanning and automatic wiring. Sometimes automatic configuration is not an option. When? Let's imagine that we want to wire components from 3rd party libraries [where we don't have the source code, so we can't annotate the class with @Component], so automatic configuration is not possible.

So, in this case, we should use @Bean.

The @Bean annotation returns an object that Spring should register as a bean in application context/Spring IOC container. The body of the method bears the logic responsible for creating the instance.

Another difference is that:

@Component does not decouple the declaration of a bean from the class definition whereas @Bean decouples the declaration of the bean from the class definition.

There are two ways to create beans:

One is to create a class with an annotation @Component. The other is to create a method and annotate it with @Bean.

@Bean requires the @Configuration annotation on the class in which @Bean is used. Whereas @Component doesn't require the class to be annotated with @Configuration.

Once we run the Spring project, the class with @ComponentScan annotation would scan every class with @Component on it and store the instance of this class to the IOC container.

Another thing, the @ComponentScan would do is, running the methods with @Bean on it and store the returned object to the IOC container as a bean.

5.Explain the use case with example where @Bean is more beneficial to use than @Component?

Let's suppose, we want some specific implementation depending on some dynamic state. @Bean is perfect for that case.

```
@Bean
@Scope("prototype")
public Service anyService(){

    switch(state){
        case 1:
            return new Impl1();

        case 2:
            return new Impl2();

        case 3:
            return new Impl3();

        default:
            return new Impl();
    }
}
```

```
}  
}
```

In the above case, only @Bean can be used , not the @Component.

Note: We can use both @Component and @Configuration annotations in our same application/module.

6.Explain the difference between @Service and @Component in Spring.

@Component is a generic for other stereotypes.

So, we can replace @Component with @Service, @Repository or @Controller and nothing will change. But for better readability , we should use @Repository, @Service and @Controller.

They all inform spring that the class is involved in the DI context.

In order to configure spring so that it can provide us with the instances of the class we need , we are supposed to tell spring what objects are involved and how they are built. To do this, we can use XML configuration file or java annotations.

They also have semantic meaning:

- @Controller : @Component belonging to presentation layer
- @Service : @Component belonging to Service/Use case layer
- @Repository : @Component belonging to Persistence layer

Interview @ SirionLabs

1.What is the difference between Saga and 2 Phase Commit patterns?

Saga and 2 Phase Commit patterns are used for distributed transactions.

A distributed transaction would do what a transaction would do but on multiple databases.

Differences:

- 2 Phase Commit is for immediate transactions. Saga pattern is for long running transactions.
- In 2 Phase Commit, there is a single controlling node and if it goes down, then entire system fails. In Saga Pattern, There are multiple coordinators: Saga Execution Coordinator.
- 2 Phase Commit works in 2 steps: Prepare and Commit. And the result of all the transactions are seen to the outer world at once together. But in Saga, there is only one step for execution and commit, but if any transaction fails, then compensatory transaction gets executed for rollback operation.

2.Which database should be used when storing billions of payment requests come to database and why?

In this case, data is relational in nature like payment amount, user id, payment date, payment location, account number etc.

So, I can easily use SQL Server 2016 with proper indexes on proper fields.

3.Which Event Bus is used by Saga pattern?

When we use event-based communication, a microservice publish an event when something notable happens such as when it updates a business entity. Other microservices subscribe to these events. When a microservice receives an event, it can update it's own business entities which might lead to more events being published.

This publish/subscribe system is usually performed by using an implementation of an event bus. The event bus is designed as an interface with the API needed to subscribe and unsubscribe to events and to publish events.

The implementation of this event bus can be a **messaging queue like RabbitMQ or service bus like Azure service bus.**

4.Explain ACID properties of database.

A transaction is a single unit of work which accesses and possibly modifies the contents of a database. Transactions access data using read and write operations.

In order to maintain consistency in a database, before and after transaction, certain properties are followed. These are called ACID properties.

Atomicity:

By this, we mean that either the entire transaction takes place at once or doesn't happen at all.

Consistency:

This means that integrity constraints must be maintained so that the database is consistent before and after the transaction. It refers to the correctness of a database.

Isolation:

This property ensures that multiple transactions can occur concurrently without leading to inconsistency of database state. Transactions occur independently without interference.

Changes occurring in a particular transaction will not be visible to any other transaction until that particular change in that transaction is written to memory or has been committed.

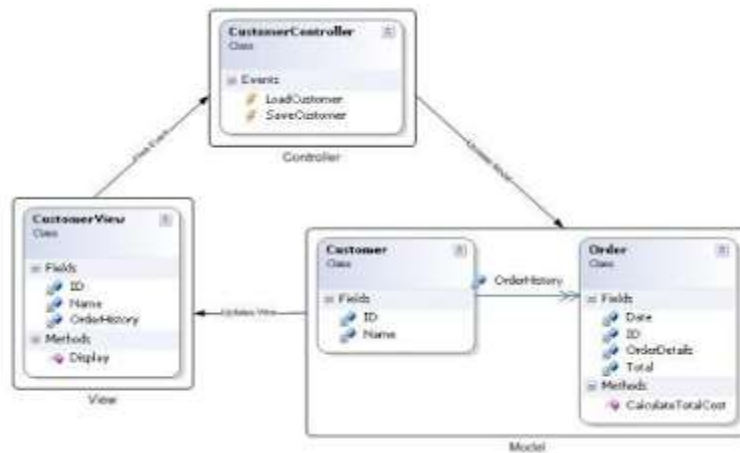
Durability:

This property ensures that once the transaction has completed execution, the updates and modifications to the database are stored in and written to disk and they persist even if system failure occurs. The effects of the transaction , thus, are never lost.

5.What is difference between MVC and MVP architectures?

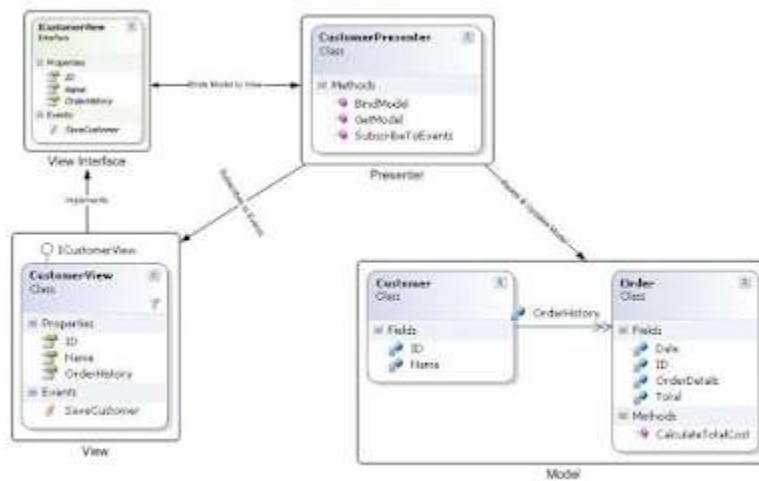
MVC Architecture:

Model-View-Controller



MVP Architecture:

Model-View-Presenter



MVP is just like MVC , only difference is that Controller is replaced with Presenter. View and Presenter are decoupled from each other using an interface. This interface is defined in Presenter .

View implements this interface and communicates with Presenter through this interface. Due to this loose coupling it is easy to mock View during unit testing.

6.What is Swagger and what are annotations used by swagger?

It is a tool for developing APIs specification with the OpenAPI specification.

Swagger vs OpenAPI:

OpenAPI : Specification

Swagger : Tool for implementing the specification

Annotations used by Swagger:

- @Api
- @ApiModel
- @ApiModelProperty
- @ApiParam
- @ApiResponse
- @ApiResponses

7.What are the steps to use Swagger?

Steps to use Swagger are defined below:

- Put dependencies :
 - <dependency>
 - <groupId> io.springfox</groupId>
 - <artifactId>springfox-swagger2</artifactId>
 - </dependency>
- Also add dependency <artifactId>springfox-swagger-ui</artifactId>
- Create a swagger config file [.java file] annotated with @EnableSwagger2 and should have accompanying @Configuration annotation. In this class, create a Docket bean by calling api() method.
- Use @Api on controller class.
- Use @ApiOperation and @ApiResponses on methods.
- Use @ApiModel on entity class.

8.What is a cloud? What are the benefits of using it?

A cloud is actually a collection of web servers [instead of a single server] owned by a 3rd party.

Cloud provides inexpensive ,efficient and flexible alternatives to computers.

Benefits of cloud:

- No need of extra space required to keep all the hardware [Servers, digital storage]
- Companies don't need to buy software or software license for all it's employees. They just pay a small fees to the cloud computing company to let their employees access a suite of software online.
- It also reduces IT problems and costs.

9.What is sharding ? When we need sharding? How to implement MongoDB sharding?

Sharding involves breaking up of one data set into multiple smaller chunks, called logical shards. The logical shards can then be distributed across separate database nodes,referred to as physical shards , which can hold multiple logical shards.

Sharding is a method of splitting and storing a single logical dataset in multiple databases.

Sharding adds more servers to database and automatically balances data and load across various

servers. These databases are called shards.

When and why we need sharding?

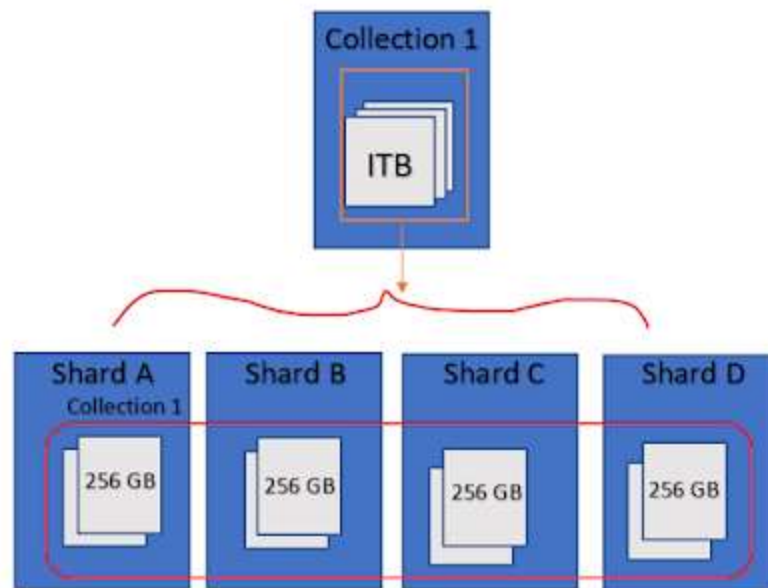
- When the dataset outgrows the storage capacity of a single database instance.
- When a single instance of DB is unable to manage write operations.

How to implement MongoDB sharding?

When deploying sharding, we need to choose a key from a collection and split the data using the key's value.

Task that the key performs:

- Determines document distribution among the different shards in a cluster.



Choosing the correct shard key:

To enhance and optimize the performance, functioning and capability of the DB, we need to choose the correct shard key.

Choosing correct shard key depends upon two factors:

- The schema of the data.
- The way database applications query and perform write operations.

10. How an application is deployed to Amazon EC2 ?

Amazon EC2 offers ability to run applications on cloud.

Steps to deploy application on EC2:

- Launch an EC2 instance and SSH into it. This instance needs to be created first on AWS Console [console.aws.amazon.com]. And we should have certificate to connect to EC2 instance.
- Install Node on EC2, if our app is in angular.
- Copy paste code on EC2 and install dependencies.
- Start server to run.

Another way:

Suppose we have Spring Boot application.

- Build out Spring Boot app in our local computer. Make .jar file.
- Upload this .jar file on S3.
- Create EC2 instance.
- SSH into EC2 from our local computer.
- Now, we are in EC2 instance.
- Now install JDK.
- And using java - .jar file path, we can run our application.

11.What is serverless Architecture?

ServerLess Architecture means: Focus on the application, not the infrastructure.

Serverless is a cloud computing execution model where the cloud provider dynamically manages the allocation and provisioning of servers.

A serverless application runs in a stateless container that are event-triggered and fully managed by the cloud provider.

Pricing is based on the number of executions rather than pre-purchased compute capacity.

Serverless computing is an execution model where the cloud provider is responsible for executing a piece of code by dynamically allocating the resources.

And only charging for the amount of the resources used to run the code. The code typically runs inside stateless compute containers that can be triggered by a variety of events including HTTP requests, database events, file uploads, scheduled events etc.

The code that is sent to the cloud provider is in the form of a function. Hence serverless is sometimes also referred to as "Functions as a Service".

Following are the FaaS offerings of the major cloud providers:

- AWS Lambdas
- Microsoft Azure : Azure Functions
- Google Cloud : Cloud Functions
- IBM Openwhisk
- Auth0 Webtask

Interview questions on Microservices

41 | Praveen Oruganti

Blog: <https://praveenorugantitech.blogspot.com>

Facebook Group: <https://www.facebook.com/groups/268426377837151>

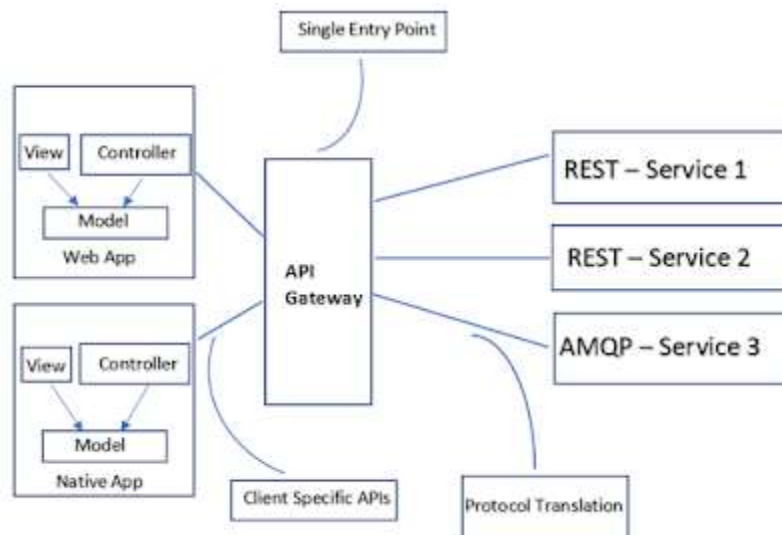
Github repo: <https://github.com/praveenoruganti>

Email : praveenorugantitech@gmail.com

1.What is the API Gateway pattern?

API Gateway is the entry point for all clients. The API Gateway handles requests in one of two ways:

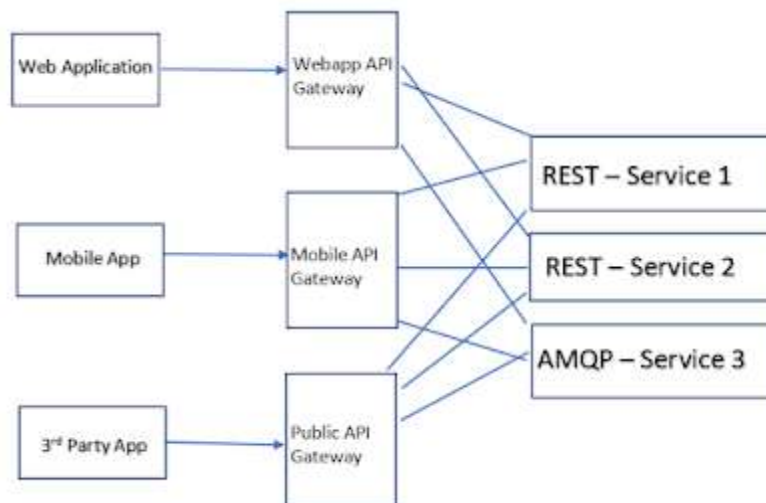
- Some requests are simply proxied/routed to the appropriate service.
- It handles other requests by fanning out to multiple services.



Rather than provide a one-size-fits-all style API, the API gateway can expose a different API for each client.

Variation : Backends for Frontends:

A variation of API Gateway pattern is the Backends for Frontends. It defines a separate API Gateway for each kind of client.



In this example, there are 3 different types of APIs for 3 different types of clients.

Benefits of using API Gateway Pattern:

- Separates the clients from how the application is divided into microservices.
- Frees the clients from determining the location of microservices.
- Providing the optimal API for each client.
- It reduces the number of requests/round trips. e.g. : The API gateway enables the clients to retrieve data from multiple services with a single round-trip. Fewer round-trips means less requests from client and less overhead.
- Simplifies the client by moving the logic for calling multiple services from client to API Gateway.

Limitations of using API Gateway Pattern:

- Increased complexity: API Gateway needs to be developed, deployed and managed.
- Increased response time due to the additional network hop through the API Gateway.

Security using API Gateway's Access Token Pattern:

Problem: How to implement the identity of the requestor to the services that handle the request?

Solution:

The API Gateway authenticates the request and passes an access token [e.g. JSON Web Token] that securely identifies the requestor in each request to the services. A service can include the access token in requests it makes to other services.

This pattern has following benefits:

- The identity of the requestor is securely passed around the system.
- Services can verify that the requestor is authorized to perform the operation.

2. Client Side Discovery/Server Side Discovery

As, we know, services need to call one another. In a monolithic application, services invoke each other through language-level method or procedure calls.

In a traditional distributed system deployment , services run at fixed, well known locations, so can easily call one another using REST API or some RPC mechanism.

However a microservice-based application runs in a virtualized or containerized environments where the number of instances of a service and its locations change dynamically.

So, the **problem** is:

How does the client service - API Gateway or another service - discover the location of a service instance?

Forces:

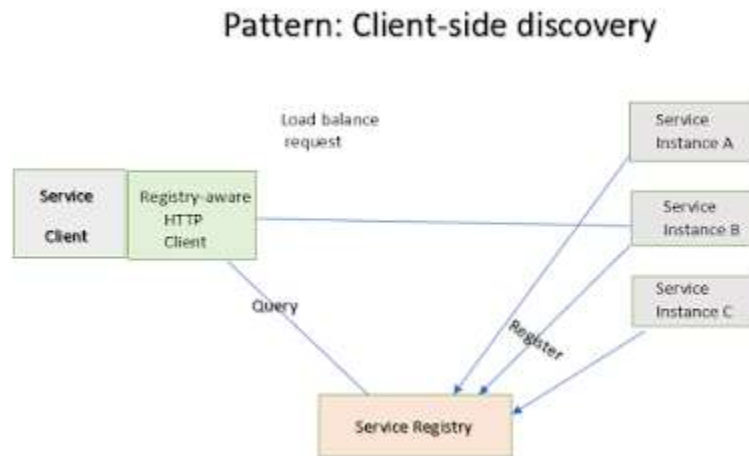
- Each instance of a service exposes a remote API such as HTTP/REST at a particular location.
- The number of service instances and their location change dynamically.

- Virtual machines and containers are usually assigned dynamic IP addresses.

Solution:

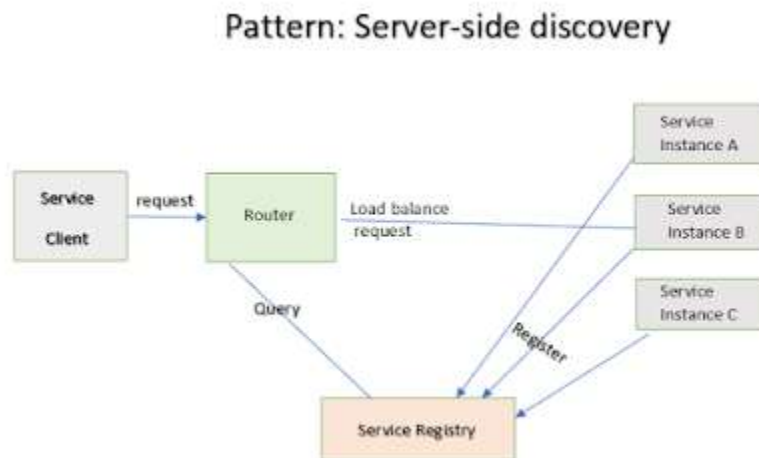
Client Side Service Discovery:

When making a request to a service, the client obtains the location of the service instance by querying a service registry, which knows the locations of all service instances.



So, in Client Side Service discovery, Client service - API Gateway send request to Service registry to find a service and then load balance the request to any instance of that service.

Server Side Service Discovery:



In server side service discovery, client service - API Gateway sends request to a Router a.k.a. Load Balancer which in turn send a query to service registry ,finds the service and propagates request to appropriate instance.

Benefits of server side service discovery:

- Compared to client side discovery, the client code is simpler since it does not have to deal with discovery. Instead, a client just makes a request to the router.
- Some cloud environments provide this functionality , e.g.: AWS Elastic Load Balancer.

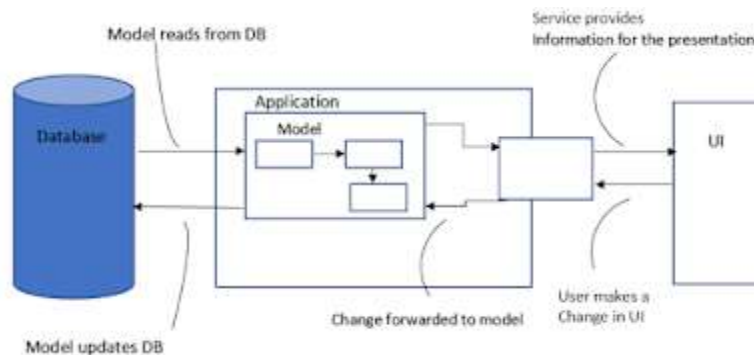
3.CQRS

CQRS : Command Query Responsibility Segregation

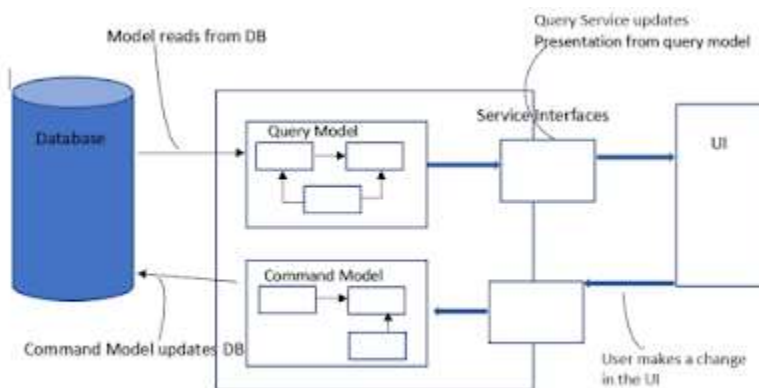
This pattern says, we can use a different model to update information than the model we use to read information.

For some situations, this separation can be valuable, but for more systems, CQRS adds risky complexity.

Non-CQRS Model:



CQRS Model :



In this we have used different models for read (Query) and update (Command) operations.

By different models we mean different object models, probably running in different logical processes, perhaps on separate hardware.

A web example would see a user looking at a web page that's rendered using the Query model. If the user initiate a change, that change is routed to the separate command model for processing, the resulting

change is communicated to the query model to render the updated state.

In this case, there is room for considerable variation here:

The in-memory models may share the same database, in which case, the database acts as the communication between the two models. However, they may also use separate databases, effectively making the query-side database into a real time Reporting database.

In this case, there needs to be some communication mechanism between the two models of their databases.

The two models might not be separate object models, it could be that the same objects have different interfaces for their command side and their query side.

