
Sapient Interview Questions

Core Java
---
1. SOLID Principles
	--> Single Responsibilty Principle
	--> Open-Closed Principle
	--> Liscov Substitution Principle
	--> Interface Segregation Principle
	--> Dependency Inversion Principle
2. Design Pattern in Java. Which design pattern you have used in your project?
--> Creational Design Pattern
	--> Factory Design Pattern
	--> AbstractFactory Design Pattern (means Factory of Factories)
	--> Builder design pattern (Step by step construction of objects)
	--> Singleton design pattern (Create single object from a bean)
	--> Prototype design pattern
--> Behavioural Design Pattern
	--> Adapter Design pattern
	--> Bridge Design pattern
	--> Composite Design pattern
	--> Decorator Design pattern
	--> Facade Design pattern
	--> Flyweight Design pattern
	--> Proxy Design Pattern
--> Structural Design Pattern
	--> Chain of Responsibilty pattern
	--> Command Pattern
	--> Interpreter Design pattern
	--> Iterator Pattern
	--> Mediator Pattern
	--> Momento Pattern
	--> Observer Pattern
	--> State Pattern
	--> Strategy Pattern (Decide which algorithm to use on runtime)
	--> Template Pattern
	--> Visitor Pattern
3. Runtime or Dynamic polymorphism
4. Abstraction
5. Create custom Immutable class
	--> Class should be final so can't be inherited.
	--> Variables should be private and final.
	--> There's no setter method to update the values later.
	--> If there's any object type variable, always return its deep copy in constructor and getter method.
6. Create custom Singleton class
	--> static variable should be created to store the Class object.
	--> Create getInstance() method which contains the logic as if the object is already present, will return the same else create new for the first time.
	--> constructor should be private.
6. Volatile Keyword

--> In Java, the volatile keyword is a modifier applied to variables. It ensures that the value of the variable is always read from and written to the main memory, rather than being cached in a thread's local memory (CPU cache).
--> Since the purpose of volatile is to control memory access for a specific variable, it only makes sense to apply it to variables. Methods and classes don't have the same kind of memory access characteristics that variables do.

7. Executor Interface, Future and CompletableFuture and its examples
--> Exception handling in CompletableFuture
--> handle(), exceptionally(), whenComplete()

8. Concurrent Execution
--> When multiple threads are executing the process concurrently or utilizing the same resource.

9. Fail safe and Fail fast iterators and its examples. What exception is thrown in case of Fail fast iterators?
--> ConcurrentModificationException thrown in case of Fail fast iterators if we modify object while iterating.
--> Fail Safe iterators are iterating over cloned copy of the object and do modifications on the original object.

10. HashMap Implementation
--> HashMap internally uses Array as bucket and if collision occurs, it uses LinkedList to store the data means its an array of LinkedList.
--> HashMap