CQRS naturally fits with some other architectural patterns:

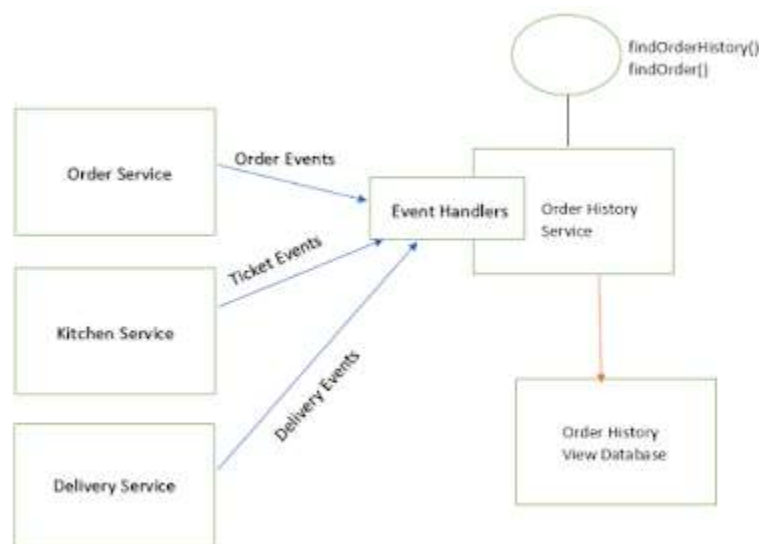
- As we move away from a single representation that we interact with via CRUD, we can easily move to a task-based UI.
- CQRS fits well with event-based programming models. It's common to see CQRS system split into separate services communicating with event collaboration. This allows these services to take advantage of Event Sourcing.
- Having separate models raises questions about how hard to keep those models consistent, which raises the likelihood of using eventual consistency.

It also solves a problem:

How to implement a query that retrieves data from multiple services in a microservice architecture?

Solution:

Define a view database that is a read-only replica which is designed to support that query. The application keeps the replica up-to-date by subscribing to Domain events published by the service that own the data.



When to use CQRS:

46 | Praveen Oruganti

Blog: <https://praveenorugantitech.blogspot.com>

Facebook Group: <https://www.facebook.com/groups/268426377837151>

Github repo: <https://github.com/praveenoruganti>

Email: praveenorugantitech@gmail.com

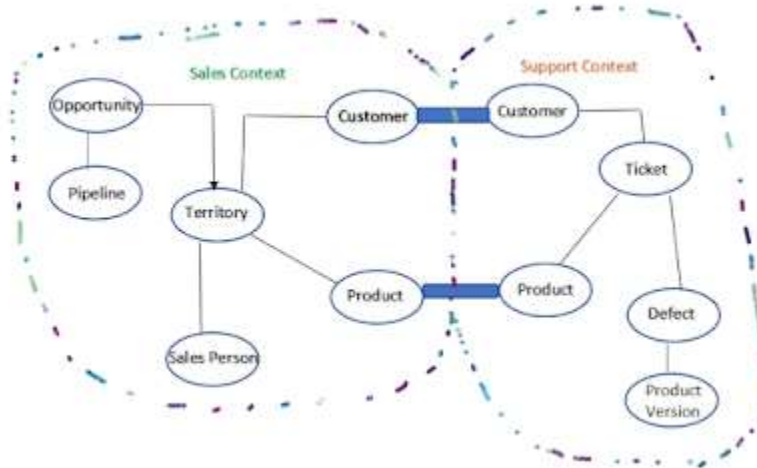
In particular, CQRS should only be used on specific portions of a system (a Bounded Context in DDD lingo) and not the system as a whole.

BoundedContext:

Bounded Context is a central pattern in Domain-Driven Design.

Domain Driven Design deals with large models by dividing them into different bounded contexts and being explicit about their relationships.

e.g. of Bounded Context:



4.Circuit Breaker

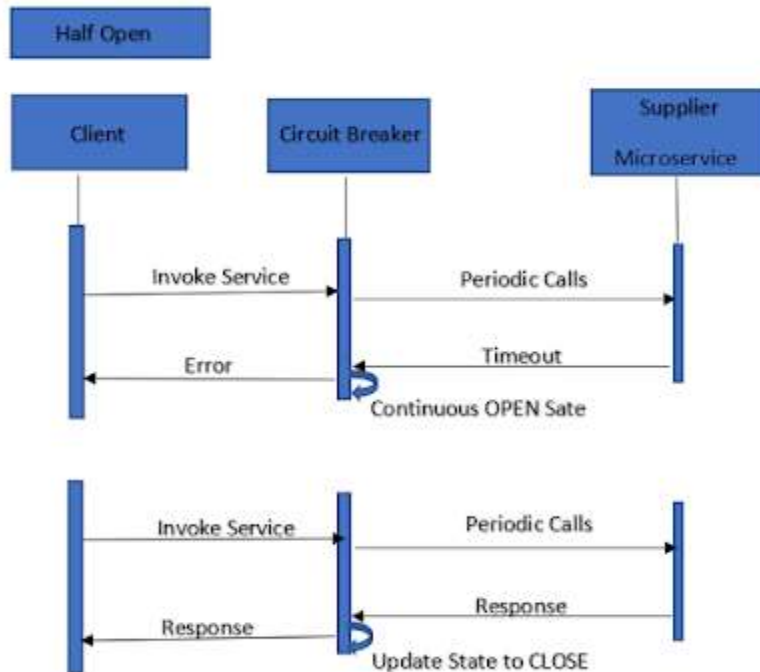
It is very important to prepare system in case of partial failure, especially for a microservice-based architectures, where there are many applications running in separate processes.

A single request from the client point of view might be forwarded through many different services and it is possible that one of these services is down because of a failure, maintenance or just might be overloaded, which causes an extremely slow response to client requests coming into the system.

There are several best practices for dealing with failures and errors:

- The first practice recommends that we should always set network connect and read timeouts to avoid waiting too long for the response.
- The second approach is about limiting the number of accepted requests if a service fails or responses take too long. In this case, there is no sense in sending additional requests by the client.

The last two patterns are closely connected to each other. Here I'm thinking about the circuit breaker pattern and fallback.



Few points about the Circuit breaker diagram:

- Initially the circuit is closed. Means the service is working fine , accepting requests and returning response.
- When few requests get failed, circuit gets tripped and enter into Open state and doesn't allow any further request to go to that non-responding service.
- In between, after some timeout , circuit breaker allows few requests to the non-responding service to check the status of that service. This state is called Half-open state as it allows only few requests to go further. If service responds well, then circuit gets into Closed state , else it will get into Open state again.

Circuit Breaker Pattern:

The major assumption of this approach relies on monitoring successful and failed requests. If too many requests fail or services take too long to respond, the configured circuit breaker is tripped and all further requests are rejected.

The most popular implementation of Circuit Breaker pattern is available in Netflix Hystrix which is used by many java frameworks like Spring Cloud or Apache Camel.

Implementation of a circuit breaker with Spring Cloud Netflix is quite simple. In the main class, it can be **enabled with one annotation**:

```
@SpringBootApplication
@EnableFeignClients
@EnableCircuitBreaker
public class Application{
```



```

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

```

And also need to add dependency :

In gradle: Use :

compile group: 'org.springframework.cloud', name: 'spring-cloud-starter-hystrix', version: '1.4.7.RELEASE'

For fallback mechanism, we can use FeignClient like:

Our service A is going to interact with service B , but circuit breaker is tripped and requests to service B are rejected.

So , we need to call any fallback method for providing the response.

A.java:

```

@FeignClient(name = "A", url = " URL or endpoint of service B" , fallback = A.Fallback.class)
public interface A{

```

```

    @GetMapping(value =("/{id}?app_id={app_id}")
    ResponseEntity<String> getInfo( @PathVariable("id") String id);

```

```

class Fallback implements A{

```

```

    @Override
    public ResponseEntity<String> getInfo(String id){

        // Send back any info.
    }

```

```

}

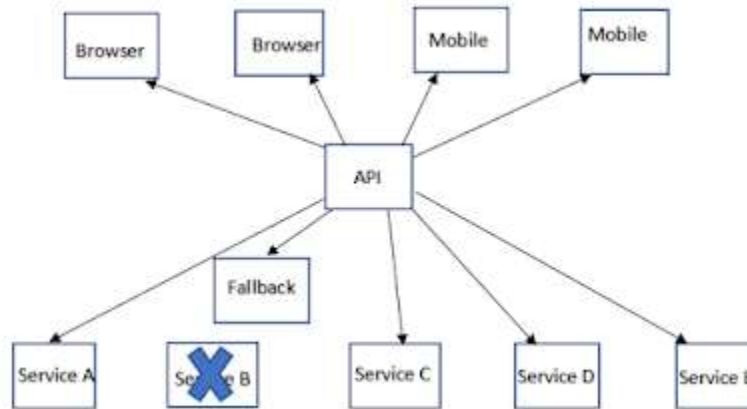
```

```

}

```

A pictorial diagram is like:

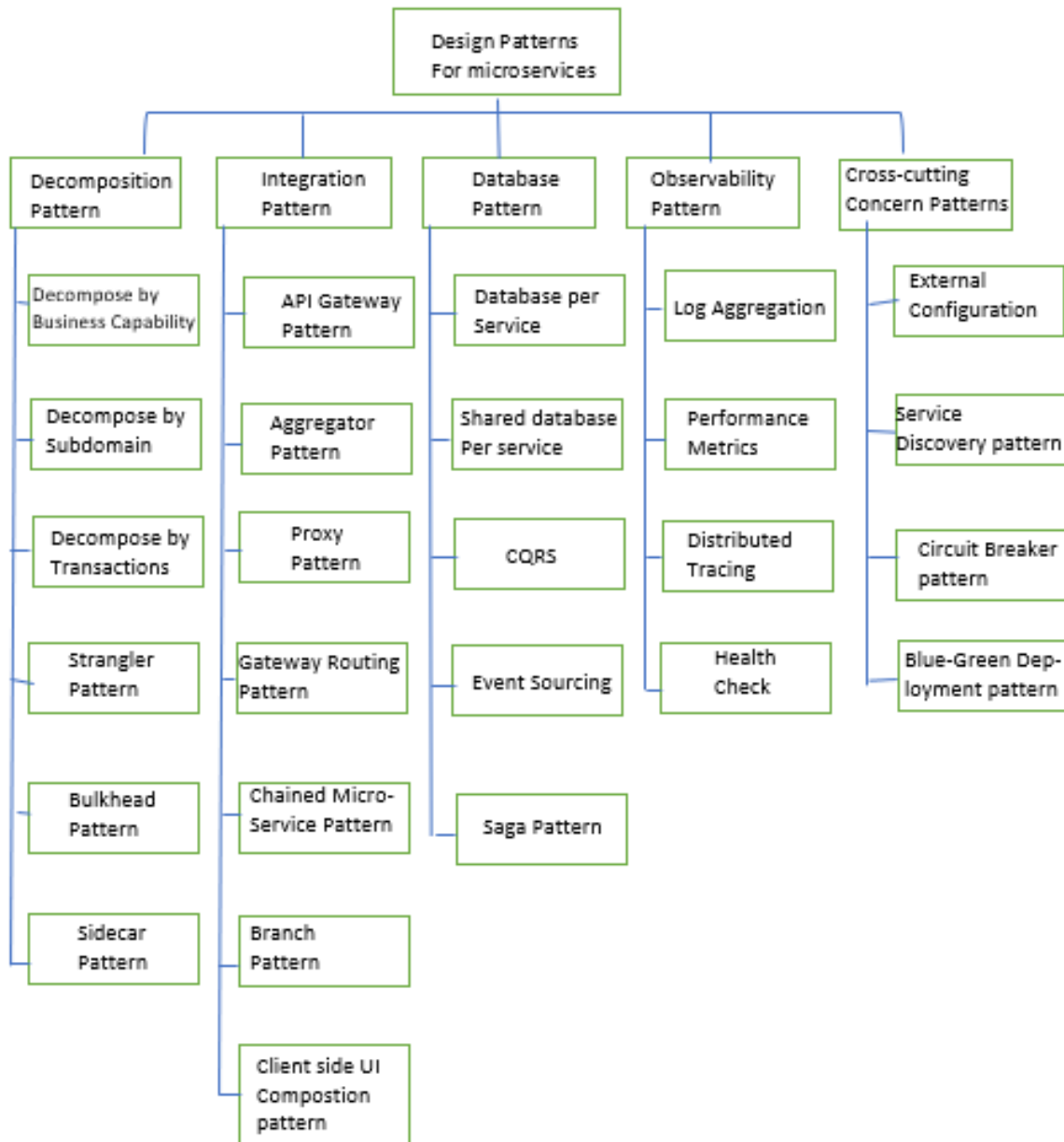


5.What principles microservice architectures has been built upon?

- Scalability
- Availability
- Resiliency
- Flexibility
- Independent, autonomous
- Decentralized governance
- Failure Isolation
- Auto-Provisioning
- Continuous delivery through DevOps

Adhering to the above principles brings several challenges and issues while bring our system to live. These challenges can be overcome by using correct and matching design pattern.

6.What types of design patterns are there for microservices?



7.Explain in detail Decomposition patterns.

- **Decompose by Business capability:** This pattern is all about making services loosely coupled and applying the single responsibility principle. It decomposes by business capabilities. Means define services corresponding to business capability. It is something that a business does in order to generate values. A business capability often corresponds to a business object, e.g.:
 - Order management is responsible for orders
 - Customer management is responsible for customers
- **Decompose by Subdomain:** This pattern is all about defining services corresponding to Domain-Driven-Design [DDD] subdomains. DDD refers to the application's problem space - the business

- as the domain. A domain consists of multiple subdomains. Each subdomain corresponds to a different part of the business. Subdomains can be classified as follows:

- Core
- Supporting
- Generic

The subdomains of Order management consists of:

- product catalog service
- Inventory management service
- Order management services
- Delivery management services

- **Decompose by transactions/Two-Phase commit pattern** : This pattern can decompose services over transactions. There will be multiple transactions in a system. One of the important participants in a distributed transaction is the transaction coordinator. The distributed transaction consists of two steps:

- Prepare phase
- Commit phase

- **Sidecar Pattern**: This pattern deploys components of an application into a separate processor container to provide isolation and encapsulation. This pattern can also enable applications to be composed of heterogeneous components and technologies. It is called sidecar as it resembles to a sidecar of a bike. In this pattern, a sidecar is attached to an application and provides supporting features for the application.

8.Explain in detail Integration patterns.

- **API Gateway Pattern**: When an application is broken down into multiple microservices, there are a few concerns that need to be considered:

- There are multiple calls for different microservices from different channels
- There is a need for handling different type of protocols
- Different consumer might need a different format of the responses.

An API gateway helps to address many of the concerns raised by the microservice implementation, not limited to the ones above.

- An API gateway is the single point of entry for any microservice calls.
- It can work as a proxy service to route a request to the concerned microservice.
- It can aggregate results to send back to the user.
- This solution can create a fine-grained API for each specific type of client.
- It can also offload the authentication/authorization responsibility of the microservice.

- **Aggregator Pattern**: This pattern helps aggregate the data from different services and then send the final response to the client. This can be done in two ways:

- A composite microservice will make calls to all the required microservices, consolidate the data and transform the data before sending back.

- An API gateway can also partition the request to multiple microservices and aggregate the data before sending it to the consumer. An API gateway can have different modules:

- Mobile API
- Browser API
- Public API

9.Explain in detail the database patterns, you have worked upon.

To define the database architecture for microservices, we need to consider the below points:

- Services must be loosely coupled.They can be developed, deployed and scaled independently.
- Business transactions may enforce invariants that span multiple services.
- Some business transactions need to query data that is owned by multiple services.
- Databases must be sometimes replicated and shared in order to scale.
- Different services have different data storage requirements.
- **Database per service:** To solve the above concerns, one database per service must be designed. It must be private to that service only. It should be accessed by the microservice API only. It cannot be accessed by other services directly. e.g.: for relational database, we can use private-tables-per-service, schema-per-service or database-server-per-service.
- **Shared database per service:** This pattern is useful when we have an application which is monolith and we try to break it into microservices.
- **Command Query Responsibility Segregation [CQRS]:** Once we implement database per service, there is a requirement to query, which requires joint data from multiple services, it's not possible. CQRS suggests splitting the application into two parts: the command side and the query side.
 - The command side handles the create, update and delete requests.
 - The query side handles the query part by using the materialized views.

The event sourcing pattern is generally used along with it to create events for any data change. Materialized views are kept updated by subscribing to the stream of events

- **Event Sourcing Pattern:** This pattern defines an approach to handling operations on data that is driven by a sequence of events , each of which is recorded in an append-only store. Application code sends a series of events that imperatively describe each action that has occurred on the data to the event store, where they're persisted. Each event represents a set of changes to the data.
- **Saga Pattern:** When each service has it's own database and a business transaction spans multiple services, how do we ensure data consistency across services? Each request has a compensating request that is executed when the request fails. It can be implemented in two ways:
 - **Choreography:** In microservice choreography, each microservice performs their actions independently. It does not require any instructions. It is like the decentralized way of broadcasting data known as events. The service which are interested in those events , will use it and perform actions. It is like an asynchronous approach.
 - **Orchestration:** In the microservice orchestration, the orchestration handles all the microservice interactions. It transmits events and responds to it. The microservice orchestration is more like a centralized service. It calls one service and waits for the response before calling the next service. This follows a request/response type paradigm.

Technical Architect interview in Incedo

1.What is database sharding? Why we need database sharding?

Sharding involves breaking up one's data into two or more smaller chunks, called logical shards. The logical shards are then distributed across separate **database** nodes, referred to as physical shards, which **can** hold multiple logical shards.

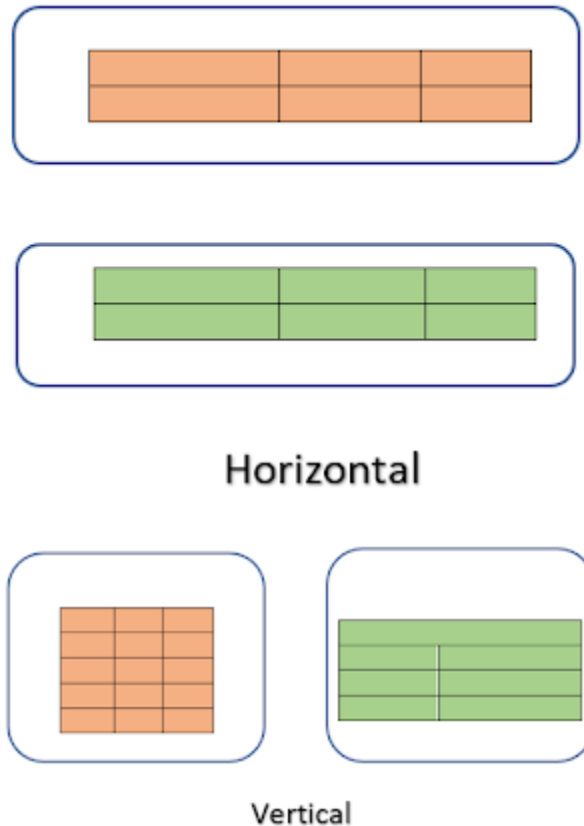
Sharding is a method of splitting and storing a single logical dataset in multiple databases.

Sharding adds more servers to a database and automatically balances data and load across various servers. These databases are called shards.

Sharding is also referred as **horizontal partitioning**. The distinction of **horizontal** vs **vertical** comes from the traditional tabular view of a database.

2.What are the types of database sharding?

A database can be split vertically — storing different tables & columns in a separate database, or horizontally — storing rows of a same table in multiple database nodes.



Example of Vertical partitioning:

```
fetch_user_data(user_id) -> db["USER"].fetch(user_id)
fetch_photo(photo_id) -> db["PHOTO"].fetch(photo_id)
```

54 | Praveen Oruganti

Blog: <https://praveenorugantitech.blogspot.com>

Facebook Group: <https://www.facebook.com/groups/268426377837151>

Github repo: <https://github.com/praveenoruganti>

Email : praveenorugantitech@gmail.com

Example of Horizontal partitioning:

```
fetch_user_data(user_id) -> user_db[user_id % 2].fetch(user_id)
```

Vertical sharding is implemented at application level – A piece of code routing reads and writes to a designated database.

Natively sharded DB's are : Cassandra, MongoDB

Non-sharded DB's are : Sqlite, Memcached etc.

When we need to do sharding?

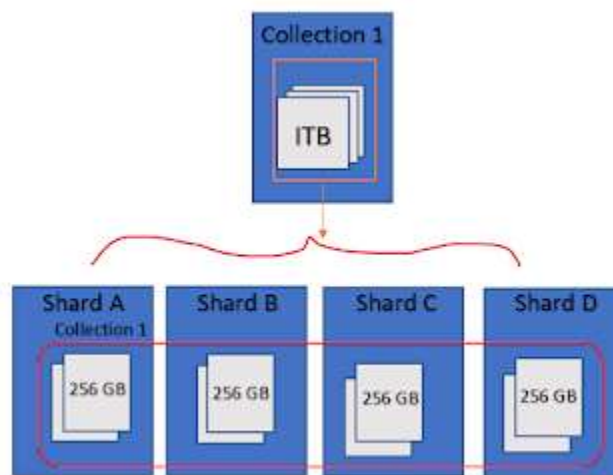
- When the data set outgrows the storage capacity of a single MongoDB instance.
- When a single MongoDB instance is unable to manage write operations.

3.How to implement sharding in MongoDB?

When deploying sharding, we need to choose a key from a collection and split the data using the key's value.

Task that the key performs:

- Determines document distribution among the different shards in a cluster.



Choosing the correct shard key:

To enhance and optimize the performance, functioning and capability of the DB, we need to choose the correct shard key.

Choosing correct shard key depends on two factors:

- The schema of the data
- The way database applications query and perform write operations.

Using range-based Shard key:

In range-based sharding, MongoDB divides data sets into different ranges based on the values of shard keys. In range-based sharding, documents having "close" shard key values reside in the same chunk and shard.



Data distribution in range-based partitioning can be uneven, which may negate some benefits of sharding.

For example, if a shard key field size increases linearly, such as time, then all requests for a given time range will map to the same chunk and shard. In such cases, a small set of shards may receive most of the requests and system would fail to scale.

Hash-based sharding:

In this, MongoDB first calculates the hash of a field's value and then creates chunks using those values. In this sharding, collections in a cluster are randomly distributed.

No real schema is enforced:

- We can have different fields in every document if we want to.
 - No single key as in other databases:
- o But we can create indices on any fields we want, or even combinations of fields.
 - o If we want to shard, then we must do so on some index.

4.What is the use of Amazon EC2? What are the steps to deploy on EC2?

Amazon EC2 : Amazon Elastic Compute Cloud

It offers ability to run applications on the public cloud.

It eliminates investment for hardware. There is no need to maintain the hardware. We can use EC2 to launch as many virtual servers as we need.

Steps to deploy on EC2:

- Launch an EC2 instance and SSH into it. **Note:** This instance needs to be created first on AWS console[console.aws.amazon.com]. And we should also have certificate to connect to EC2 instance.
- Install Node on EC2 instance, if our app is in Angular.
- Copy paste code on EC2 instance and install dependencies.

. Start server to run.
OR:

- Build out Spring boot app in our own computer. Make .jar file.
- Upload this .jar on S3
- Create EC2 instance.
- SSH into it from our computer.
- Now, we are in EC2 instance.
- We can install JDK now.
- And using java - .jar file path, we can run our application.

5.How will we use Amazon S3?

For using S3, we need to first choose S3 from AWS console and then we need to create a bucket in that for storing our files.

After creating the bucket, we need to click on the upload option and select jar file or any file from our computer and upload it to S3.

While uploading file to S3, we need to provide it some access level so that we can download it from S3 to EC2 instance.

So before that we need to make EC2 instance up and running.

And then we need to connect from local to EC2 instance using private key that we have. For connecting to EC2, we need to do SSH login [from terminal] using some certificate and using some SSH command.

Now we need to install java on EC2 [As , it is not there by default]. After that we can download our jar file from S3 to EC2 instance.

Now just run this jar using java -jar <filename> to run the springboot application.

6.What are the best Code Review Practices?

Clean Code

- Use intention-revealing names
- Use Solution-Problem domain names
- Classes should be small
- Functions should be small
- Functions should do one thing
- Don't repeat yourself(Avoid duplication)
- Explain yourself in code : Comments
- Use Exceptions rather than return codes
- Don't return Null

Security

- Make class final if not being used for inheritance
- Avoid duplication of code
- Minimize the accessibility of classes and members
- Document security related information
- Input into a system should be checked for valid data size and range
- Release resources (Streams, Connections) in all cases
- Purge sensitive information from exceptions

57 | Praveen Oruganti

Blog: <https://praveenorugantitech.blogspot.com>

Facebook Group: <https://www.facebook.com/groups/268426377837151>

Github repo: <https://github.com/praveenoruganti>

Email : praveenorugantitech@gmail.com

- Don't log highly sensitive information
- Avoid dynamic SQL, use prepared statement
- Limit the accessibility of packages, classes, interfaces, methods and fields
- Avoid exposing constructors of sensitive classes
- Avoid serialization of sensitive classes
- Only use JNI when necessary

Performance

- Avoid excessive synchronization
- Keep synchronized sections small
- Beware the performance of String concatenations
- Avoid creating unnecessary objects

General

- Don't ignore exceptions
- Return empty arrays or collections , not nulls
- In public classes, use accessor methods not public fields
- Avoid finalizers
- Refer to objects by their interfaces
- Always override toString()
- Document thread safety
- Use marker interfaces to define types

Static Code Analysis

- Check static code analyzer report for the classes added/modified

7.What are the types of Http Error codes?

There are 5 types of Error codes:

1XX Informational:

100: Continue

2XX Success:

200 : OK
 201 : Created
 202 : Accepted
 204 : No Content

3XX Redirection:

4XX Client Error:

400 : Bad Request
 401 : Unauthorized
 402 : Payment Required
 403 : Forbidden
 404 : Not Found

5XX Server Error:

58 | Praveen Oruganti

Blog: <https://praveenorugantitech.blogspot.com>

Facebook Group: <https://www.facebook.com/groups/268426377837151>

Github repo: <https://github.com/praveenoruganti>

Email : praveenorugantitech@gmail.com

500 : Internal Server Error
501 : Not Implemented
502 : Bad Gateway
503 : Service Unavailable [Website's server is simply not available]

8. What are the Asymptotic Notations?

To calculate running time complexity of an algorithm, we use following asymptotic notations:

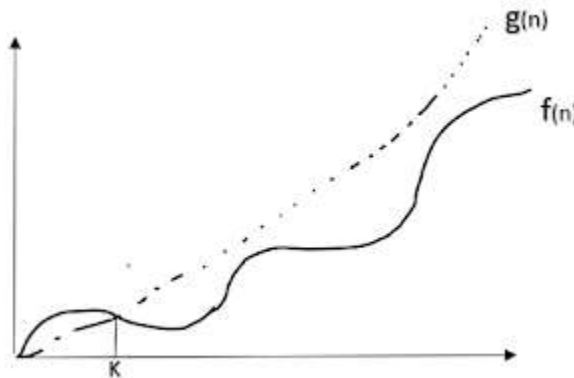
 Notation

 Notation

 Notation

Big Oh Notation O:

The notation $O(n)$ is the formal way to express the upper bound of an algorithm's running time. It measures the worst case time complexity or the longest amount of time an algorithm can possibly take to complete.

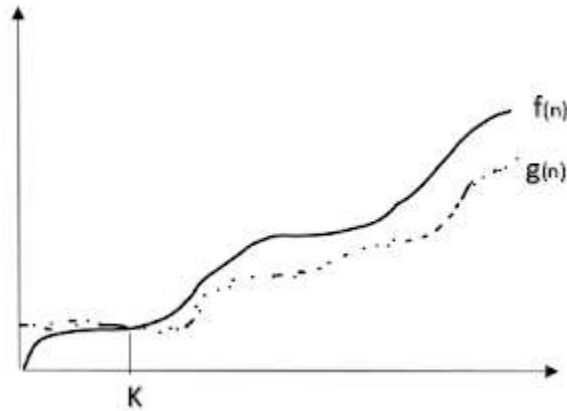


For example, for a function $f(n)$

$O(f(n)) = \{ g(n) : \text{there exists } c > 0 \text{ and } n_0 \text{ such that } f(n) \leq c \cdot g(n) \text{ for all } n > n_0. \}$

Omega Notation:

The Omega notation is the formal way to express the lower bound of an algorithm's running time. It measures the best case time complexity or the best amount of time an algorithm can possibly take to complete.

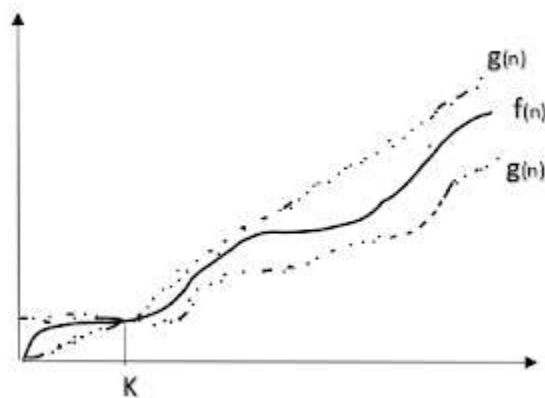


For example, for a function $f(n)$

$$O(f(n)) = \{ g(n) : \text{there exists } c > 0 \text{ and } n_0 \text{ such that } g(n) \leq c \cdot f(n) \text{ for all } n > n_0. \}$$

Theta Notation:

The notation theta (n) is the formal way to express both the lower bound and upper bound of an algorithm's running time. It is represented as follows:

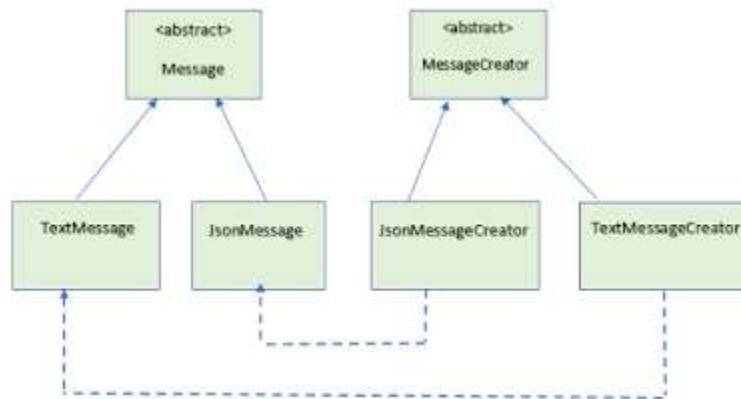


9.Explain when and why to use Factory Design Pattern.

Factory Design Pattern is used in following scenarios:

- When we need to separate the logic of object creation from our other code or client code.
- Whenever we need the ability to add more subclasses to the system and modified them without changing the client code.
- We let subclasses to decide which object to instantiate by overriding the factory method.

UML diagram for Factory Design Pattern is :



10.Explain when and why to use Abstract Factory Design Pattern.

Whenever we have to work with two or more objects which work together forming a set/kit and there can be multiple sets or kits that can be created by client code, then we will use Abstract Factory Design Pattern.

Lets take an example of a War Game:

Medieval:

- Land Unit
- Naval Unit

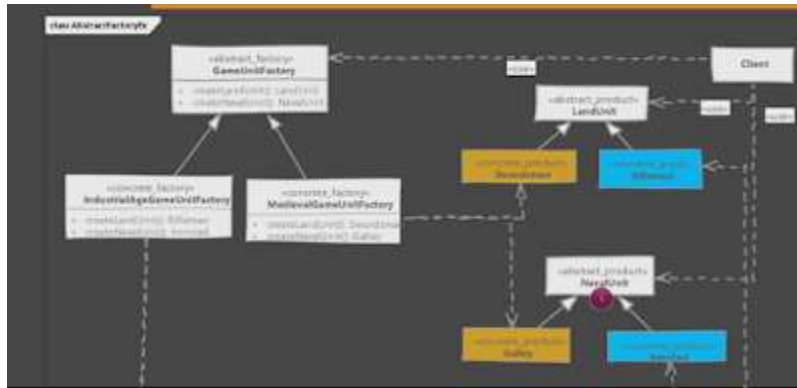
Industrial:

- Land Unit
- Naval Unit

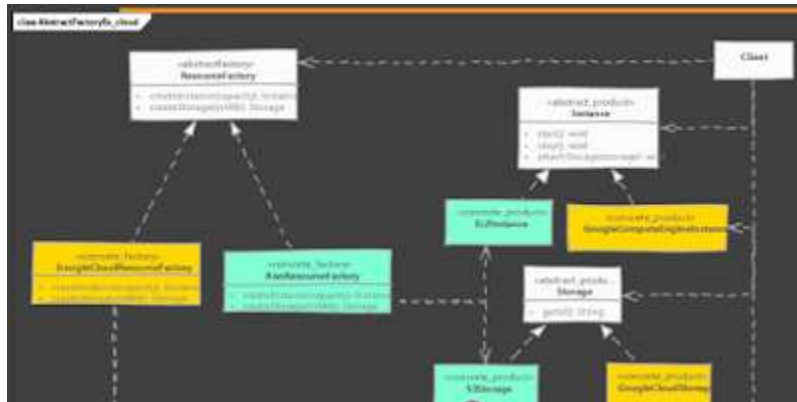
Here , we have two types of objects Land Unit and Naval Unit under Medieval age. And also two types of objects under Industrial age.

So, we can take two factories : Medieval Age Factory and Industrial Age Factory.

UML Diagram for above scenario is :



Another example we can take is :



11. When and why to use Generic class? Explain with example. Can we use generics with array?

A **generic type** is a type with formal type parameters.

e.g.:

```
interface Collection<E>{
    public void add(E e);
    public Iterator<E> iterator();
}
```

A **parameterized type** is an instantiation of a generic type with actual type arguments.

Example of a parameterized type:

```
Collection<String> coll = new LinkedList<String>();
```

Generic class is used when we need to create similar types of object with similar features.

e.g.:

Suppose , we are developing a database library called Data Access Object (DAO) for a program that manages resources in a university.

We would write a DAO class for managing Students like:

```
public class StudentDAO{

    public void save(Student st){
        // some code here ....
    }

    public Student get(long id){

        // Some code here ...
    }

}
```

This looks fine. But if we need to persist Professor objects to database. Then we would write another DAO class as:

```
public class ProfessorDAO{

    public void save(Professor st){
        // some code here ....
    }

    public Professor get(long id){

        // Some code here ...
    }

}
```

Note: These two classes are same. And if we need to add more entity classes to the system like course, facility etc, then?

So, to avoid such situation ,we write a Generic class:

```
public class GeneralDAO<T>{

    public void save(T entity){

    }

    public T get(long id){

    }

}
```

Here T is called type parameter of GeneralDAO class. T stands for any type. The following code illustrates how to use this generic class:

```
GeneralDAO<Student> studentDAO = new GeneralDAO<Student>();
```

63 | Praveen Oruganti

Blog: <https://praveenorugantitech.blogspot.com>

Facebook Group: <https://www.facebook.com/groups/268426377837151>

Github repo: <https://github.com/praveenoruganti>

Email : praveenorugantitech@gmail.com

```
Student student = new Student();
```

```
studentDAO.save(student);
```

Note: Type parameters are never added in names of methods and constructors. They are only added in names of classes and interfaces.

12.What is Type Erasure?

Generics provide compile time type safety and ensures we only insert correct type in Collection and avoids ClassCastException.

Generics in java are implemented using Type Erasures.

It is a process of enforcing type constraints only at compile time and discarding the element type information at runtime.

There are two types of Type Erasures:

- Class Type Erasure
- Method Type Erasure

e.g.:

```
public static <E> boolean containsElement(E[] elements , E element){  
  
    for(E e : elements){  
  
        if(e.equals(element)){  
            return true;  
        }  
    }  
    return false;  
}
```

When compiled, the unbound type E gets replaced with an actual type of Object as:

```
public static boolean containsElement(Object[] elements , Object element){  
  
    for(Object e : elements){  
  
        if(e.equals(element)){  
            return true;  
        }  
    }  
    return false;  
}
```

The compiler ensures type safety of our code and prevents runtime errors.

13.Explain Exception handling in java.

Types of Exceptions:

64 | Praveen Oruganti

Blog: <https://praveenorugantitech.blogspot.com>

Facebook Group: <https://www.facebook.com/groups/268426377837151>

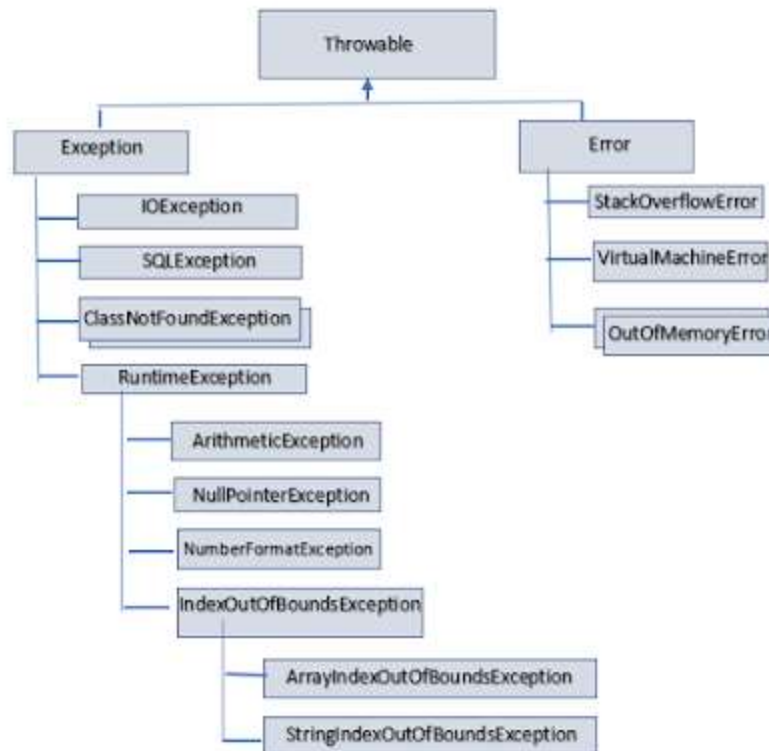
Github repo: <https://github.com/praveenoruganti>

Email : praveenorugantitech@gmail.com

- Checked Exceptions
- Unchecked Exceptions
- Error

Checked Exception: Exceptions that directly inherit from Throwable class except RuntimeException and Error are known as checked exceptions.

Unchecked Exceptions: The classes which inherit RuntimeException are known as unchecked exceptions.



14. What is static import in java? How to use it?

Static import allows developers to access static members of a class directly. There is no need to qualify it by the class name.

Example:

```

import static java.lang.System.*;

class StaticImport{

    public static void main(String[] args){

        out.println("Ok"); // No need to use System.out
        out.println("Java");
    }
}
  
```

```
}  
  
}  
Output
```

Ok
Java

Interview questions on SQL

1.What is the difference between MySQL and SQL Server?

Difference between MySQL and SQL server are as follows:

- MySQL server is owned by Oracle while SQL server is owned and managed by Microsoft
- MySQL supports multiple languages like Perl, Scheme, Tcl, Eiffel etc than SQL server.
- MySQL supports multiple storage engines [InnoDB, MYISAM] which makes MySQL server more flexible than SQL server.
- MySQL blocks the database while backing up the data. And the data restoration is time consuming due to execution of multiple SQL statements. Unlike MySQL, SQL server doesn't block database while backing up the data.
- SQL Server is more secure than MySQL. MySQL allows database files to be accessed and manipulated by other processes at runtime. But SQL server doesn't allow any process to access or manipulate it's database files or binaries.
- MySQL doesn't allow to cancel a query mid-execution. On the other hand, SQL server allows us to cancel a query execution mid-way in the process.

2.What is a Primary key ? How is it different from Unique key?

Primary key is a column or set of columns that uniquely identifies each row in the table. There will be only 1 primary key per table.

Important points about Primary key:

- Null values are not allowed.
- It must contain unique values. Duplicates are not allowed.
- If the primary key contains multiple columns , the combination of values of these columns must be unique.
- When we define a primary key for a table, MySQL automatically creates an index named primary.

Difference between Primary key and Unique key:

- Primary can be only 1 per table. Unique can be many per table.
- Primary key doesn't allow null value. While Unique key allows null value.
- Primary key can be made foreign key in another table. While Unique can not be made foreign key in MySQL but it can be made foreign key in SQL server.
- By default primary key is clustered index and data in the database table is physically organized in the sequence of clustered index. By default, unique key is a unique non-clustered index.

3.What is a join and why to use it?

A join clause is used to access/retrieve data from multiple tables based on the relationship between the

66 | Praveen Oruganti

Blog: <https://praveenorugantitech.blogspot.com>

Facebook Group: <https://www.facebook.com/groups/268426377837151>

Github repo: <https://github.com/praveenoruganti>

Email : praveenorugantitech@gmail.com

fields of the tables.

Keys play a major role when Joins are used.

4.What are different normalization forms? Explain.

There are multiple normalization forms available in SQL:

- 1NF
- 2NF
- 3NF
- BCNF: Boyce Codd Normal Form
- 4NF

Explanation of each normalized form:

1NF Rules:

- Each table cell should contain a single value.
- Each record needs to be unique

2NF Rules:

- Be in 1NF
- Single column primary key

3NF Rules:

- Be in 2NF
- Has no transitive functional dependencies

5.What is an index and types of indexes?

An index is a performance tuning method of faster accessing the records from a table. An index creates an entry for each value and hence it will be faster to retrieve data.

There are following types of indexes:

- Normal Index
- Unique Index
- Clustered Index
- Non-Clustered Index
- BitMap Index
- Composite Index
- B-Tree Index
- Function based index

Unique Index:

67 | Praveen Oruganti

Blog: <https://praveenorugantitech.blogspot.com>

Facebook Group: <https://www.facebook.com/groups/268426377837151>

Github repo: <https://github.com/praveenoruganti>

Email : praveenorugantitech@gmail.com

This index doesn't allow the field to have duplicate value if the column is unique indexed. If a primary key is defined, Unique index can be applied automatically.

Clustered Index:

This index reorders the physical order of the table and searches based on the basis of key values. Each table can only have one clustered index.

Non-Clustered Index:

This index doesn't alter the physical order of the table and maintains a logical order of the data. Each table can have many non-clustered indexes.

6.What is a subquery and what are types of subqueries?

A query within a query is called a subquery. The outer is called a main query and inner query is called a subquery. Subquery is always executed first and the result of subquery is passed to the main query.

There are two types of subqueries:

Correlated Query: It can't be considered as independent query but it can refer the column in a table listed in the FROM of the main query.

NonCorrelated Query: It can be considered as an independent query and the output of subquery are substituted in the main query.

7.What is the difference between char and varchar2?

Both char and varchar2 are used to display character datatype but varchar2 is used for character strings of variable length whereas char is used for strings of fixed length.

e.g.: char(10) can store only 10 characters and will not be able to store a string of any other length whereas varchar(10) can store any length till 10 characters.

8.What is the use of UNION clause?

UNION clause is used to remove duplicate records from the result of a query.

9.Which aggregate functions are there in SQL?

There are multiple aggregate functions in SQL:

- MIN()
- MAX()
- SUM()
- COUNT()
- FIRST()
- LAST()
- AVG()

10.What is a View in SQL and how to create and execute a View ?

A View can be defined as a virtual table that consists of rows and columns from one or more tables.

68 | Praveen Oruganti

Blog: <https://praveenorugantitech.blogspot.com>

Facebook Group: <https://www.facebook.com/groups/268426377837151>

Github repo: <https://github.com/praveenoruganti>

Email : praveenorugantitech@gmail.com

Data in the virtual table is not stored permanently.

Note: Views are stored in system tables : sys.sysobjvalues

Use case or benefits of View:

- Views can hide complexity: If we have a query that requires joining several tables or have complex logic or calculations, we can code all that logic into a view, then select from the View just like you would a table.
- Views can be used as a security mechanism: View can select certain columns and/or rows from a table (or tables) and permissions set on the view instead of the underlying tables. This allows surfacing only the data that a user needs to see.

Creating View:

```
Create View view_name AS  
SELECT column1, column2, ...  
FROM table_name  
WHERE condition;
```

```
Create View [Items Above Average Price] AS  
SELECT ItemName, Price  
From Orders  
where Price > (Select AVG(Price) from Orders);
```

Note: "Items Above Average Price" is the view name.

Querying the above created View:

```
Select * from [Items Above Average Price]
```

11: What are the JDBC statement interfaces?

JDBC statement interfaces are used for accessing the database.

These are of 3 types:

1. Statement
2. PreparedStatement
3. CallableStatement

Statement: Use this for general purpose access to the database. It is useful when we are using static SQL statements at runtime. The Statement interface cannot accept parameters.

PreparedStatement: It is used when we plan to use SQL statements many times. This interface accepts input parameters at runtime.

CallableStatement: It is used when we want to access database stored procedures. This interface also accepts runtime input parameters.

Creating Statement Object:

69 | Praveen Oruganti

Blog: <https://praveenorugantitech.blogspot.com>

Facebook Group: <https://www.facebook.com/groups/268426377837151>

Github repo: <https://github.com/praveenoruganti>

Email: praveenorugantitech@gmail.com

We need to create Statement object before we can use it. We can call createStatement() method on Connection object as follows:

```
try{
    Statement st = Connection.createStatement();
}
catch(Exception e){

}
```

Now, we can call any of the methods given below to execute SQL statement:

- execute()
- executeUpdate()
- executeQuery()

Creating PreparedStatement Object:

```
try{
    String SQL = "update Student set age = ? where id = ?";
    Statement st = Connection.prepareStatement(SQL);
}
catch(Exception e){

}
```

All parameters in JDBC are represented by the ? symbol, which is known as the parameter marker. We must supply values for each parameter before executing the SQL statement.

12.What is the difference between BLOB and CLOB?

Blob and Clob are both known as LOB [Large Object Types].

BLOB: It is a variable length Binary large object string that can be upto 2GB long. Primarily intended to hold non-traditional data such as voice or mixed media.

CLOB: It is a variable length character large object string that can be upto 2GB long. A CLOB can be used to store single byte character strings or multibyte character-based data.

Following are the differences between BLOB and CLOB:

1. The full form of Blob is Binary Large Object. The full form of Clob is Character Large Object.
2. Blob is used to store large binary data. Clob is used to store large textual data.
3. Blob stores values in the form of binary streams. Clob stores values in the form of character streams.
4. Using Clob, we can store files like text files, PDF documents, word documents etc. Using Blob, we can store files like videos, images, gifs and audio files.
5. MySQL supports Blob with the following datatypes:
 1. TinyBlob
 2. Blob

3. MediumBlob
4. LongBlob

MySQL supports Clob with the following datatypes:

1. TinyText
2. Text
3. MediumText
4. LongText

6. In JDBC API, it is represented by java.sql.BLOB interface. While Clob is represented by java.sql.Clob interface.

13.What is the benefit of using PreparedStatement?

PreparedStatements help us prevent SQL injections attack.

In this the query and the data are sent to the database server separately.

While using prepared statement, we first send prepared statement to the DB server and then we send the data by using execute() method.

e.g.:

```
$db-> prepare("Select * from Users where id = ?");
```

This exact query is sent to the server. Then we send the data in second request like:

```
$db->execute($data);
```

If we don't use prepared statement then, SQL injection attack can occur like:

```
$spoiledData = "1; DROP table Users";  
$query = "Select * from Users where id=$spoiledData";
```

will produce a malicious sequence:

```
Select * from Users where id = 1; DROP table Users;
```

14.How to handle indexes in JDBC?

Indexes in a table are pointers to the data which speeds up the retrieval of data from table.

If we use indexes, INSERT and UPDATE statements execute slower whereas SELECT and WHERE get executed faster.

Creating an Index:

```
create index index_name on table_name(column_name);
```

Displaying the Index:

```
Show indexes from table_name;
```

Dropping the index:

```
drop index index_name;
```

Interview questions on Executor Framework

1. Why to use executor framework?

Executor framework is used to decouple command submission from command execution.

2. What is executor framework?

Executor framework gives us the ability to create and manage threads. It provides below functionalities:

- Creating Thread
- Managing Thread
- Task submission and execution

3. What interfaces are there in executor framework?

There are multiple interfaces in executor framework. Let me describe a few of them here:

Executor: A simple interface that contains a method called `execute()` to launch a task specified by a `Runnable` object.

ExecutorService: A sub-interface of `Executor` that adds functionality to manage the lifecycle of the tasks. It also provides a `submit()` method whose overloaded versions can accept a `Runnable` as well as a `Callable` object. `Callable` objects are similar to `Runnable` except that the task specified by a `Callable` object can also return a value.

ScheduledExecutorService: A sub-interface of `ExecutorService`. It adds functionality to schedule the execution of tasks.

4. What are the classes in Executor framework?

Executors: It contains factory methods for creating different kinds of executor services.

5. How to use Executor Service?

```
import java.util.concurrent.Executors;
import java.util.concurrent.ExecutorService;

public class ExecutorExample{

    public static void main(String[] args){

        ExecutorService exeService = Executors.newSingleThreadExecutor();

        Runnable runnable = () -> {

            System.out.println("Executing thread");
        }

        exeService.submit(runnable);
    }
}
```



```
}
```

```
}
```

Note: In the above code, Executor service uses only one thread for executing tasks. If that thread is busy, then the task will wait in a queue until the thread is free to execute it.
This program never exits, because the executor service keeps listening for new tasks until we shut it down explicitly.

6.How to shutdown the executor service?

We can shutdown the executor service using 2 methods:

shutdown(): When shutdown() method is called on an executor service, it stops accepting new tasks, waits for previously submitted tasks to execute and then terminates the executor.

shutdownNow(): This method interrupts the running task and shuts down the executor immediately.

7.How to schedule a task using Executor framework?

```
ScheduledExecutorService exeService = Executors.newScheduledThreadPool(1);
```

```
Runnable task= () ->{
```

```
    System.out.println("Executing thread");  
}
```

```
exeService.schedule(task, 5, TimeUnit.SECONDS);
```

```
exeService.shutdown();
```

Note: If we want to execute the task periodically, we can do so by:

```
exeService.scheduleAtFixedRate(task, 0, 2, TimeUnit.SECONDS);
```

8.Why should we use ThreadPoolExecutor, when we have Executor Framework?

Source code of Executors.newFixedThreadPool() is:

```
public static ExecutorService newFixedThreadPool(int nThreads){  
  
    return new ThreadPoolExecutor(nThreads, nThreads, 0L, TimeUnit.MILLISECONDS,  
new                                     LinkedBlockingQueue<Runnable>());  
  
}
```

This method uses ThreadPoolExecutor class with default configuration as is seen in above code. Now there are scenarios where default configuration is not suitable say instead of LinkedBlockingQueue, a priority queue needs to be used etc.

In such cases, caller can directly work on underlying ThreadPoolExecutor by instantiating it and passing

desired configuration to it.

Note: One advantage of using ThreadPoolExecutor is that we can handle RejectedExecutionException using ThreadPoolExecutor.DiscardPolicy().

9.How the number of threads contained in thread pool is determined when using ThreadPoolExecutor?

The number of threads in the thread pool is determined by these variables:

- corePoolSize
- maximumPoolSize

If less than corePoolSize threads are created in the thread pool when a task is delegated to the thread pool, then a new thread is created, even if idle threads exist in the pool.

If the internal queue of tasks is full and corePoolSize threads or more are running, but less than the maximumPoolSize threads are running, then a new thread is created to execute the task.

10.What is the difference between Fixed thread pool and cached thread pool?

Fixed thread pool allows to create a pool with fixed number of threads.

Cached thread pool creates any number of threads depending upon the tasks provided. If number of tasks passed to ExecutorService is large, then cached thread pool creates too many threads and may increase overhead of the system.

11.What is the difference between submit() and execute() methods in executor framework?

There are multiple differences between submit() and execute() methods in executor framework which are described below:

- execute() method is declared in Executor interface. While submit() method is declared in ExecutorService interface which extends Executor interface.
- execute() method can take only Runnable object as parameter. While submit() has 3 overloaded forms which accept Runnable, Callable, Runnable and result type.
- execute() method returns void. While all forms of submit() method returns Future instance.

Note: A thread can be created only with Runnable instance.

12.When result will be returned by Future.get() , when we call submit(Runnable instance)?

As we know, run() method in Runnable doesn't return anything. So, Future instance in this case will not hold any result and calling get() on Future object will return null upon successful completion.

13.How to execute a collection of tasks at once in Executor Framework?

We can call invokeAll() method from ExecutorService passing a collection of all tasks in it.

Syntax:

```
List<Future<T>> invokeAll(Collection<? extends Callable<T>> tasks)
```

14. Describe the purpose and use case of Fork/Join framework.

Answer:

The fork/join framework allows parallelizing recursive algorithms. The main problem with parallelizing recursive algorithms with ThreadPoolExecutor is that we may quickly run out of threads because each recursive step requires its own thread.

The Fork/Join framework entry point is ForkJoinPool class which is an implementation of ExecutorService. It implements the work-stealing algorithm, where idle threads try to steal work from busy threads.

This allows to spread the calculations among various threads and make progress while using fewer threads than it would require with a usual thread pool.

15. How many ways are there to create a thread and run it?

There are 3 ways to create and run threads in java:

First way:

```
Thread t = new Thread(() -> System.out.println("Running thread using lambda expression"));
t.start();
```

Second way:

```
Thread t = new Thread(new Runnable(){
    public void run(){
        System.out.println("Running thread using Runnable instance");
    }
});
t.start();
```

Third way:

```
Thread t = new Thread(){
    public void run(){
        System.out.println("Running thread using anonymous class");
    }
}
t.start();
```

16. What are the available implementations of ExecutorService interface?

There are mainly 3 implementations of ExecutorService interface.

- ThreadPoolExecutor
- ScheduledThreadPoolExecutor
- ForkJoinPool

17.What special guarantee does the Java Memory Model hold for final fields of a class?

Java Memory Model guarantees that final fields of the class will be initialized before any thread gets hold of the object.

Without this guarantee, an object of the class may be published i.e. become visible to other threads before all the fields of this object are initialized due to reorderings or other optimizations. This could cause racy access to these fields.

Interview Questions on Java 8

1.How to use Default method in java 8?

Java 8 introduced default or "Defender methods" as a new feature which allows developers add new methods definitions in interface without breaking existing functionalities/implementation of these interfaces.

Lets take an example for understanding "How to use it":

```
public interface OldInterface{

    public void existingMethod();

    default public void newDefaultMethod(){

        System.out.println("Code goes here");

    }

}

public class OldImplementation implements OldInterface{

    public void existingMethod(){

    }

}
```

Now create instance of OldImplementation instance and call default method as:

```
OldImplemenattaion oll = new OldImplementation();
oll.newDefaultMethod();
```

Example of Default method added in Java JDK is forEach() which is added in Iterable and Stream interfaces as:

```
public interface Iterable<T>{

    default public void forEach(Consumer<? super T> action){
        for(T t : this)
```

```

        action.accept(t);
    }
}

```

2. Define Multiple inheritance ambiguity problem in Default methods.

Since java class can implement multiple interfaces and each interface can define a default method with same method signature , therefore the inherited methods can conflict with each other.

Let's understand it through an example:

```

public interface InterfaceA{

    public void default method(){

    }

}

public interface InterfaceB{

    public void default method(){

    }

}

public class Implementation implements InterfaceA, InterfaceB {

}

```

The above code will fail to compile.

In order to fix this, we need to provide default method implementation as:

```

public class Implementation implements InterfaceA, InterfaceB {

    public void default method(){

    }

}

```

If we want to invoke default implementation provided by any of the super Interface, then we can do that as:

```

public class Implementation implements InterfaceA, InterfaceB {

    public void default method(){
        Interface.super.method();
    }

}

```

Note: We can choose any default implementation or both as part of our new method.

3.What other method has been introduced in interface in Java 8?

In java 8 , static method has also been introduced. These are called by interface name. They are introduced to increase the degree of cohesion of a design by putting together all related methods in one single place without having to create an object.

For example, see below code:

```
public interface Vehicle{  
  
    static int getPower(){  
  
        return 0;  
    }  
  
}
```

Call it as:

Vehicle.getPower();

4.What is the difference between default method and regular method?

Different between default method and regular method are as follows:

- Default method comes with default keyword by default. Regular method doesn't use any such keyword.
- Default method is defined in interface and an interface doesn't have any state, so default method cannot change the state of the object. While regular method can change the state of the object by changing value of it's parameters.

5:How Lambda is useful in java?

Lambda is a type of functional interface. A functional interface is an interface that has exact one abstract method. e.g. : Runnable, Callable, ActionListener, Comparator etc.

Lets' take an example to understand how Lambdas are useful in java:

```
@FunctionalInterface  
public interface ITrade{  
  
    public boolean check(Trade t);  
}
```

So, above interface has only one abstract method that takes a Trade object.

Lets create a Lambda expression for this interface:

```
ITrade newTradeChecker = (t) -> t.getStatus().equals("new");
```

The real power of Lambdas come when we start creating a multitude of them representing real world behavioral functions as:

```
ITrade bigLambda = (t) -> t.getQuantity() > 13000;
```

```
ITrade smallLambda = (t) -> t.getStatus() < 10000;
```

Now, we have a class `filterTrades` that will filter trades based on which Lambda we have passed for filtering.

```
private List<Trade> filterTrades(ITrade trade, List<Trade> trades){  
    List<Trade> newTrades = new ArrayList<>();  
    for(Trade t : trades){  
        if(trade.check(t)){  
            newTrade.add(t);  
        }  
    }  
}
```

So, in this way, it will behave differently based on Lambda passed to `filterTrades()` method.

6.What are the differences between Lambda and anonymous class?

Below is the list of differences between Lambda and anonymous class:

- **Usage of 'this' keyword:** In anonymous class, this keyword refers to anonymous class itself. While in Lambda, this keyword refers to enclosing class of Lambda.
- **Compilation:** Anonymous class gets compiled to .class file in java. While Lambdas are compiled to private static methods in enclosing class. It uses invokedynamic bytecode instruction to bind this method dynamically.
- Lambdas implement a functional interface, so implements only 1 method. Anonymous class can extend a class or implement any number of interfaces.

7.What are the types of Lambdas?

There are two types of Lambdas in Java 8:

Non-capturing Lambdas: These lambdas only use fields inside their bodies, as:

```
public class NonCapturingLambda{  
  
    public static void main(String[] args){  
  
        Runnable nonCapLambda = () -> System.out.println("NonCapturingLambda");  
        nonCapLambda.run();  
    }  
}
```

```
}  
}
```

Capturing Lambdas: These lambdas access fields outside their bodies, as:

```
public class CapturingLambda{  
  
    public static void main(String[] args){  
  
        String str = "CapturingLambda";  
        Runnable capLambda = () -> System.out.println("CapturingLambda = "+str);  
        capLambda.run();  
    }  
}
```

8.What are the benefits of using Streams?

Streams were added in java 8 as part of java.util.stream package.

There are multiple benefits of using streams in java :

- Using streams, we can process data in declarative way as we do in SQL.
- Using parallel streams, we can use all available cores in the processor and process data parallelly, which improves performance.
- Using streams, we can compose functions and data flows through the functions.

9.Can we use streams with primitives?

Yes, there are two ways to use stream on primitives:

- Use wrapper classes.
- Use primitive types of streams: IntStream, LongStream and DoubleStream.

10.What is the difference between passing int array and string array to Stream.of()?

Stream.of(int[]) gives Stream<int[]>.

Stream.of(String[]) gives Stream<String>

So, when using Stream.of() with int[] array , we get Stream<int[]> and then for getting ints from this stream, we use flatMapToInt(i-> Arrays.stream(i)) to get IntStream and then we can use either map() or forEach().

11.What is a boxed stream in java 8?

If we want to convert stream of objects to collection, we can use:

```
List<String> list = Stream.of("a","b","c","d","e").collect(Collectors.toList());
```

The same process doesn't work on streams of primitives, however.

```
IntStream.of(1,2,3,4,5).collect(Collectors.toList()); // Compilation Error!!
```


To convert a stream of primitives, we must first box the elements in their wrapper class and then collect them. This type of stream is called boxed stream.

```
List<Integer> list = IntStream.of(1,2,3,4,5).boxed().collect(Collectors.toList());
```

```
System.out.println(list);
```

O/P: [1,2,3,4,5]

12.What is the difference between sequential stream and parallel stream?

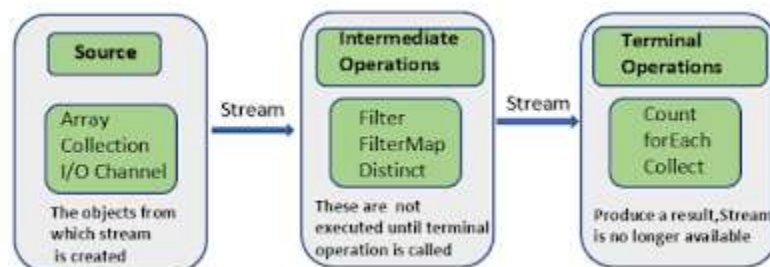
Parallel stream divides the task into many and run them in different threads, using multiple cores of the computer. It uses main thread along with ForkJoinPool.commonPool worker threads. Sequential stream uses a single core. So, it uses only 1 thread that is main thread.

13.What are the characteristics of a stream?

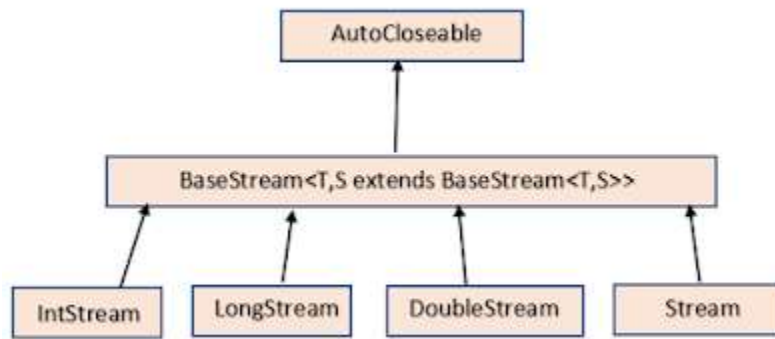
Below is the list of characteristics of streams in java 8:

- **Sequence of elements:** A stream provides a set of elements of specific types in a sequential manner. A stream computes elements on demand. It never stores the elements.
- **Source:** Stream takes Collections, Arrays or I/O resources as input source.
- **Aggregate Operations:** Stream supports aggregate operations like filter, map, limit, reduce, find, match and so on.
- **Pipelining:** Most of the stream operations return stream itself so that their result can be pipelined. These operations are called intermediate operations.
- **Automatic Iterations:** Stream operations do the iterations internally over the source elements provided, in contrast to collections where explicit iteration is required.

14.Draw Stream lifecycle diagram.



15.What is architecture of Steam?



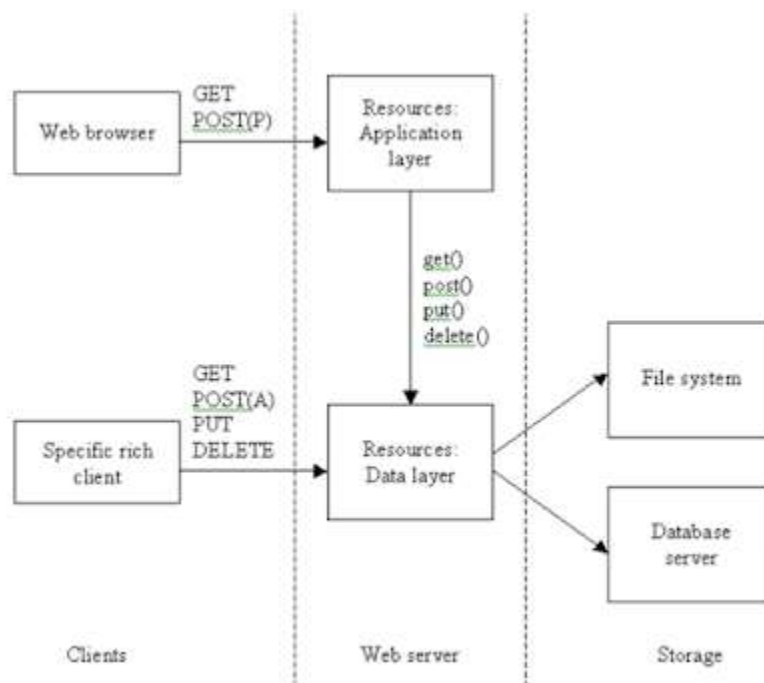
16.How streams are lazy?

Streams are lazy because intermediate operations are not evaluated unless terminal operation is invoked.

Each intermediate operation creates a new stream, stores the provided operation/function and return the new stream. The pipeline accumulates these newly created streams.

Java Interview @ SAP

1.Draw REST Architecture



2.Name core components of HTTP request.

Each HTTP request includes five key elements:

- The verb which indicates HTTP methods such as GET, POST, PUT, DELETE.
- URI: Stands for Uniform Resource Identifier. It is the identifier for the resource on server.
- HTTP version which indicates HTTP version , for example -- HTTP v1.1
- Request Header carries metadata (as key-value pairs) from the HTTP request message. Metadata could be a client type, the format that client supports, message body format and cache settings.
- Request Body: It indicates the message content or resource representation.

3.Explain the term statelessness with respect to RESTFUL Web service.

In REST, ST itself defines State Transfer and statelessness means complete isolation. This means, the state of the client's application is never stored on the server.

In this process, the client sends all the information that is required for the server to fulfill the HTTP request that has been sent. Thus every client request and the response is independent of the other with complete assurance of providing required information.

4.State the core components of an HTTP response.

Every HTTP response includes 4 key elements:

- Status/Response code: Indicates server status for the resource present in HTTP request e.g. 404 means, resource not found and 200 means , response is OK.
- HTTP Version
- Response Header: Contains metadata for the HTTP response in the form of key-value pairs. e.g. content length, content type, response date and server type
- Response Body: Indicates response message content or resource representation.

5.What is cloud and what are the benefits of Cloud Computing?

A cloud is actually a collection of web servers [instead of single server] owned by 3rd party. Cloud provides inexpensive, efficient and flexible alternative to computers.

Benefits of Cloud are:

- No need of extra space required to keep all the hardware [Servers, Digital storage]
- Companies don't need to buy software or software license for all it's employees. They just pay a small fees to the cloud computing company to let their employees access a suite of software online.
- It also reduces IT problems and costs.

6.

```
class A{
```

```
}
```

```
class B extends A{
```

83 | Praveen Oruganti

Blog: <https://praveenorugantitech.blogspot.com>

Facebook Group: <https://www.facebook.com/groups/268426377837151>

Github repo: <https://github.com/praveenoruganti>

Email : praveenorugantitech@gmail.com

```

}

class C extends B{

}

public class MainClass{

    static void overloadedMethod(A a){
        System.out.println("One");
    }

    static void overloadedMethod(B b){

        System.out.println("Two");
    }

    static void overloadedMethod(Object obj){

        System.out.println("Three");
    }

    public static void main(String[] args){

        C c = new C();
        overloadedMethod(c);

    }

}

```

Given above code, what will be the output?

Output: Two

Explanation:

In method overloading, more specific method is chosen over generic.

7.As a best practice , what should be the maximum nesting level for associations in URI (e.g. /vacations/{id}/reviews)?

3

8.In a class, one method has two overloaded forms.One form is defined as static and another form is defined as non-static. Is that method properly overloaded?

Yes. That method is properly overloaded. Compiler checks only method signature to verify whether a particular method is properly overloaded or not. It doesn't check static or non-static feature of the method.

9. Method signature consists of which one out of below:

1. **Method name, return type and number of arguments**
2. **Access modifier, method name and types of arguments**
3. **Method name, number of arguments, types of arguments and order of arguments**
4. **Return type, access modifier and order of arguments**

Java Interview @ GalaxE India

1.What are Binary Literal? How to use them in java?

Binary Literals were introduced in java 7. Now using them, we don't need to convert binary to decimal or hexadecimal.

Binary Literals must be started with 0b or 0B.

Binary Literals are used in Protocols, Processors and bitmapped hardware devices.

Example showing their usage:

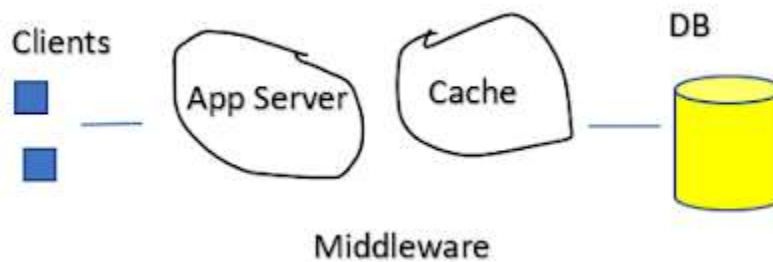
```
public class BLiterals{  
  
    public int a = 0b0110;  
    public byte b = (byte) 0b0110;  
    public long l = (long) 0b0110L;  
  
    System.out.println("a = "+a);  
    System.out.println("b = "+b);  
    System.out.println("l = "+l);  
  
}
```

Output:

```
a = 6  
b = 6  
l = 6
```

**2.Which Application server have you used? Where does it occur in multi-tier architecture?
What benefits we get while using an Application server?**

I have used Weblogic server.



Weblogic server provide support for Network protocols [HTTPS, SOAP etc.] It also provides data access and persistence from database server. It also supports SQL transactions for data integrity. Weblogic also provides security. So this means, when we use Weblogic server, we don't have to care about protocol, security, database integrity, transactions etc. All these are handled by Weblogic itself. We just have to focus on business logic.

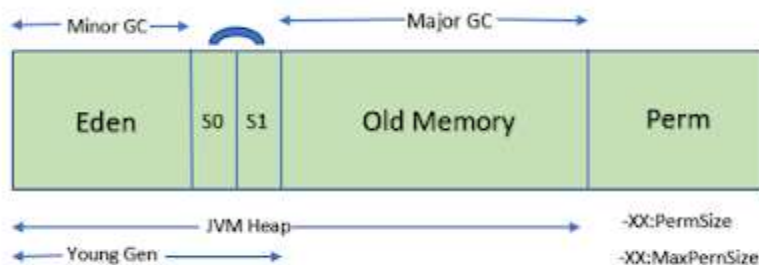
3. Write algorithm for Level-Order traversal of a Binary tree.

Level-Order traversal means moving from root to leaf step-by-step horizontally.

Algorithm for Level-Order Traversal:

1. Check if root node is empty. If yes, then return.
2. If root not null, create a queue and put root node in the queue.
3. Take a while loop on if Queue is not empty.
4. store the size of queue in a variable named size.
5. Create another while loop inside outer loop. IN this loop, check the value of size variable. It should be > 0. Use size-- in while loop.
6. Now print element[node] from queue. And put all child nodes of that node on queue if these are not null.
7. Continue from step 5 until it is false and then continue from step 3.

4. Explain Java Memory Model.



5. Explain JVM memory structure.

As per the spec, JVM is divided into 5 virtual memory segments:

- Heap
- Method [Non-heap]
- JVM Stack
- PC Registers
- Native Stack

JVM Stack:

- Has a lot to do with methods in java classes
- Stores local variables and regulates method invocation, partial result and return values.
- Each thread in java has it's own copy of stack and is not accessible to other threads.
- Tuned using -Xss JVM option.

Native Stack:

- Used for native methods [Non-java code]

6.What are the common Java Heap related issues?

Below is the list of all java heap related issues which occur in java applications at runtime.

- **'OutOfMemory' error due to insufficient Heap**
- To identify it, we can use JvisualVM
- To fix it, we can increase heap size or we can revisit the code to see why the demand is high in first place.
- **Poor application response time due to long garbage collection pauses**
- To identify it, we can use JvisualVM
- To fix this, we can tune GC [Garbage Collector].
- **OutOfMemory error due to memory leak**
- To identify it, we can use JvisualVM
- To fix it, we need to analyse the code and correct it.
- **Heap Fragmentation**
- It is due to when , small and large objects are allocated in a mixed fashion. To some extent, we cannot avoid heap fragmentation -- over time, heap will get fragmented.
- To identify, we see poor application response times, longer GC pauses and in some cases 'OutOfMemory' errors.
- To fix it, tuning can help.

7.What are the ways to capture Java Heap dump?

There are great tools like Eclipse MAT and Heap Hero to analyze Heap dumps. However we need to provide these tools with heap dumps captured in the correct format.

Options to capture heap dump are:

- Jmap
- Jcmd
- JvisualVM
- JMX

8. Why reflection is slow?

Reflection needs to inspect metadata in bytecode instead of just using precompiled addresses and constants.

Everything requires to be searched. That's why reflection is slow.

Java Interview @ Gemini Solutions

1. Write algorithm for Infix expression evaluation.
expression is : $3*(5-1)/3 + 1$

Algorithm:

Approach: Use two Stacks

- operand stack
- operator stack

Process:

1. Pop out two values from operand stack, let's say it is X and Y.
2. Pop out operator from operator stack. Let's say it is '+'
3. Do $X + Y$ and push the result on the operand stack.

Iterate through given expression, one character at a time.

1. If a character is an operand, push it on the operand stack.
2. If the character is an operator,
 - If the operator stack is empty, push it onto the operator stack.
 - Else if the operator stack is not empty,
 - If the character's precedence is greater than or equal to the precedence of the stack top of the operator stack, then push the character to the operator stack.
 - If the character's precedence is less than the precedence of the stack top of the operator stack, then do Process [As described above] until character's precedence is less or stack is not empty.
3. If the character is "(", then push it onto the operator stack.
4. If the character is ")", then do Process [As described above] until the corresponding "(" is encountered in operator stack. Now just pop out the "(".

2. What is the difference between Lambda and anonymous class?

There are multiple differences between lambda and anonymous class:

- Usage of 'this' keyword: IN anonymous class, this keyword resolves to anonymous class itself. Whereas for Lambda, this keyword resolves to enclosing class where lambda expression is written.
- Compilation: Java compiler compiles lambda expression and convert them into static private method of the class. It used invokedynamic bytecode instruction that was added in java 7 to bind this method dynamically. While anonymous class compiles to a class. The compiler generates a class file for each anonymous inner class.
- Lambdas implement a functional interface, so implements only 1 method. Anonymous class can extend a class or implement multiple interfaces with any number of methods.

3. Design Email [Inbox, Outbox, attachment]. Write all technologies that will be used. Create Class diagram.

4. What is the use of volatile keyword? How does it help in synchronization? Does it provide full synchronization?

Volatile is used to avoid the local copy of a variable inside a thread. All threads will access the volatile variable from main memory, updates it and immediately put it back in main memory.

All, reads and writes would be made directly to main memory instead of to registers or local processor cache.

volatile variable is not thread safe. If multiple threads try to write on a volatile variable, then Race condition occurs.

Each thread has a separate memory space known as working memory. This holds the value of different variables for performing operations. After performing an operation, thread copies the updated value of the variable to main memory and from there other threads can read the latest value.

But if the variable is non-volatile, then new value may not be flushed immediately to main memory. And other threads may not read the updated value.

5. Why to use transient variable? What will be the value of transient variable after deserialization? Where should we use transient variable?

transient variable is used when we don't want value of that variable to be serialized. So, if a variable is declared as transient, then its value will not be serialized.

On deserialization, we get the default value of transient variable. e.g. :

```
transient int i ;
```

When the instance of this class is serialized, then upon deserialization, we get 0 in i , as 0 is the default value for an int in java.

Even if we declare i as transient int i = 10;
In this case, as well we get value of i as 0 after deserialization.

Use case [Where should we use it]:

When we have a variable whose value can be calculated from other variables, then we should make it transient, as it should be calculated from other variables every time.

6. How will you write an immutable class?

There are two ways to write an immutable class:

- Make the class final and all its instance variables as final, so that they can be initialized in constructor only.
- Make all instance variable of the class as private and don't modify them inside in method. Means don't provide any setters methods. And also make all getters methods as final so that they can't be overridden and subclasses cannot override them and return some other values from overridden forms.

7. Write Lambda expression for printing even numbers from a list.

Suppose , we have a list of integers.

```
int[] array = {1,2,3,4,5};
```

```
List<Integer> list = Arrays.asList(array);
```

```
list.stream().filter(i-> i%2 == 0).collect().forEach(System.out :: println);
```

8. Difference between default method and regular method in java 8.

Differences between default method and regular method are as follows:

- default method has default keyword in its method signature. While regular method has no such keyword.
- default cannot change the state of the object as it works only inside interface and can only access its local variables. While a regular method can change the state of the object by modifying method arguments as well as fields of the class.

Java Interview @ OLX

1. If you have three, you have three. If you have two, you have two, but if you have one, you have none. What is it?

Choices

2. 3 bulbs 3 switches problem:

There is a room with a door [Closed] and three light bulbs. Outside the room, there are 3 switches, connected to the bulbs. You may manipulate the switches as you wish, but once you open the door, you can't change them. Identify each switch with its bulb.

Turn on 1 switch and keep it on for 5 minutes. Now turn it off and turn on 2nd button and enter the room. Now the bulb which is ON maps to ON switch. The hot bulb maps to previous switch [which was turned on first] and 3rd bulb maps to 3rd switch.

3. REST API must use HTTP. Is that true or false?

FALSE

4. A resource in the context of REST is (or may be) which one of these:

- Thing
- Object
- Real World Entity
- An account
- An Item
- A Book
- All of the above

All of the above

5. Suppose you have a "Worker" table as shown below. You have to write a SQL query to find employees with different salary.

90 | Praveen Oruganti

Blog: <https://praveenorugantitech.blogspot.com>

Facebook Group: <https://www.facebook.com/groups/268426377837151>

Github repo: <https://github.com/praveenoruganti>

Email: praveenorugantitech@gmail.com

Show only 2 Columns in output: first_name and salary.

	worker_id	first_name	last_name	salary	joining_date	department
▶	1	Monika	Arora	100000	2014-02-20 09:00:00	HR
	2	Niharika	Verma	80000	2014-06-11 09:00:00	Admin
	3	Vishal	Singhal	300000	2014-02-20 09:00:00	HR
	4	Amitabh	Singh	500000	2014-02-20 09:00:00	Admin
	5	Vivek	Bhati	500000	2014-06-11 09:00:00	Admin
	6	Vipul	Diwan	200000	2014-06-11 09:00:00	Account
	7	Satish	Kumar	75000	2014-01-20 09:00:00	Account
	8	Geetika	Chauhan	90000	2014-04-11 09:00:00	Admin
•	NULL	NULL	NULL	NULL	NULL	NULL

```
select distinct w1.salary , w1.first_name
from Worker w1, Worker w2
where w1.salary = w2.salary
AND w1.worker_id = w2.worker_id;
```

6. Write SQL query to display first 5 records from the table shown above?

```
select * from worker LIMIT 5;
```

7. What is the difference between UNION and UNION ALL?

UNION removes duplicate records. UNION ALL does not.

There is a performance hit when using UNION instead of UNION ALL, since the database server must do additional work to remove the duplicate rows. But usually, we don't want the duplicates especially when developing reports.

8. What are foreign key and super key?

Foreign Key:

Foreign key maintains referential integrity by enforcing a link between the data in two tables.

Foreign key in child table references the primary key in parent table.

The foreign key constraint prevents actions that would destroy the link between the child and parent table.

Super key : It is a column or a combination of columns which uniquely identifies a record in a table.

e.g.:

Super key stands for superset of a key. e.g. We have a table Book with columns:
Book (BookID, BookName, Author)

So, in this table we can have:

- (BookID)
- (BookID, BookName)

91 | Praveen Oruganti

Blog: <https://praveenorugantitech.blogspot.com>

Facebook Group: <https://www.facebook.com/groups/268426377837151>

Github repo: <https://github.com/praveenoruganti>

Email: praveenorugantitech@gmail.com

- (BookID, BookName, Author)
- (BookID, Author)
- (BookName, Author)

as our super keys.

Each super key is able to uniquely identify each record.

Java Interview @ OrangeMantra

1.List all types of HTTP Error code categories

There are mainly 5 categories in HTTP Response codes:

- 1XX : Informational
 - 100 : Continue
- 2XX : Success
 - 200 : OK
 - 201 : Created
 - 202: Accepted
 - 204: No Content
- 3XX : Redirection
- 4XX : Client Side Error
 - 400 : Bad Request
 - 401 : Unauthorized
 - 402 : Payment Required
 - 403 : Forbidden
 - 404 : Not Found
- 5XX : Server Side Error
 - 500 : Internal Server error
 - 501: Not Implemented
 - 502: Bad Gateway
 - 503: Web service not available

2.List all the features in Java 7

There are multiple features which have been added in java 7:

- String in switch
- try with resources
- Binary Literals
- multiple Exception types in catch block

3. Explain about one very typical designing/Coding situation which you have solved.

4. Explain all differences between Stack and Heap?

- Heap is very large in size while Stack is small in size as compared to Heap.
- Every thread has it's own Stack while Heap is shared among all threads.

- When Heap gets full, java.lang.OutOfMemoryException is thrown. While in case of Stack, java.lang.StackOverflowException is thrown.
- Heap is mainly used to store the objects, while Stack is used for storing local data, method body , method return values etc.
- Heap is tuned by using -Xmx and -Xms options. While Stack size is tuned by -Xss.
- In Heap, data is stored in random order. While in Stack , data is stored in LIFO [Last-In-First-Out] order.

5.What is the contract between natural ordering and equals() method in java?

Contract between natural ordering or ordering provided by comparator interface and equals() method is that the output returned by natural ordering / comparator interface should be consistent with value returned by equals() method.

If these values are not consistent, then it will lead unexpected results in TreeMap and TreeSet Collections.

6.What is the use case of Marker interface ? How to write custom marker interface?

Marker interface is used to indicate something special to JVM/Compiler so that it can perform necessary actions on the class implementing marker interface.

Examples of marker interfaces are : Serializable, Cloneable, Remote

Custom marker interface:

```
public interface Ingestible{
    //nothing
}

public interface Edible extends Ingestible{

}

public interface Drinkable extends Ingestible{

}

public class Food implements Edible{

}

public class Drink implements Drinkable{

}

public class Person{
```

```

public void ingest(Ingestible something){

    if(something instanceof Edible){

        System.out.println("something can be eaten");

    }

}
}

```

In above example, we are telling JVM that the class implementing Ingestible interface can be ingested.

Use case:

A marker interface is used for Identification purpose.

7. Why to use BigDecimal over Float or Double?

Float and Double cannot be used for precise values: e.g.:

```

double d1 = 131.44;
double d2 = 130.34;

```

```

System.out.println("d1 - d2 = "+ (d1-d2));

```

Output: 1.100000000000013

That's why in financial applications, where while doing calculations , scale and rounding mode for the numbers is an important aspect . So it's a better choice to go with BigDecimal.

Using BigDecimal:

```

BigDecimal bd1 = new BigDecimal("131.44");
BigDecimal bd2 = new BigDecimal("130.44");

```

```

System.out.println(bd1-bd2); gives 1.10

```

8.

```

Integer i = 127;
Integer j = 127;

```

```

Integer ii = 128;
Integer kk = 128;

```

```

if( i == j){
    System.out.println("true");
}
else{
    System.out.println("false");
}

```

```

if(ii == kk){
    System.put.println("true");
}

```

94 | Praveen Oruganti

Blog: <https://praveenorugantitech.blogspot.com>

Facebook Group: <https://www.facebook.com/groups/268426377837151>

Github repo: <https://github.com/praveenoruganti>

Email : praveenorugantitech@gmail.com

```

}
else{
    System.out.println("false");
}

```

What will be the output?

true
false

9.What is Serverless architecture?

Serverless is a cloud computing execution model where the cloud provider dynamically manages the allocation and provisioning of servers.

A serverless application runs in stateless compute containers that are event-triggered , ephemeral(may last for one invocation) and fully managed by the cloud provider.

Pricing is based on the number of executions rather than pre-purchased compute capacity.

Our applications are installed in terms of cloud functions. There are multiple cloud services that act as Functions as a service [FaaS].

- IBM OpenWhisk
- AWS Lambda
- Microsoft Azure Functions
- Google Cloud Functions
- Auth0 Webtask

So, serverless computing [or serverless for short] is an execution model where the cloud provider is responsible for executing a piece of code by dynamically allocating the resources and only charging for the amount of resources used to run the code.

The code typically runs inside stateless containers that can be triggered by a variety of events including Http requests, database events, file uploads, monitoring alerts etc.

10. What happens when we call SpringApplication.run() method in main class of SpringBootApplication?

Syntax of the class containing main method is:

```

@SpringBootApplication
public class TestApplication{

    public static void main(String[] args){

        SpringApplication.run(TestApplication.class); // returns
        ConfigurableApplicationContext instance
    }

}

```

When we run this class as a java application then our application gets started.

When this method is called, then Spring Container gets started.

Spring container once started is responsible for:

- Creating all objects: This is done by component scan. As @SpringBootApplication is a combination of @Configuration , @ComponentScan and @EnableAutoConfiguration.
- Dependency Injection
- Managing the lifecycle of all beans.

Steps executed under this method:

- Application Context is started
- Using application context , auto discovery occurs: @ComponentScan
- All default configurations are setup i.e. based on dependencies mentioned, spring boot automatically sets up defaults.
- An embedded servlet container is started.[No need to setup a separate web server]. Note: Embedded servlet container is launched only if web is mentioned in a dependency.

11.What are Spring Boot Starter projects. Name some of them?

Starters are a set of dependency descriptors that we can include in our application. We get one-stop-shop for all required technologies in our application. without having to hunt through sample code and copy paste loads of dependency descriptors.

e.g.: If we want to get started using Spring and JPA for database access, just include the sprint-boot-starter-data-jpa dependency in the project and we are good to go.

Below is a list of few starter projects provided by Spring Boot:

- sprint-boot-starter-web
- spring-boot-starter-data-jpa
- spring-boot-starter-web-services
- sprint-boot-starter-test
- sprint-boot-starter-data-rest
- spring-boot-starter-jdbc
- spring-boot-starter-security

12.How does spring enables creating production ready applications in quick time?

Spring Boot enables creating production ready applications by providing a few non-functional features out of the box like caching, monitoring, logging and embedded server.

Spring-boot-starter-actuator: For monitoring and tracing of application

Spring-boot-starter-undertow, spring-boot-starter-jetty, spring-boot-starter-tomcat: To pick the choice of Embedded Servlet Container.

Spring-boot-starter-logging: For logging using Logback

Spring-boot-starter-cache: Enabling spring framework's caching support

96 | Praveen Oruganti

Blog: <https://praveenorugantitech.blogspot.com>

Facebook Group: <https://www.facebook.com/groups/268426377837151>

Github repo: <https://github.com/praveenoruganti>

Email : praveenorugantitech@gmail.com

13.What is the minimum baseline version for spring boot 1.5 and 2.1.0 versions?

Java 8

14.How many ways are there to create ApplicationContext?

If we are using xml configuration using application-context.xml then

```
ApplicationContext context = new ClassPathXmlApplicationContext("Path to xml file");
```

If we are using Java configuration, then:

```
ApplicationContext context = new AnnotationConfigApplicationContext(ConfigClass.class);
```

Java Interview in Concirrus

1.How Singleton is handled in Deserialization? What, if we use clone() in singleton?

When we serialize Singleton instance and try to deserialize it, then if we call deserialization multiple times, then it can result in more than one instance of singleton class.

To avoid this problem, we can implement readResolve() method. This readResolve() method is called immediately after an object of this class is deserialized, Means when ObjectInputStream has read an object from input stream and is preparing to return it to the caller, then it checks whether the readResolve() method is implemented or not.

Note: Both objects [Read by ObjectInputStream and returned by readResolve() method] should be compatible [Identical] else ClassCastException is thrown.

Implementation of readResolve() method:

```
protected void readResolve(){  
    // Instead of the object we are on, return the class variable singleton.  
    return singletonInstance;  
}
```

What if we use clone() method in Singleton?

If we override clone() method in singleton class, then we must throw CloneNotSupportedException from this method.

2.How HashMap works?

HashMap in java contains key-value pairs. It doesn't maintain the insertion order of key-value pairs.

Whenever a key-value pair is required to be stored in HashMap, The overridden hashCode() method [if there is one] is used to calculate the hashcode and then Object's class hash() method is called on that

value which provides the index in underlying bucket of HashMap.

HashMap uses array based structure to store each key-value entry. That array is called bucket and each location in bucket maintains a simple optimized LinkedList.

So, after the index in the bucket is found, then if there is no entry matching the key in that bucket indexed LinkedList, then that key-value pair is stored. Else , if there is some key-value pairs already there in the LinkedList, then key is compared to every key in LinkedList using equals() method and if a match found, then the value for that key is replaced with new value.

If there is no match found, then that new key-value pair is stored as a new entry in HashMap.

3.Create Immutable class?

There are 2 ways to create immutable class in java:

- Make the class final and all it's instance variable as final. So they can be initialized in constructor only.
- Make all the instance variables private and don't change them except constructor. Don't provide any setter methods for them. Make the getter methods final so that subclasses don't override these getter methods and return other values.

4.Detect loop in LinkedList

```
class LinkedList{

    static Node head;

    static class Node{
        int data;
        Node next;

        Node(int d){
            data = d;
            next = null;
        }
    }

    int detectLoop(Node node){

        Node slow = node, fast = node;

        while(slow != null && fast != null && fast.next != null){

            slow = slow.next;
            fast = fast.next.next;

            // If fast node reaches slow node, means there is a loop.
            if(slow == fast){

                return 1;
            }
        }
    }
}
```

```

    }
    return 0;
}

}

```

5.Dynamically configuring a new microservice. Does it require deploying all microservices, if we use static way?

No. We can just deploy our microservice to some host and it will register itself with Service Registry. From there it can be discovered by other microservices and also it can discover other required microservices.

And this way, communication can be established among various microservices without any external configuration.

6.Is there any loophole in using Git-> Jenkins CI/CD way?

There is no loophole in Git-> Jenkins CI/CD way. The problems/loopholes occur in the way these tools are configured by Software Delivery teams [DevOps].

There are multiple things that can cause problems in using Jenkins as CI/CD tool:

- Jenkins has too many plugins
- Jenkins was not designed for the Docker age
- Jenkins doesn't support microservices well

Jenkins has too many plugins:

Plugins are good when they are used properly and efficiently. Plugins give users the choice to add various features to the tools they use.

But in Jenkins, for achieving every single basic task, you need a separate plugin.

e.g.: for creating a build for Docker environment, you need a plugin.
To pull code from Github, you need a plugin.

Jenkins was not designed for the Docker age:

CI servers don't match with Docker container infrastructure easily. They require multiple plugins to integrate with Docker. There are more than 14 plugins with Docker in their names. And almost 6 of them are for core Docker platform.

7.Design TinyURL algorithm.

8.Which Docker command is used to know the list of running containers?

docker ps

9.Difference between GitHub and GitLab?

GitHub vs GitLab

GitHub	GitLab
GitHub is the most popular web-based hosting service for Git repositories	GitLab has everything GitHub has, but with extra features and increased control over repositories.
It does not offer detailed documentation for the common git repositories.	It offers a detailed documentation for the common git repositories on how to import/export data.
It offers an easy-to-use, intuitive UI for project management.	It offers more convenient UI allowing users to access everything from one screen.
It offers various third-party integrations for continuous integration and continuous delivery work.	It offers its very own CI/CD which comes pre-built, meaning users do not have to install it separately.

Interview @ NexGen Solutions

1. What is the benefit of Executor framework?

Executor Framework provides separation of Command submission from Command execution.

It has several interfaces. Most important ones are:

- Executor
- ExecutorService
- ScheduledExecutorService

Executors is Factory methods class used in this framework.

2. Asked me to write custom ArrayList.

//You can write your own implementation

3. In your Custom ArrayList, there are multiple methods. How will you provide synchronization?

We have two ways of providing synchronization in this source code:

- Make each method synchronized by placing synchronized keyword before return type of method
- Use synchronized block inside method body.

4.How did you use Amazon EC2?

I installed EC2 instance using Amazon Management console after login into it.
I used General instance for my purpose, as in my application I needed to send multiple emails to client very fast.

5.What is the benefit of Generics in java?

Generics provides Compile-time safety. And it also avoids ClassCastException. It was introduced in Java 5.

6.How Generic code written in java 5 runs in Java 4 compiler?

100 | Praveen Oruganti

Blog: <https://praveenorugantitech.blogspot.com>

Facebook Group: <https://www.facebook.com/groups/268426377837151>

Github repo: <https://github.com/praveenoruganti>

Email : praveenorugantitech@gmail.com

Java compiler uses type check information at compile time and after all validation it generates type erased byte code which is similar to the byte code generated by Java 4. So, it provides backward compatibility.

7. Asked about my last project.

I explained it.

8. Why we can't pass List<String> to List<Object> in java?

String is a subclass of Object, but List<String> is not a subclass of List<Object>.

If it would be allowed, then look at below code:

```
List<Object> listO = new ArrayList<String>();  
listO.add("String");  
listO.add(new Object());
```

Now, in the same List, we have both a String and Object object. Which is wrong.

Java Interview @ Dew Solutions

1. Java is pass by value or reference.

Everything in java is pass-by-value.

When we pass object reference to some method, we actually pass address of that reference which in turn contains memory address of actual object.

And as we know, memory address is a value, so means, we always use pass-by-value in java.

e.g.:

```
public static void main(String[] args){  
  
    Book book = new Book("Java");  
    Book newBook = book;  
  
    change(book);  
    book.getName();// Prints Java  
  
}  
  
public void change(Book book){  
    book.getName(); //Prints Java  
  
    book = new Book("Angular");  
    book.getName();// Prints Angular  
}
```

So here, what we see Book name doesn't change after change() method, because passed book reference in change() method now points to new Book Object with value "Angular".

Now, lets look at another example:

```
public static void main(String[] args){

    Book book = new Book("Java");
    Book newBook = book;

    change(book);
    book.getName();//Prints .Net
}

public void change(Book book){
    book.getName();//Prints Java

    book.setName(".Net");
}
```

Here, what we see, value of book object after change() method call has been changed to .Net. It is because book reference in change() method still contains memory address of Book object with java as value.

2.We have multiple Employee objects and we need to store these objects in TreeMap. What problems we can face while storing Employee objects?

We will get error at runtime : "No suitable method found for sort(List<Employee>)". It is because Employee class doesn't implement the Comparable interface so the sort() method cannot compare the objects.

As TreeMap stores object in Ascending order by default using natural ordering. So, each object which needs to be stored in TreeMap should implement Comparable interface or Comparator interface.

3.What is Callable interface and how it is similar/different to/from Runnable?

Callable interface is just like Runnable interface.

Difference between Callable and Runnable is given below:

- Runnable is defined in Java 1.0 . While Callable was introduced in Java 5.
- Callable includes call() method which returns object and also throws Checked exception. On the other hand , Runnable's run() method neither return any value nor throws any Checked exception.
- Runnable instance can be passed to a thread. While Callable instance can't be passed to a thread.

Similarities between Runnable and Callable:

- Both are used to encapsulate code which needs to be executed parallelly in separate thread.
- Both interfaces can be used with Executor framework introduced in java 5.
- Both includes single abstract method. Means , both can be used in Lambda expressions in java 8.

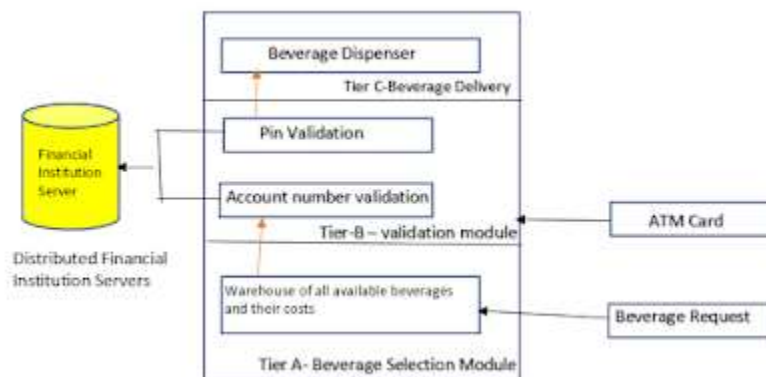
4.Can HashSet store duplicate objects?

No, HashSet cannot store duplicate objects. As HashSet is implementation of Set interface and Set interface is meant to store unique elements. That's why HashSet doesn't store duplicate objects.

5.SQL query to find employee with highest salary and corresponding department.

```
select department , max(salary) from employee  
group by department;
```

6.Design the architecture of Vending machine?



7.Describe one problem that you have solved in production build?

In my previous company, client complained about slowness in the system performance. We tracked the logs and found there was problem with some memory leaks happening.

We just immediately asked client to increase the JVM heap size for the moment so that it worked and later on we corrected the problem by removing all the memory leaks in the code.

8. Why to use Lock in multithreading? What are the locking mechanisms available in java?

Lock is required in multithreading to properly distribute access of Monitor associated with an object among multiple threads.

If we don't use Locks, then multiple threads will try to acquire shared resources and we will get corrupted outputs from threads.

Locking mechanisms available in java are:

- synchronized block
- synchronized methods
- Read/Write lock
- Atomic classes
- final classes

Java Interview @ CapGemini

1.How to handle transaction in Spring?

In Spring, there are two ways of managing transactions:

- Programmatically manage by writing custom code
- Using Spring to manage transaction
- Programmatic Transaction management
- Declarative Transaction management

Programmatically manage by writing custom code:

```
EntityManagerFactory factory = Persistence.createEntityManagerFactory("persistence_unit_name");
EntityManager entityManager = factory.createEntityManager();
Transaction transaction = entityManager.getTransaction();
```

```
try{
    transaction.begin();
    //.....Business logic
    transaction.commit();
}
catch(Exception e){
    transaction.rollback();
    throw e;
}
```

Pros:

- The scope of the transaction is very clear in the code

Cons:

- It's repetitive and error prone
- Any error can have a very high impact
- A lot of boilerplate code needs to be written and if you want to call any another method from this method, then again you need to manage it in the code.

Using Spring to manage transaction:

1. Programmatic Transaction Management:

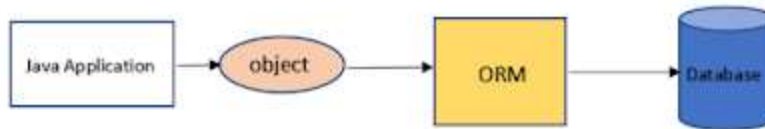
- Using the transaction template
- Using a PlatformTransactionManager implementation directly

2. Declarative Transaction

2.What is ORM and how to use it?

ORM means Object Relational Mapping. It is a tool that helps in interacting with the database.

for example JPA is an ORM which specifies some specifications which needs to be implemented. In Spring, Hibernate is the implementation of JPA specifications.



So using ORM, Java object is mapped to relational table in database.

3.What is Heap dump?

Heap Dump: It is a snapshot of the memory of a java process.

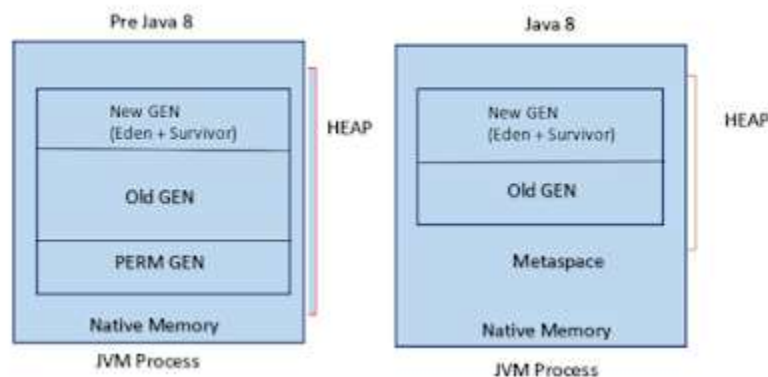
This snapshot contains information about java objects and classes in the heap at the moment the snapshot is triggered.

Typically , a full GC is triggered before the heap dump is written, so the heap dump contains information about the remaining objects in the heap.

Typical information in a heap dump are:

- All objects: Class, fields, primitive values and references
- All Classes: Class Loader, name, super class and static fields
- Garbage collection roots: It is an object that is accessible from outside the heap.

4.Memory management in Java 8?



Perm Gen in java 7 and earlier version has a default maximum size of 64 MB for 32-bit machine and 84 MB for 64-bit machine.

While default maximum size for metaspace is unlimited.

In java 8 ,we can set the initial and maximum size of metaspace using:

-XX:MetaspaceSize = N : Sets the initial and minimum size of metaspace

-XX:MaxMetaspaceSize=N : Sets the maximum size of the metaspace

5.Difference between Serial and Parallel streams.

Parallel streams divide the task into many and run them in different threads, using multiple cores of the computer. It uses main thread along with ForkJoinPool.commonPool worker threads.

Sequential stream uses a single core. So it uses only 1 thread that is main thread.

Sequential vs parallel streams running in 4 cores



6.How to find OutOfMemoryError in production app? And how to avoid this error immediately for long time?

If we face OutOfMemoryError in production and we cannot really reason about it from stacktraces or logs, we need to analyze what was in there.

Get the VM Heap dump on OOM:

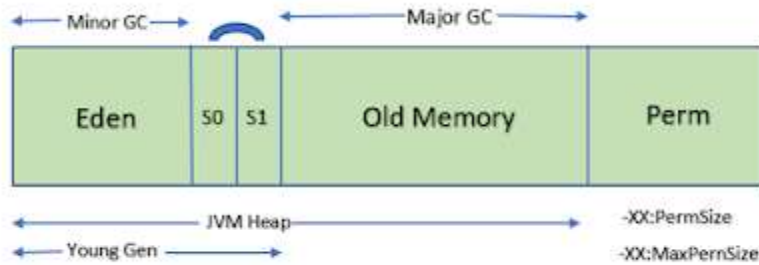
- -XX:+HeapDumpOnOutOfMemoryError
- -XX:HeapDumpPath="/tmp"

And use that for analysis. We can use memory analyzer tool for this.

How to avoid this error immediately for long time?

We can increase Heap space using -XX argument for a while and it will work fine.

7.What is internal structure of heap memory?



8. How to configure GC in Java application?

GC tuning is only required when we have not set memory size in the application and too many Timeout logs are printed.

CLASSIFICATION	OPTION	DESCRIPTION
Heap area size	-Xms	Heap area size when starting JVM
	-Xmx	Maximum heap area size
New area size	-XX:NewRatio	Ratio of New area and Old area
	-XX:NewSize	New area size
	-XX:SurvivorRatio	Ratio of Eden area and Survivor area

9. What are Serial and Parallel GC's?

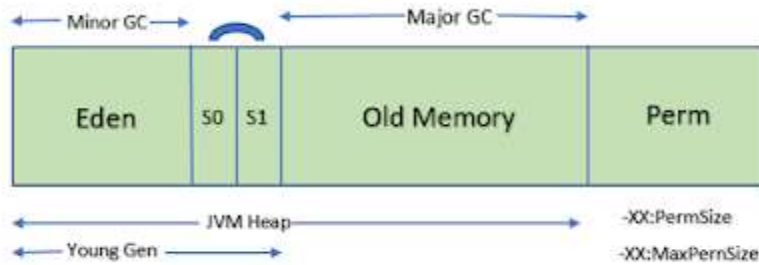
Serial Collector:

- Used for single threaded environments [32-bit Windows] and apps that use small heap
- It stops all threads when it is running. Means it is not suitable for server environment.
- We can use it by turning on the `-XX:+UseSerialGC` JVM argument.

Parallel Collector:

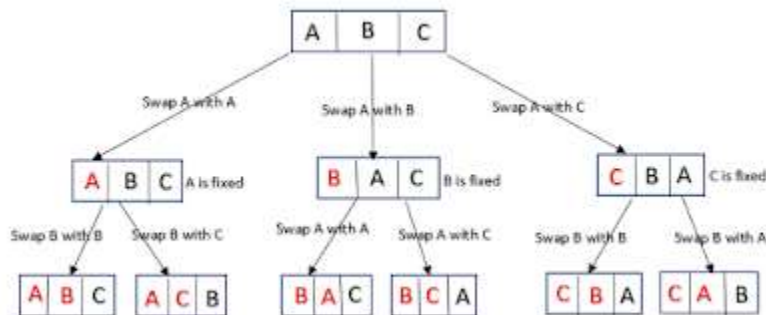
- This is JVM's default garbage collector
- It uses multiple threads to scan through and compact the heap and speeds up garbage collection.
- The downside is that it will stop application threads when performing either a minor GC or full GC.
- It is best suited for apps that can tolerate application pauses.
- Although it is default collector, but it can be configured using `-XX:UseParallelGC` JVM argument

10. What is Java Memory model?



Java Interview @ BirdEye

1. Print all unique permutations on a String?



2. Write Algorithm for custom BlockingQueue.

Algorithm steps:

- Define an array to store elements for queue. Specify the initial size for that array.
- Use Lock and conditions objects to create custom blocking queue.
- Define two methods , put() and take().
- While putting the elements in queue, check the size of array. If it is full, then the producer will wait for the queue to have some space.
- While consuming element from queue, if the queue is empty, then consumer will wait for the queue to have some objects in it.

Packages to be included:

```
java.util.concurrent.locks.Condition;
java.util.concurrent.locks.Lock;
java.util.concurrent.locks.ReentrantLock;
```

3. Singleton and Synchronization question:

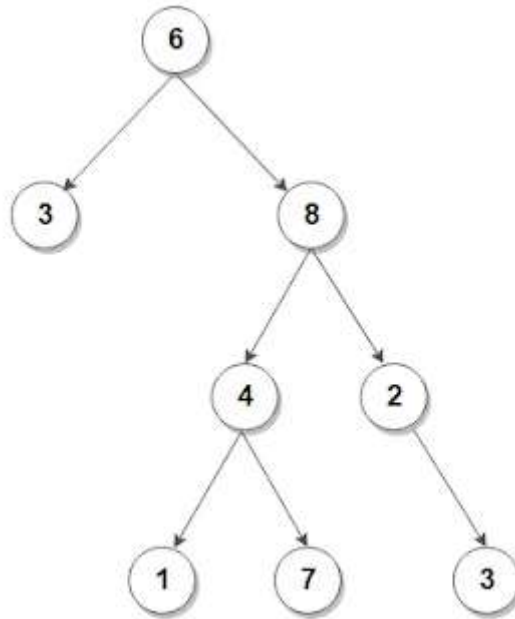
If Thread1 calls synchronized method in Singleton class, then can another thread call getInstance() method [If synchronize(Singleton.class) is 1st statement in getInstance() method] of singleton class?

Yes, another thread can call getInstance() method of singleton class. It is because, this time thread will

acquire lock on Class object [Singleton.class].

So first thread acquired lock on Singleton instance and this another thread will acquire lock on Singleton Class's object.

4. Print all edge/corner nodes of a binary tree.



Follow Level order traversal of binary tree. So , while doing level order traversal, if the current node happens to be the first node or last node in current level, print it.

```
void print(Node *root){  
  
    //return if tree is empty.  
    if(root == null)
```

109 | Praveen Oruganti

Blog: <https://praveenorugantitech.blogspot.com>

Facebook Group: <https://www.facebook.com/groups/268426377837151>

Github repo: <https://github.com/praveenoruganti>

Email : praveenorugantitech@gmail.com

```

        return;

// Create an empty queue to store tree nodes.

Queue<Node*> q;

//enqueue root node
q.push(root);

// run till queue is not empty

while(!q.empty()){

    //get size of current level
    int size = q.size();
    int n = size;

    //Process all nodes present in current level
    while(n--){

        Node* node = q.front();
        q.pop();

        // If corner node found, print it.
        if(n == size-1 || n == 0)
            println(node);

        //enqueue left and right child of current node
        if(node-> left != null)
            q.push(node->left);

        if(node -> right != null)
            q.push(node->right);
    }
    //terminate level by printing newline
    println();
}
}

```

5.How locking mechanism is implemented by JVM?

The implementation of locking mechanism in java is specific to the instruction set of the java platform. For example with x86, it might use the CMPXCHG instruction - atomic compare and exchange - at the lowest level to implement the fast path of the lock.

The CMPXCHG instruction is a compare-and-swap instruction that guarantees atomic memory access at the hardware level.

If the thread cannot acquire the lock immediately , then it could "spinnlock" or it could perform a syscall to schedule a different thread. Different strategies are used depending on the platform , JVM Switches.

6. Is Java pass-by-value or pass-by-reference?

Java is always pass-by-value. Whenever we pass an object to some method, then we actually send a reference variable that points to the actual object. Means that reference variable will be containing the memory address as value. That's why Java is called as pass-by-value.

Let's take an example:

```
public static void main(String[] args){

    Book book = new Book("Java");
    Book bookNew = book;

    change(book);
    book.getName(); //Prints Java
}

public void change(Book book){
    book.getName() // Prints Java

    book = new Book("Angular");
    book.getName(); // Prints Angular
}
```

So, here what we see, book name doesn't change after change() call, because passed book reference value points to new Book object [Angular].

Now, let's look at another example:

```
public static void main(String[] args){
    Book book = new Book("Java");
    Book bookNew = book;

    change(book);
    book.getName(); //Prints .Net
}

public void change(Book book){

    book.getName(); //Prints Java
    book.setName(".Net");
}
```

Here, what we see, value of book object gets changed after change() method call. Because book reference value in change() method still contains the address of actual Book object and it will act upon it only.

Java Interview @ RBS

1. What is the difference between PUT and POST? Which request you will use to recharge a mobile phone?

PUT vs POST:

- PUT is used for update operation. While POST is mainly used for Create operation.
- PUT is idempotent. So if we retry a request multiple times, that should be equivalent to single request modification. While POST is not idempotent. So, if we retry a request N times, you will end up having N resources with N different URI's created on server.

Generally, in practice, always use PUT for Update operations and always use POST for create operations.

We will use POST request to recharge a mobile, because we need to create an entry for that mobile number on the server side. And we know, for creation purpose, we use POST.

2.Difference between Collections API and Stream API.

Collection API was introduced in Java 5. While Stream API was introduced in Java 8.

Collection API is used to persist the elements. While Stream API doesn't persist the elements. It just performs operations on elements.

In Collection API, Iterator is used to iterate the collection elements. While in Streams, Spliterator is used to iterate over elements.

3.Is Stream API faster than collection?

Yes, if we are using parallel streams, then performance of streams is better than collections.

As parallel streams use multi cores available in today's hardware, and it uses Fork Join pool and gives better performance.

4.What are the changes in Collections API in Java 8?

Java 8 has made various changes in Collection API:

- Sorting map directly with Comparators
- Iterate over Map easily with forEach.
- Get rid of if-else condition, use getOrDefault method
- Replace and Remove utilities
- Operate directly on values
- Merge maps with merge method
- Performance improvement in HashMap, LinkedHashMap, ConcurrentHashMap

5.Have you used JProfiler? How does it work and why you will use it?

Yes, I have used JProfiler. It is a code profiling tool.

We have to install it and then integrate with any IDE, Eclipse or IntelliJ.

It provides all the information we need with proper views and filters.

It displays the information while the program is running.

It is used to analyze performance bottlenecks like memory leaks, CPU load etc. and is also used to resolve threading issues.

In Eclipse, we use the Profile command to profile the code using JProfiler.

JProfiler is very easy to use and finds problems in the code.

112 | Praveen Oruganti

Blog: <https://praveenorugantitech.blogspot.com>

Facebook Group: <https://www.facebook.com/groups/268426377837151>

Github repo: <https://github.com/praveenoruganti>

Email: praveenorugantitech@gmail.com

6.What are Microservices?

The main principle behind microservices is to break a single large monolithic system into multiple independent components/processes.

Microservices architecture allows decoupled components to be built and deployed independently to integrate into a single larger system.

These components interact with each other through a standard XML/JSON interface, irrespective of the technologies used to create the component.

7.What is REST API?

REST stands for Representational State Transfer

REST is an architectural style for designing networked applications and developing web services.

REST API uses HTTP protocol methods/verbs for all its operations.

It revolves around resources where every component is a resource which can be accessed by a common interface using HTTP methods.

8.Have you worked with Spring Boot? Which version?

Yes, I've worked with Spring Boot. And version used is Spring Boot 2.

9.What are the features added in Java 8?

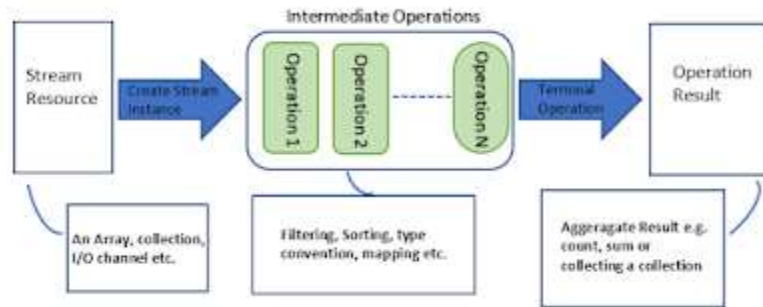
Below are all the features added in java 8:

- Functional interfaces, Lambda Expressions
- New Date and Time API
- Streams
- default methods in Interfaces
- static methods in interfaces
- Collection API improvements
- forEach() method in Iterable interface.

10.What are Stream Lifecycle methods?

Stream lifecycle is divided into three types of operations:

- Stream Source : Array, Collection, IO Channel
- Intermediate operations: filter, sort, map, flatMap
- Operation result: List, count, sum



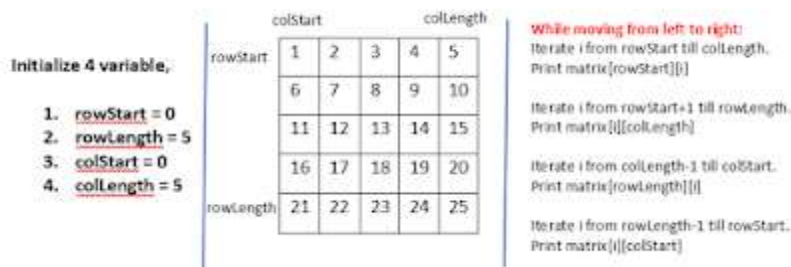
Java Interview @ Virtusa Polaris

1. Find frequency of a character and if more than one characters have same frequency then find character with more ASCII value.

Algorithm to solve this problem:

- Convert String into char array and take a HashMap, count variable and a char variable.
- Iterate this array.
- for every character, store it inside HashMap as key and with its frequency as value. Also increase its frequency value by 1. Along with that compare this frequency with count variable and if it is > count, then update count with frequency and also update char variable with that specific character for which we have updated count variable.
- If frequency = count, then check if new character's ASCII code is > ASCII of char. If it is, the update char with new character.
- At last, we will have the required character in char variable with more frequency or with more ASCII value, in case some other char has same frequency.

2. Implement spiral movement in 2D array.



Left to Right:

Move variable i from rowStart till colLength. Print data from first row till last column.

Top to Bottom:

Move variable i from (rowStart+1) till rowLength. Print data in last column.

We need to start from rowStart+1, because we already printed corner element in Left to Right printing and no need to include it again. Same treatment for corner elements in other directions.

Right to Left:

Move variable i from colLength - 1 till colStart. Print data in last row.

Bottom to Up:

Move variable i from rowLength - 1 till rowStart. Print data in first column.

After printing all 4 directions , in next iteration, we need to start from second row and second column , so increment rowStart++ and colStart++.

We need to print till second last column and till second last row , so decrement (colLength--) and (rowLength--).

3.What are the approaches for REST API versioning ?

There are multiple approaches for doing versioning in REST API.

Important ones are described below:

URI Versioning:

Using the URI is the most straight forward approach[and also most commonly used]. Though it does violate the principle that URL should refer to a unique resource.

`http://api.example.com/v1`

`http://apiv1.example.com`

Versioning using custom Request Header:

e.g.:

`Accept-version : v1`

`Accept-version : v2`

Versioning using Accept Header:

e.g.:

`Accept : application/vnd.example.v1+json`

`Accept : application/vnd.example+json;version=1.0`

4. Explain Builder Design pattern and in which scenario it is used?

What problem does it solve?:

- Class constructor requires a lot of information.

So, whenever we have an immutable class, then we need to pass all the information/parameters inside the constructor of the class.

When to use Builder Pattern:

- When we have a complex process to construct an object involving multiple steps, then builder design pattern can help us.

115 | Praveen Oruganti

Blog: <https://praveenorugantitech.blogspot.com>

Facebook Group: <https://www.facebook.com/groups/268426377837151>

Github repo: <https://github.com/praveenoruganti>

Email: praveenorugantitech@gmail.com

- In builder, we remove the logic related to object construction from "client" code and abstract it in separate classes.

5. Difference between split() and StringTokenizer class?

split() vs StringTokenizer:

Using StringTokenizer class, we have the option to use multiple delimiters with the same StringTokenizer object. But that's not possible with split() method.

split() method in String class is more flexible and easy to use. But StringTokenizer class is faster than split() method.

e.g.:

```
StringTokenizer st = new StringTokenizer("a:b:c" , ":");
```

```
while(st.hasMoreTokens()){
    System.out.println(st.nextToken());
}
```

StringTokenizer with multiple identifiers:

```
StringTokenizer st = new StringTokenizer("http://100.90.80.3/", "://.");
```

```
while(st.hasMoreTokens()){
    System.out.println(st.nextToken());
}
```

Output:

```
100
90
80
3
```

Using split() method:

```
for(String token : "a:b:c".split(":")){
    System.out.println(token);
}
```

6. Why String is immutable in java?

String is immutable due to multiple reasons:

- Due to String Pool facility. The String Pool is managed by String class internally.
- Due to Network Security, as URL is sent in String format.

- Due to thread security. Strings can be shared among multiple threads.
- Class loading mechanism
- Immutability allows string to store its hashcode and we don't need to calculate hashcode every time we call hashCode() method, which makes it very fast as hashmap keys to be used in HashMap in java.

7. Why the variables defined in try block cannot be used in catch or finally?

When we define any variable in try block and also use that variable in catch or finally, then suppose some exception occurs in try block before the line where that variable is defined. In that case, control goes to catch or finally and we will be using undefined variable in these blocks, because exception occurred before the declaration.

That's why variables defined in try block cannot be used in catch or finally.

Java interview @ handygo

1. What are the features of Java 8 version?

Java 8 Features:

- Functional Interfaces
- Lambda Expressions
- Streams
- New Date and Time API
- Changes in Collections API
- Changes in Map classes, HashMap, LinkedHashMap, ConcurrentHashMap
- Added StampedLock

2. How Java Streams are lazy? Explain.

Streams in java contain ternary operations. Like count(), collect(), list() etc. So, intermediary operations on streams are not executed until ternary operation is not called. That's why streams are lazy in nature.

3. Tricky One: I have a functional interface with 3 abstract methods in it. How I'll write lambda expression for it?

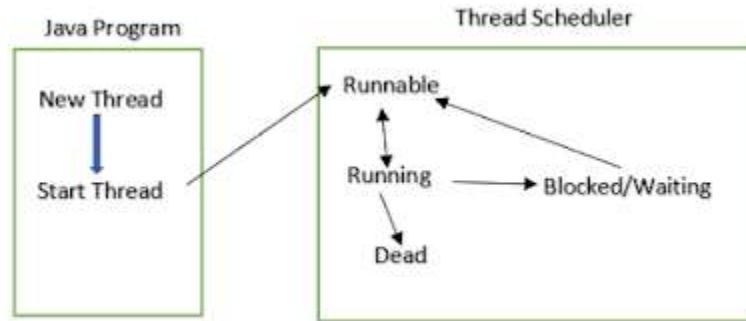
This is a tricky question. Actually interviewer wants to see the presence of mind.

Functional interfaces in Java 8 can contain only 1 abstract method. So, lambda expression will not operate on that interface, as it is not functional interface.

4. I want to send a collection to some method as parameter and want to make sure that this collection cannot be updated. How I'll do that?

In this case, we can send unmodifiable collection as the method parameter. In Collections class, we have methods like unmodifiableCollection(), unmodifiableMap(), unmodifiableList() etc. Using these, we can create and send unmodifiable collection as method parameter.

5. What is lifecycle of a thread?



Java Interview @ Rocketalk

1.Can we use String in switch case?

We can use String in switch case from java 7.

Note: switch uses `string.equals()` method to compare the passed value with case values, so make sure to add a NULL check to avoid `NullPointerException` on the string passed.

According to java 7 documentation , java compiler generates more efficient bytecode for String in switch statement than chained if-else-if statements.

2. Why char array is preferred over String for storing password?

String is immutable and final and it's value cannot be changed. So it's a security risk, as it takes time to garbage collect that string value and it resides in memory.

So, always use char array so that it's value can be changed to null.

Also , if string object is printed by mistake in some logs statement, then it prints the actual value, while char array prints memory location.

3.Why String is popular HashMap key?

String is immutable and it's hashCode is calculated once and doesn't need to calculate again. It makes it a good candidate as HashMap key.

4.Find maximum repetitions of a number in integer array.

First create a HashMap, a int variable max and an int variable temp.

HashMap: Stores key-value pairs

max : maximum count of occurrences of a number

temp: number with maximum occurrences till time

Steps to solve this problem are:

- Check whether a key exists in HashMap or not. If it exists , then get its count value, increment it by 1 and put it back and if this new incremented value is > max, then update max with this new value. And also put this new key in temp variable.

- This way, keep iterating and follow step 1 and atlast value stored in temp variable will be the key with max occurrences and integer max will give total maximum count of that key.

5.What are the technologies required in Web application development?

In Web application development, technologies are required on client side and server side.

Below is the list of technologies required:

Client side development: HTML, CSS, JavaScript, Angular, Ajax

Server side development: PHP, Java , .NET, Perl, Python

6.What is the difference between Web server and Application server?

Web server is basically used to provide static content. While Application server contain the main business logic.

The clients for Web server are Web Browsers, Mobile app etc.

The clients for Application server are Web Server, Application servers, Mobile devices etc.

Most of the application servers also contain Web server as integral part of the system.

Examples of Web servers are : Tomcat, JBoss

Examples of Application servers are : Weblogic, WebSphere etc.

Java Interview @ Aricent

1.When does deadlock occur?

- Due to nested synchronized block
- Due to calling of synchronized method from another synchronized method
- Trying to get lock on two different objects

Code that causes Deadlock:

```
public class DeadlockDemo{

    public void method1(){

        synchronized(String.class){

            synchronized(Integer.class){

            }

        }

    }

    public void method2(){

        synchronized(Integer.class){

            synchronized(String.class){
```

```

    }
}

}
}

```

2.What is LiveLock?

When all the threads are blocked or unable to proceed due to unavailability of required resources, then it is known as LiveLock.

1. It occurs when all threads call `Object.wait(0)` on an object with 0 as parameter. The program is live-locked and cannot proceed until one or more threads call `Object.notify()` or `Object.notifyAll()` on the relevant objects.
2. When all the threads are stuck in infinite loops.

3.Is there any way to find a deadlock has occurred in java?

Yes. There is a way to find it.

From JDK 1.5, we have `java.lang.management` package to diagnose and detect deadlocks.

`java.lang.management.ThreadBean` interface is management interface for the thread system of JVM.

It has methods like `findMonitorDeadlockedThreads()` and `findDeadlockedThreads()`.

4. Sort list of Employee objects using Designation, then age and then salary [nth level sorting]

```

public class ComparatorChain implements Comparator<Employee>{

    private List<Comparator<Employee>> listComparators;

    public ComparatorChain(Comparator<Employee>... comparators){
        this.listComparators = Arrays.asList(comparators);
    }

    @Override
    public int compare(Employee emp1, Employee emp2){

        for(Comparator<Employee> comparator : listComparators){
            int finalResult = comparator.compare(emp1, emp2);
            if(finalResult !=0){

                return finalResult;
            }
        }
        return 0;
    }

}

```

DesignationComparator.java:

```

public class DesginationComparator implements Comparator<Employee>{

```



```

@Override
public int compare(Employee emp1, Employee emp2){

    return emp1.getDesignation().compareTo(emp2.getDesignation());
}

}

```

AgeComparator.java

```

public class AgeComparator implements Comparator<Employee>{

    @Override
    public int compare(Employee emp1, Employee emp2){

        return emp1.getAge() - emp2.getAge();
    }

}

```

SalaryComparator.java:

```

public class SalaryComparator implements Comparator<Employee>{

    @Override
    public int compare(Employee emp1, Employee emp2){

        return emp1.getSalary() - emp2.getSalary();
    }

}

public class ListObjectsComparisonAndSortingExample{

    public static void main(String[] args){

        List<Employee> employees = new ArrayList<Employee>();

        employees.add(new Employee("Mittal", "Developer", 35, 100000));
        // Add more Employee objects

        Collections.sort(employees, new ComparatorChain(
            new DesignationComparator(),
            new AgeComparator(),
            new SalaryComparator()
        ));

    }

}

```

5. How to change logs from Info to Debug using log4j?

In Log4j, we have multiple methods like debug(), info() , error() which can be called based on some condition and after that this Log API will print only that types of logs.

So, whenever we need to change logs from Info level to Debug level, we can easily do that by switching the call from info() to debug() as shown in example below:

```
static Logger logger = Logger.getLogger(MyClass.class.getName()); // Creating logger instance
```

```
logger.info("info"); // Suppose , initially it was printing info logs
```

```
logger.setLevel(Level.DEBUG); From now on, only debug, info, warn, error and fatal logs will be printed.  
But trace logs will not get printed.
```

```
logger.debug("debug");
```

```
logger.error("error");
```

```
logger.trace("trace");//it will not get printed.
```

6.What is the Log4j log level hierarchy order?

	FATAL	ERROR	WARN	INFO	DEBUG	TRACE	ALL
OFF							
FATAL	X						
ERROR	X	X					
WARN	X	X	X				
INFO	X	X	X	X			
DEBUG	X	X	X	X	X		
TRACE	X	X	X	X	X	X	
ALL	X	X	X	X	X	X	X

From above diagram, we can see that for WARN, FATAL, ERROR and WARN are visible.
And for OFF, nothing will be visible.

7.What is a Daemon thread? How it works?

Daemon thread acts like service providers for other threads running in the same process.

Daemon threads will be terminated by JVM when there are no other threads running, it includes main thread of execution as well.

To specify that a thread is a Daemon thread, call the setDaemon() method with the argument true.

To determine if a thread is a daemon thread, use the accessor method isDaemon().

Daemon threads are used to provide background support to the user threads.

Example of a daemon thread is Garbage Collection thread. gc() method is defined in System class that is used to send request to JVM to perform garbage collection.

```
public class DaemonThread extends Thread{
```

```
    public DaemonThread(){  
        setDaemon(true);  
    }  
}
```

```
    public void run(){
```

```

        System.out.println("Is this thread Daemon? - "+isDaemon());
    }

    public static void main(String[] args){

        DaemonThread dt = new DaemonThread();
        dt.start();
    }
}

```

8. Describe two code cases where Race condition occur in java?

Two code scenarios where race condition occur are:

- Check and Act race condition
- Read. Modify, Update race condition

Check and Act race condition:

In this, we take example of creating Singleton instance.

```

if(instance == null){

    return class.getInstance();
}

```

Here, in this code, Suppose we have two threads T1 and T2. When T1 crosses if check, then it goes inside and CPU switches to T2. Now, it T1 is taking more time to create instance, then T2 also checks if statement and find instance as null. It also goes inside if() check and creates another instance.

So 2 instances will be created for Singleton instance.

9. How does Time complexity for get() and put() methods in HashMap is O(1)?

Whenever we search for a key in HashMap or HashSet, it follows these steps:

- Calculate hashcode for the key using their own hash() method.
- This key acts as a m/m address and it is used to find array index/bucket location.
- Then entries in LinkedList are compared.

As, m/m address from array can be get in one step, that's why it is O(1).

But in case, there are 1000 entries in LinkedList in a bucket, then it is not O(1). Then it is based on number of entries in LinkedList. So, in worse case, it is O(n).

10. What is the difference between map and flatMap in Java 8?

map() method is used to map an object or entry in stream to some other value. While flatMap() method, first applies map() on the entries and then flatten the result.

e.g.:

Suppose we have a string array with entries :

123 | Praveen Oruganti

Blog: <https://praveenorugantitech.blogspot.com>

Facebook Group: <https://www.facebook.com/groups/268426377837151>

Github repo: <https://github.com/praveenoruganti>

Email : praveenorugantitech@gmail.com

```
String[] strArray = {"12" , "46"};
```

And we have to find all the permutations of these strings.

So output with map() method will be : [12, 21] , [46, 64].

While output with flatMap() method will be : [12,21,46,64]

Let's take another example:

```
List<Integer> evens = Arrays.asList(2,4,6,8);  
List<Integer> odds = Arrays.asList(1,3,5,7);  
List<Integer> primes = Arrays.asList(5,7,11,13,17);
```

```
List numbers = Stream.of(evens, odds, primes).flatMap(list-> list.stream()).collect(Collectors.toList());
```

```
System.out.println("List = "+numbers);
```

O/P: 2,4,6,8,1,3,5,7,5,7,11,13,17

Note: This output is possible only with use of flatMap() method.

11.What change has been done in HashMap performance in java 8 and how does that change is implemented?

In HashMap, a customized LinkedList is used for each entry in underlying bucket. From Java 8, if length of LinkedList gets increased to some threshold, then this LinkedList is converted to Binary Search Tree, which increases performance by reducing search time.

This Binary search tree is constructed using hashcode values of keys. If the hashcode values are different then they are arranged in left or right subtrees. But if the hashcode value is same, in that case there is no such performance improvement.

12:What is the difference between PUT and PATCH?

PUT is used to completely replace a resource at given URI.

While PATCH is used just for partial updation of a given resource at given URI.

Another difference is that while using PUT, we have to send the full payload as the request, while using PATCH , we can only send the parameters which we want to update.

13.Draw CI/CD pipeline process that you follow in your company.



Explanation:

First part is Developer Machine where all the source code and Database scripts are written. Now all these source code/DB scripts files are pushed to SCM[Source Control Management] like Git/GitHub/GitLab/Azure Repo/Bitbucket etc.

From there it goes to Build pipeline where all this code is compiled and generate a package file for the application [.dll or .jar].

Build Pipeline:

From here CD pipeline starts.

To build the package files, commands like "mvn package install" or "mvn build" are run .

Also all the unit tests run here.

It also runs static code analyzer tool like Sonarqube.

It also includes configuration for generating scripts.

Release Pipeline:

Here built package files from build pipeline are deployed on multiple servers like QA server, Development server and Production server.

All the Integration tests run here.

QA team takes the build from here and run User Acceptance Testing.
Also deploys SQL scripts on Database server.

14.What is the upper limit for a payload to pass in the POST method?

GET appends data to the service URI. But it's size should not exceed the maximum URL length.
However POST doesn't have any such limit.

So , theoretically , a user can pass unlimited data as the payload to POST method. But, if we consider a real use case, then sending POST with large payload will consume more bandwidth. It will take more time and present performance challenges to the server. Hence, a user should take action accordingly.

15.Suppose our REST API requires an int value and we send String "abcd" in query parameter , then what response we get?

We get error : Not found.

```
"timestamp": "2019-01-17T18",  
"status" : 404,  
"error" : "Not found",
```

```
"message" : "JSON parse error: Cannot deserialize value of type int from String \"abcd\": not a valid  
integer value;  
nested exception is com.fasterxml.jackson.databind.exc.InvalidFormatException:
```

16.What are Spring Core annotations?

Spring core annotations are following:

- @Qualifier
- @Autowired
- @Required
- @ComponentScan
- @Configuration
- @Bean
- @Lazy
- @Value

17. Explain Spring Data JPA vs Hibernate

JPA: Java Persistence API which provide specification for creating, deleting, persisting and data management from java objects to relations [tables] in database.

Hibernate: There are various providers which implement JPA. Hibernate is one of them . So , we have

126 | Praveen Oruganti

Blog: <https://praveenorugantitech.blogspot.com>

Facebook Group: <https://www.facebook.com/groups/268426377837151>

Github repo: <https://github.com/praveenoruganti>

Email : praveenorugantitech@gmail.com

other provider as well e.g.: Eclipse Link etc.

Spring Data JPA: This is another layer on top of JPA which spring provides to make coding easier.

Java Interview @ WDTS [Walker Digital Table Systems]

1.What is Reentrant Lock?

ReentrantLock is a mutually exclusive lock with the same behavior as the intrinsic/implicit lock accessed via synchronization.

ReentrantLock, as the name suggests , possesses reentrant characteristics. That means , a thread that currently owns the lock can acquire it more than once without any problem.

ReentrantLock forces one thread to enter critical section and queues all other threads to wait for it to complete.

Basically, java 5 introduces the concept of a lock. Lock and ReadWriteLock are interfaces in java 5.

And their implementations are ReentrantLock and ReentrantReadWriteLock.

2.How to handle service fallback in MicroServices?

We can use Circuit breaker pattern implementation Netflix Hystrix for handling fallback in microservices.

Actually, in Microservices based arhitecture, there are many applications running in different processes on different machines. And any of these service may be down at some point of time. So to avoid sending all requests to this misroservice, Circuit breaker pattern can be used.

3.What are REST Call End Points?

e.g.: <https://api.github.com/users/zellwk/repos?sort=pushed>

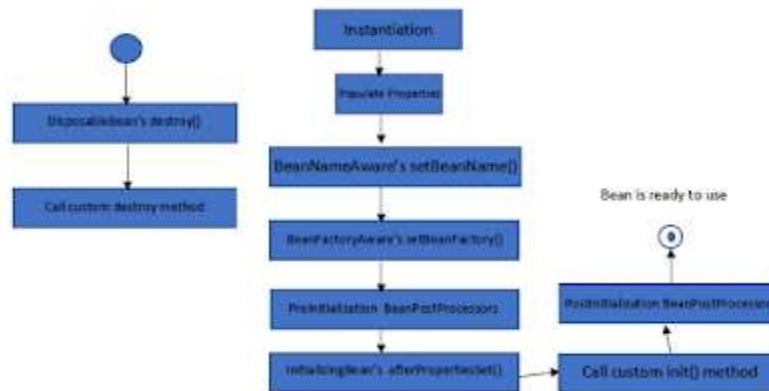
In above URL, <https://api.github.com> is root endpoint

/users/zellwk/repos is path which determines the resource we request for.

The last part of an endpoint is query parameters. Query parameters give us the option of modifying the request with key-value pairs. They always begin with a question mark [?]. Each parameter pair is then separated with an ampersand [&] , like this:

?query1=value1&query2=value2

4.Explain Lifecycle of Spring Bean.



5.Explain the steps of sending a request from browser and handling it at server side?

Steps :

- First request from browser [client side] will be received by DispatcherServlet
- DispatcherServlet will take the help of HandlerMapping and get to know the Controller class name associated with the given request.
- So now, request transfers to the Controller and then controller will process the request by executing appropriate methods and returns ModelAndView object back to the DispatcherServlet.
- Now DispatcherServlet send the view name to the ViewResolver to get the actual view page.
- Finally DispatcherServlet will pass the Model object to the view page to display the result.

6. Name all the HTTP verbs?

HTTP verbs are:

- **GET**
- **PUT**
- **POST**
- **DELETE**
- **HEAD**
- **PATCH**

7. What is a DispatcherServlet?

DispatcherServlet handles all HTTP requests and responses.

It is front controller in Spring MVC based applications. DispatcherServlet uses it's own WebApplicationContext which is a child of ApplicationContext created by ContextLoaderListener.

8.When we override hashCode() method, then how to retrieve the actual default hash code value for that object?

Just use System.identityHashCode(object);

The value returned by default implementation of hashCode() is called identity hash code.

Identity hashcode is usually the integer representation of the memory address.

HashCode of an object is a 32-bit signed int that allows an object to be managed by hash-based data structure.

9. Write a logic of two threads printing question and answer one-by-one.

```
class Chat{

    boolean flag = false;

    public synchronized void Question(String message){
        if(flag){
            try{
                wait();
            }
            catch(InterruptedException ie){
                ie.printStackTrace();
            }
        }

        System.out.println(message);
        flag = true;
        notify();
    }

    public synchronized void Answer(String message){
        if(!flag){
            try{
                wait();
            }
            catch(InterruptedException ie){
                ie.printStackTrace();
            }
        }

        System.out.println(message);
        flag = false;
        notify();
    }
}
```

```
class Question implements Runnable{

    Chat chat;
    String[] str = {"Hi", "How are you?", "I'm also doing fine"};

    public Question(Chat c1){
        this.chat = c1;
        new Thread(this, "Question").start();
    }

    public void run(){
```

```

        for(int i = 0; i< str.length; i++){
            c1.question(str[i]);
        }
    }
}

class Answer implements Runnable{

    Chat chat;
    String[] str = {"Hi", "I'm good", "Great"};

    public Answer(Chat c1){
        this.chat = c1;
        new Thread(this, "Answer").start();
    }

    public void run(){

        for(int i=0; i<str.length; i++){
            c1.answer(str[i]);
        }
    }
}

public class Test{

    Chat chat = new Chat();
    new Question(chat);
    new Answer(chat);
}

```

Technical Architect interview @ Xebia

1.What is distributed transaction? What architectures are available for performing distributed transactions?

Simple Transaction:

A transaction is a logically atomic unit of work which may span multiple database queries. They ensure locks over resources they acquire in order to maintain the consistency of the database.

All this is done with the help of ACID properties.

Distributed Transaction:

A distributed transaction would do what a transaction would do but on multiple databases.

In a distributed scenario, the architecture would be split into services like handling service, payment gateway service etc who would house a respective database and all the actions would be performed in the respective databases.

One way of doing is the Two phase commit.

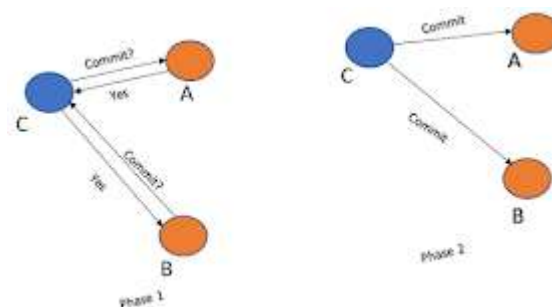
In Two phase commit, we have a controlling node which houses most of the logic and we have a few participating nodes on which the actions would be performed.

Prepare Phase:

In this phase, controlling node would ask all the participating nodes, if they are ready to commit. The participating nodes would then respond in yes or no.

Commit Phase:

Then, if all the nodes have replied in affirmative, the controlling node would ask them to commit, else if one node replies in negative, it'll ask them to rollback.



Drawbacks of Two Phase Commit:

- Whole logic gets concentrated in single node and if that node goes down, the whole system fails.
- The whole system is bound by slowest node, since any ready node will, have to wait for response from the slower node which is yet to confirm its status.

Another approach is Saga Pattern:

Saga pattern is one of the ways by which we ensure data consistency in a distributed architecture. But there is absence of atomicity in Saga pattern.

For example, in the e-commerce website, we first do the transaction pertaining to the customer selection. Once that is complete, we start selecting a product and so on. So all these constituent transactions together will be known as a Saga.

A successful Saga looks something like this:

- Start Saga
- Start T1
- End T1
- Start T2
- End T2
- Start T3
- End T3
- End Saga

But things do not go straight. Sometimes, we might not be in position to perform a transaction in the middle of the saga. At that point, the previously successful transactions would've already committed. So, apart from not continuing with the Saga, we also need to undo whatever changes we may have already committed. For this, we apply compensatory transactions. Thus for each transaction T_i , we implement a compensatory transaction C_i , which tries to semantically nullify T_i .

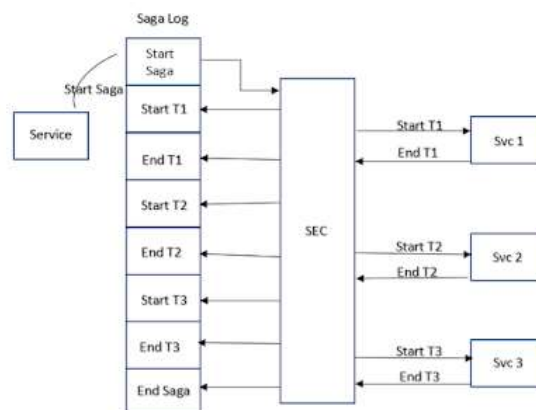
It's not always possible to get back to the same state. e.g. if T_i involved in sending the email, we can't really undo that. So, we send a corrective email which semantically undoes T_i . So a failed saga looks something like this.

- Begin Saga
- Start T1
- End T1
- Start T2
- Abort Saga
- Start C2
- End C2
- Start C1
- End C1
- End Saga

There can be various ways of implementing Saga pattern, but to actually implement it in a scalable manner, we can introduce a central process aka Saga Execution Controller or SEC.

SEC is merely a process that handles the flow of execution of transactions or compensatory transactions. It helps us centrally locate the logic of execution.

Another important constituent in order to implement our form of saga pattern is Saga log. Just like a database log, it's a basic, durable but distributed source of information. Every event that SEC executes is logged in Saga log. A good example of that could be a Kafka log.



What if our SEC fails down. The solution for that is , we can launch another SEC and it will start from where the first SEC left off.

2.How to write custom annotation?

Java annotations are mechanism for adding metadata information to our source code.

Annotations were added in java 5.

Annotations offer an alternative to the use of XML descriptors and marker interfaces.

We can attach them to packages, classes, interfaces and methods but they have no effect on the execution of the program.

@interface keyword is used to declare an annotation.

Class level annotation:

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.Type)
public @interface JsonSerializable{
}
```

In above code, I have defined a class level annotation. @Target and @Retention are used to define metadata [target and scope respectively].

ElementType.TYPE means, it is applied on types (classes).

RetentionPolicy.RUNTIME means, annotation has runtime visibility,

Field Level annotation:

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.FIELD)
public @interface JsonElement{
    public String key() default "";
}
```

This annotation declares one String parameter with name key and an empty string as the default value.

Method Level annotation:

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.METHOD)
public @interface Init{
}
```

When creating custom annotation for methods, keep in mind that the methods (on which these annotation are applied) cannot have any parameter and cannot throw any exception.

Now, we can simply apply these annotations on classes, fields and methods as shown below:

@JsonSerializable

```

public class Student{

    @JsonElement
    public String name;

    public String age;

    @Init
    private void getName(){

    }

}

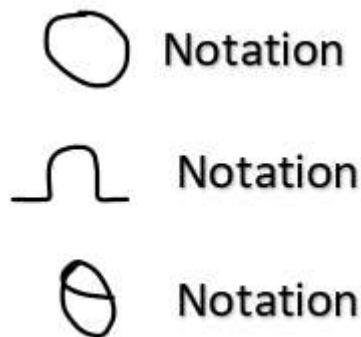
```

This getName() method is called before serialization. And only name field be serialized. And as we have made this method private, so that we can't initialize our object by calling it manually.

3.What are Asymptotic notations? What is their use?

Asymptotic notations are used to calculate running time complexity of an algorithm.

Below are the commonly used asymptotic notations:

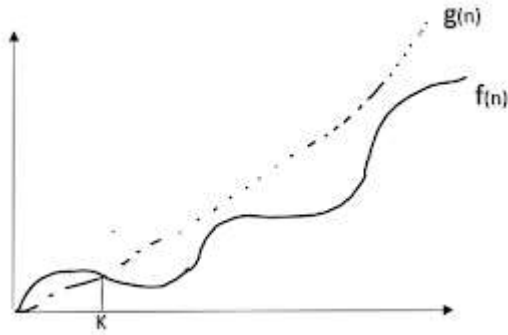


Big Oh notation , O:

The notation $O(n)$ is the formal way to express the upper bound of an algorithm's running time. It measures the worst case time complexity or the longest amount of time an algorithm can possibly take to complete.

For example , for a function $f(n)$

$o(f(n)) = \{g(n) : \text{there exists } c > 0 \text{ and } n_{\text{theta}} \text{ such that } f(n) \leq c \cdot g(n) \text{ for all } n > n_{\text{theta}}.\}$

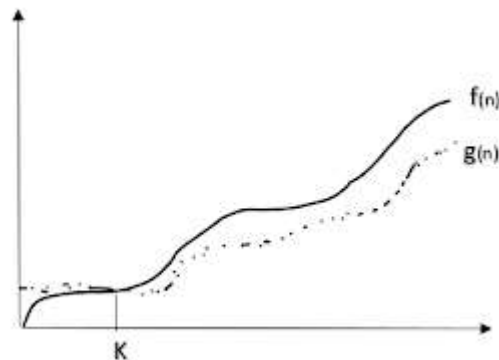


Omega Notation:

The notation $\Omega(n)$ is the formal way to express the lower bound of an algorithm's running time. It measures the best case time complexity or the best amount of time an algorithm can possibly take to complete.

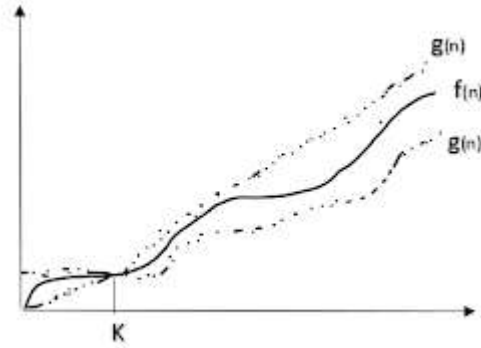
For example, for a function f(n):

$\Omega(f(n))$ if $g(n)$: There exists $c > 0$ and n_0 such that $g(n) \leq c \cdot f(n)$ for all $n > n_0$.



Theta Notation:

The notation $\theta(n)$ is the formal way to express both the lower bound and upper bound of an algorithm's running time. It is represented as follows:



4. Write some HTTP error codes with their meaning.

1XX : Informational

- 100 : Continue

2XX : SUCCESS

- 200 : OK
- 201 : Created
- 202 : Accepted
- 204 : No Content

3XX : Redirection

4XX : Client Error

- 400 : Bad Request
- 401 : Unauthorized
- 402 : Payment Required
- 403 : Forbidden
- 404 : Not Found

5XX : Server Error

- 500 : Internal Server Error
- 501 : Not Implemented
- 502 : Bad Gateway
- 503 : Service Unavailable [Website's server is simply not available]

5. What are the different ways of managing REST API versioning? And how do you manage the versioning for some update() method for the new client?

There are multiple ways of managing the REST API versioning:

- Through a URI path: We include the version number in the URI path of the endpoint.
e.g.: api/v1/shares
- Through query parameters: We pass the version number as a query parameter with a specified name e.g.: api/shares?version=1

- Through custom HTTP headers: We define a new header that contains the version number in the request.
- Through a content negotiation: The version number is included in the "Accept" header together with the accepted content type.

Suppose we have an update method with endpoint given below. Initially, it was accessible under /v1.0 path. Now, it is available under /v1.1/{id} path.

```
@PutMapping("/v1.0")
public ShareOld update(@RequestBody ShareOld share){

    return (ShareOld)repository.update(share);

}

@PutMapping("/v1.1/{id}")
public ShareOld update(@PathVariable("id") long id, @RequestBody ShareOld share){

    return (ShareOld)repository.update(share);

}
```

And if have a GET mapping that remains same for both versions, then , we can write GET mapping as:

```
@GetMapping("/v1.0/{id}", "/v1.1/{id}")
public Share findByIdOld(@PathVariable("id") long Id){

    return (Share)repository.findById(id);

}
```

In this, as we have 2 different versions of our API, we need to create two Docket objects using api() method call.

e.g.:

```
@Bean
public Docket swaggerShareApi10(){

    return new Docket(DocumentationType.SWAGGER_2)
        .groupName("share-api-1.0")
        .select().apis(RequestHandlerSelectors.basePackage("pl.piomin.service
s.versioning.controller"))
        .paths(regex("/share/v1.0.*"))
        .build()
        .apiInfo(new ApiInfoBuilder().version("1.0").title("Share API").description("Documentation Share API
v1.0"));

}

@Bean
public Docket swaggerShareApi11(){

    return new Docket(DocumentationType.SWAGGER_2)
        .groupName("share-api-1.1")
```

```

        .select()                .apis(RequestHandlerSelectors.basePackage("pl.piomin.services.versioning.controller"))
        .paths(regex("/share/v1.1.*"))
        .build()
        .apiInfo(new ApiInfoBuilder().version("1.1").title("Share API").description("Documentation Share API v1.1"));
    }

```

Now, when we launch Swagger UI, it shows us the dropdown displaying both versions and we can easily switch between them.

6.What is the difference between MongoDB and Cassandra?

- MongoDB is free and open source , cross platform , document oriented database system. While Cassandra is open source, distributed and decentralized , column-oriented database system.
- MongoDB does not have triggers while Cassandra has triggers.
- MongoDB has secondary indexes while Cassandra has restricted secondary indexes.
- Cassandra uses a selectable replication factor while MongoDB uses a master-slave replication factor.
- MongoDB is used when we need to store data in JSON style format in some documents which consists of key-value pairs. While Cassandra is used as decentralized database for big data.

7.What are the steps for using MongoDB in java project?

Steps for using MongoDB are described below:

- Need to add following dependencies in build.gradle file:
- compile group: 'org.mongodb', name: 'mongodb-driver', version: '3.11.0'
- compile group: 'org.mongodb' name: 'bson', version: '3.11.0'
- compile group: 'org.mongodb' name:'mongodb-driver-core', version: '3.11.0'
- compile group: 'org.mongodb' name:'mongo-java-driver', version: '3.11.0'
- Specify the mongo DB URL and mongo DB name in application.yml file.
- Create MongoClient instance using MongoCredential, MongoClientOptions.Builder and a list of Mongo server addresses.
- Create MongoCollection bean.
- Create a class implementing HealthIndicator interface overriding the health() method.
- Now in the DAO implementation class, call MongoCollection methods such as find(), insertOne(), countDocuments(), count(), bulkWrite() etc.

Coding interview questions in java

1.Write the code for custom BlockingQueue in java.

There are two custom implementations of BlockingQueue: Using Synchronization and using Lock and Condition objects.

Using synchronization:

```

public class CustomBlockingQueue{

    int putPtr, takePtr;
    Object[] items;
    int count = 0;

    public CustomBlockingQueue(int length){

        items = new Object[length];
    }

    public void synchronized put(Object item){

        while(count == items.length()){
            try{
                wait();
            }
            catch(InterruptedException e){

            }

            items[putPtr] = item;
            if(++putPtr == items.length)
                putPtr = 0;

            count++;
            notifyAll();

        }
    }
}

```

```

    }

    public void synchronized take(){

        while(count == 0){
            try{
                wait();
            }
            catch(InterruptedException e){

            }
        }
        Object item = items[takePtr];
        if(++takePtr == items.length)
            takePtr = 0;

        --count;
        notifyAll();
    }
}

```

Using Lock and Condition object:

```

public class CustomBlockingQueue{

    final Lock lock = new ReentrantLock();
    final Condition notFull = lock.newCondition();
    final Condition notEmpty = lock.newCondition();

```

```

public Object[] items = new Object[100];
int putPtr, takePtr, count;

public void put(Object item) throws InterruptedException{
    lock.lock();
    try{
        while(count == items.length)
            notFull.await();
        items[putPtr] = item;
        if(++putPtr == items.length)
            putPtr = 0;
        ++count;
        nonEmpty.signal();
    }
    finally{
        lock.unlock();
    }
}

```

```

public Object take() throws InterruptedException{
    lock.lock();
    try{
        while(count == 0)
            notEmpty.await();
        Object x = items[takePtr];
    }
}

```

```

        if(++takePtr == items.length)
            takePtr = 0;
        count--;
        notFull.signal();
        return x;
    }
    finally{
        lock.unlock();
    }
}
}

```

2. Write the code for custom ArrayList in java.

```

public class CustomArrayList{

    public Object[] items;

    public int ptr;

    public CustomArrayList(){
        items = new Object[100];
    }

    public void add(Object item){
        if(items.length - ptr == 5){
            increaseListSize();
        }
    }
}

```

142 | Praveen Oruganti

Blog : <https://praveenorugantitech.blogspot.com>

Facebook Group : <https://www.facebook.com/groups/268426377837151>

Github repo : <https://github.com/praveenoruganti>

Email : praveenorugantitech@gmail.com

```

        items[ptr++] = item;

    }

    public Object get(int index){
        if(index < ptr)
            return items[index];
        else{
            throw new ArrayIndexOutOfBoundsException();
        }
    }

    private void increaseListSize(){
        items = Arrays.copyOf(items, items.length*2);
    }
}

```

3.Write the code for implementing own stack.

```

public class CustomStack{

    private int maxSize;
    private long[] stackArray;
    private int top;

    public CustomStack(int size){
        maxSize = s;
        stackArray = new long[size];
        top= -1;
    }
}

```

```

    }

    public void push(long l){
        stackArray[++top] = l;
    }

    public long pop(){
        return stackArray[top--];
    }

    public long peek(){
        return stackArray[top];
    }

    public boolean isEmpty(){
        return (top == -1);
    }
}

```

4. Write the code for implementing own Queue.

```

public class CustomQueue{

    private int[] arrQueue;
    private int front , rear;

    public CustomQueue(int size){

```



```
arrQueue[size] = new int[];  
front = rear = -1;  
}
```

```
public void insert(int item){
```

```
    if(rear == -1){  
        front = rear = 0;  
    }
```

```
    else{  
        rear++;  
    }
```

```
    if(arrQueue.length == rear)  
        increaseQueueSize();
```

```
    arrQueue[rear] = item;
```

```
}
```

```
public int remove(){
```

```
    if(arrQueue.length == 0)  
        throw new NoSuchElementException();
```

```
    int elem = arrQueue[front];
```

```
    if(front == rear){  
        front = rear = -1;  
    }
```

```

        else
            front++;

        return elem;
    }

    public void increaseQueueSize(){
        arrQueue = Arrays.copyOf(arrQueue, arrQueue.length*2);
    }
}

```

Capgemini Java 3 to 8 Years Experience Interview

1. Tell me about yourself ?

Ans : Give your brief introduction.

2. Explain about your current project?

Ans : It's quite easy to describe your projects and your key role on this project. But, be careful and get ready about the functionality when you are describing your working module / part of the project. Show your confidence that you have done the major part and you can face challenges in future.

3. How many types of literals are there in JAVA ?

Ans : The literals means the value you are assigning to variable. You can specify the below types of literal in java. As per the primitive data types (int, short, long, float, double, boolean, char, etc) there is respective literal. Some literal needs to be ended with a specific character. Read More.

```

        long var=20L; //specify L or l for long literal
        int var=20; // If not mentioned any character then its can be short or int
        char var='A';
        float var=10.44f; //specify f or F for float literal
        double var=10.44;
        boolean var=true; //or false
    
```

4. What is meant by Garbage collection ?

Ans : Garbage collection is a automatic feature of java for cleaning the unused object from heap. It helps

146 | Praveen Oruganti

Blog : <https://praveenorugantitech.blogspot.com>

Facebook Group : <https://www.facebook.com/groups/268426377837151>

Github repo : <https://github.com/praveenoruganti>

Email : praveenorugantitech@gmail.com

to developer for releasing the reserved memory without any extra effort by developer. It helps developer to save time and extra mental tension for handling object allocation in memory. When there is no reference to an object found, it will clean that object from memory . You can run the garbage collection explicitly by using `System.gc()` .

5. Difference between `String s= new String ();` and `String s = "Hi Dude";` ?

Ans : Both statements are different to each other. Always 'new' keyword is used to create object.

`String s=new String();` // This statement creates new object in heap. S is the object here.
`String s="Hi Dude"` ; // This statement do not create object, its creating reference and its storing in String Constant Pool.S is the reference here.

6. What is singleton class? where is it used ?

Ans : View Answer

7. What is the difference between JSP and Servlets ?

Ans : As simple JSP is pre-compiled but Servlets are not. JSP is specially use for displaying/populating the data on browser. If we take an example of MVC architecture JSP plays the role of View (V). But, Servlets are used to handle the request and process the business logic. In MVC architecture Servlets are knows as Controller (C) .

8. What is the difference in using `request.getRequestDispatcher()` and `context.getRequestDispatcher()`?

Ans : Both are taking String parameter, but `request.getRequestDispatcher()` will dispatch the request inside the application. Where as `context.getRequestDispatcher()` will dispatch the request outside the context also. If you are using absolute path for dispatching then both are similar.

9. How the JSP file will be executed on the Server side ?

Ans : First its converted to java file. Then its compiled by java compiler and creates the .class file.Now once you get the .class file you can execute it. Internally a JSP converts into respective servlet.

Conversion => Compilation => Execution

10. What is ActionServlet ?

Ans : ActionServlet provides the controller in struts application with MVC (Model-View-Controller) Model 2.

ActionServlet is a sub class of `javax.servlet.http.HttpServlet`.It has few methods like `doGet()`,`doPost()`,`destroy()`,etc.

11. What is Struts Validator Framework ?

Ans : Struts provides a convenient way to validate. Basically we are using two xml configuration file for configuring such as `Validator.xml` and `validation-rule.xml` . `validation-rule.xml` defines the rule of validation like number format validation,email validation. Apart from this `ActionForm` having `validate()` method which we can implement for validating the form data.

12. What is the difference between the Session and SessionFactory in hibernate ?

Ans : Session and SessionFactory are playing a prominent role in hibernate. SessionFactory is used to create Session and its created once during starting of the application. You can have only one SessionFactory per application. SessionFactory is also called 2ndlevel cache. But, Session could be many per application. Session is being created by using SessionFactory object. Session is called 1st level cache.

13. What is HQL ?

Ans : HQL stands for Hibernate Query Language. Its fully object oriented and quite similar with SQL. It supports association and joins for effective entity relationship.

Java Interview @ Fresher Level

1.What is Polymorphism and many types of polymorphism are there?

It is one of the OOPS concepts that allows us to perform single action in different ways. Polymorphism is the capability of a method to do different things based on the object that it is acting upon.

Types of Polymorphism:

- Runtime Polymorphism or Dynamic Polymorphism [Method Overloading]
- Compile time polymorphism or static polymorphism [Method Overloading]

2.What is static binding and dynamic binding?

Static binding or Early binding:

The binding which can be resolved at compile time by compiler is known as static or early binding. The binding of static, private and final methods is compile-time. Because these methods can't be overridden and the type of the class is determined at the compile time.

Dynamic binding or Late binding:

When compiler is not able to resolve call/binding at compile time, such binding is known as dynamic binding or late binding. Method overriding is the perfect example of dynamic binding.

3.What are the use of super keyword?

The super keyword refers to the object of immediate parent class.

Use of super keyword:

- 1). To access the data members of parent class when both the child and parent class have member with same name.
- 2). To explicitly call no-arg and parameterized constructor of parent class.
- 3). To access the method of parent class when child class has overridden that method.

4.What is the difference between implements Runnable and extends Thread?

There are few differences which are described as below:

- Inheritance Option
- Loosely-coupled
- Functions overhead
- Thread represents "how a thread of control runs". And Runnable represents "what a Thread runs".

Inheritance Option: Means, If class implements Runnable interface, it can extends one class as well. But if class extends Thread, then it cannot extend any other class.

Loosely-coupled: Runnable object can be used in multiple threads. And a Runnable can be changed independently.

Functions Overhead: If we extends Thread class just to run a task, then there will be overhead of all Thread functions.

5.What is difference between user thread and daemon thread?

When we create a thread in java application, it is called user thread.

Daemon thread is a thread that acts as a service provider for other threads running in the same process. e.g. GC.

Daemon thread is terminated by JVM, when there is no other thread running.

6.Why will we take Daemon thread in our application?

If we take a user thread and if it is polling some websites continuously, then if we exit the app, then user thread keeps running and it will not let JVM shutdown.

But, if we make it a daemon thread , then this thread will be terminated by JVM, when there is no other thread running.

7.What do you understand about thread priority?

Every thread has a priority. Usually higher priority thread gets precedence in execution , but it depends upon Thread Scheduler implementation that is OS dependent.

We can specify thread priority using setPriority() method. We can use getPriority() method to get priority.

Thread priority is an int whose value varies from 1 to 10 where 1 is the lowest priority thread and 10 is the highest priority thread.

There are 3 variables defined in Thread class for priority:

- MIN_PRIORITY
- NORM_PRIORITY : This is the default priority. It has value 5. Default priority of a thread depends upon priority of parent thread.
- MAX_PRIORITY

8.What is thread scheduler and time slicing?

Thread scheduler is OS service that allocates CPU time to the available threads.
Time slicing is the process of divide the available CPU time to the available runnable threads.

9. Write down the definition of Iterable interface.

```
public interface Iterable{  
  
    public Iterator<T> iterator();  
}
```

10. How does HashSet work?

When we create HashSet instance, it internally creates instance of HashMap in its constructor. So when we add a duplicate key, that key is compared in HashMap entries. Actually, it is put in HashMap. So, when we call add(3), then the value 3 is stored as key in HashMap and some dummy value [new Object] is stored as value.

So, in HashSet add() method is defined as:

```
public boolean add(E e){  
  
    return map.put(e, PRESENT) == null;  
}
```

Note: map.put() returns null, if it adds key-value pair. If key already exists, it returns value.
So, if key is not present in HashMap, HashMap returns null and add() method of HashSet returns true. Else it returns false.

11. What copy technique is originally used by HashSet clone() method?

There are two copy techniques in every object oriented programming language: deep copy and shallow copy.

To create a clone or copy of the Set object, HashSet internally uses shallow copy in clone() method, the elements themselves are not cloned. In other words, a shallow copy is made by copying the reference of the object.

12. What is the difference between HashSet and TreeSet?

There are multiple differences between HashSet and TreeSet which are described below:

- Ordering of the elements
- Null Value: TreeSet doesn't allow null value.
- HashSet implements Set interface while TreeSet implements NavigableTreeSet interface.
- HashSet uses equals() method for comparison while TreeSet uses compareTo() method for comparison.

Note: Underlying data structure of TreeSet is Red-Black Tree which is a BST and thus is sorted. For it to be sorted, it uses comparator. The default comparator is not null safe, that's why TreeSet doesn't allow null value.

So, when we call TreeSet.add(null);, It compiles but at runtime, it throws NullPointerException.

13. How fast fail iterator come to know that the internal structure is modified?

150 | Praveen Oruganti

Blog: <https://praveenorugantitech.blogspot.com>

Facebook Group: <https://www.facebook.com/groups/268426377837151>

Github repo: <https://github.com/praveenoruganti>

Email: praveenorugantitech@gmail.com

Iterator read internal data structure (object array) directly. The internal data structure (i.e. object array) should not be modified while iterating through the collection.

To ensure this, it maintains an internal flag "mods". Iterator checks the mods flag, whenever it gets the next value (using hasNext() method and next() method). Value of mods flag changes whenever there is an structural modification. Thus indicating iterator to throw ConcurrentModificationException.

14.What is fail safe iterator?

Fail safe iterator makes copy of the internal data structure (object array) and iterates over the copied data structure.

Any structural modification done to the iterator affects the copied data structure. So, original data structure remains structurally unchanged.

Hence, no ConcurrentModificationException is thrown by the fail safe iterator.

15.What is the difference between Collections and Collection?

java.util.Collections is a class which contains static methods only and most of the methods throw NullPointerException if object or class passed to them is null.

java.util.Collection is an interface. Which is the base interface for all other collections like List, Map, Set etc.

16.What is the difference between Iterator and ListIterator?

We can use Iterator to traverse Set and List collections whereas ListIterator can be used with Lists only.

Iterator can traverse in forward direction only whereas ListIterator can be used to traverse in both the directions.

17.How many ways are there to traverse or loop Map, HashMap, TreeMap in java?

We have 4 ways to traverse:

- Take map.keySet() and loop using foreach loop
- Take map.keySet() and loop using Iterator
- Take map.entrySet() and loop using foreach loop
- Take map.entrySet() and loop using iterator.

18.Name the types of SQL databases.

There are multiple SQL databases. Few are listed below:

- MySQL
- SQL Server
- Oracle
- Postgres

19.What is the difference between MySQL and SQL Server?

There are multiple differences between MySQL and SQL server databases:

151 | Praveen Oruganti

Blog: <https://praveenorugantitech.blogspot.com>

Facebook Group: <https://www.facebook.com/groups/268426377837151>

Github repo: <https://github.com/praveenoruganti>

Email: praveenorugantitech@gmail.com

- MySQL is an open source RDBMS [owned by Oracle] , whereas SQL Server is a microsoft product.
- MySQL supports more programming languages than supported by SQL server. e.g.: MySQL supports Perl, Scheme, Tcl, Eiffel etc. which are not supported by SQL server.
- Multiple storage engine[InnoDB, MyISAM] support makes MySQL more flexible than SQL Server.
- While using MySQL, RDBMS blocks the database while backing up the data. And the data restoration process is time-consuming due to execution of multiple SQL statements. Unlike MySQL, SQL server does not block the database while backing up the data.
- SQL server is more secure than MySQL. MySQL allows database file to be accessed and manipulated by other processes at runtime. But SQL server does not allow any process to access or manipulate it's database files or binaries.
- MySQL doesn't allow to cancel a query mid-execution. On the other hand, SQL server allows us to cancel a query execution mid-way in the process.

20. Why should we never compare Integer using == operator?

Java 5 provides autoboxing/unboxing.

So, we store int to wrapper class Integer. But we should not use == operator to compare Integer objects. e.g.:

```
Integer i = 127;
Integer j = 127;
```

i == j will give true.

```
Integer ii = 128;
Integer jj = 128;
```

ii == jj will give false.

It is so because, Integer.valueOf() method caches int values ranging from -127 to 127. So, between this range, it will return same object. After that, it will create new object.

Note:

```
Integer i = 127;
Integer j = new Integer(128);
```

Now, == operator will give false. As , new operator will create a new object.

21. Why to use lock classes from concurrent API when we have synchronization?

Lock classes in concurrent API provide fine-grained control over locking.

Interfaces: Lock, ReadWriteLock

Classes: ReentrantLock, ReentrantReadWriteLock

22. What are the methods for fine-grained control?

ReentrantLock provides multiple methods for more fine-grained control:

152 | Praveen Oruganti

Blog: <https://praveenorugantitech.blogspot.com>

Facebook Group: <https://www.facebook.com/groups/268426377837151>

Github repo: <https://github.com/praveenoruganti>

Email : praveenorugantitech@gmail.com

- `isLocked()`
- `tryLock()`
- `tryLock(long milliseconds, TimeUnit tu)`

The `tryLock()` method tries to acquire the lock without pausing the thread. That is, if the thread couldn't acquire the lock because it was held by another thread, then it returns immediately instead of waiting for the lock to be released.

We can also specify a timeout in the `tryLock()` method to wait for the lock to be available:

```
lock.tryLock(1, TimeUnit.SECONDS);
```

The thread will now pause for one second and wait for the lock to be available. If the lock couldn't be acquired within 1 second, then the thread returns.

23.What is ReentrantReadWriteLock?

`ReadWriteLock` consists of a pair of locks - one for read and one for write access. The read lock may be held by multiple threads simultaneously as long as the write lock is not held by any thread.

`ReadWriteLock` allows for an increased level of concurrency. It performs better compared to other locks in applications where there are fewer writes than reads.

24.What is the difference between Lock and synchronized keyword?

Following are the differences between `Lock` and `synchronized` keyword:

- Having a timeout trying to get access to a synchronized block is not possible. Using `lock.tryLock(long timeout, TimeUnit tu)`, it is possible.
- The synchronized block must be fully contained within a single method. A lock can have its calls to `lock()` and `unlock()` in separate methods.

25.Why to use Executor framework?

We can use executor framework to decouple command submission from command execution.

Executor framework gives us the ability to create and manage threads.

There are 3 interfaces defined in executor framework:

- `Executor`
- `ExecutorService`
- `ScheduledExecutorService`

26.How many groups of collection interfaces are there?

There are two groups of collection interfaces:

- `Collection`
- `List`
- `ArrayList`
- `Vector`
- `LinkedList`
- `Queue`

- LinkedList
- PriorityQueue
- Set
- HashSet
- LinkedHashSet
- SortedSet
- TreeSet
- Map
- HashMap
- Hashtable
- SortedMap
- TreeMap

27.What is the definition of Iterable interface?

```
public interface Iterable<T>{
    public Iterator<T> iterator();
}
```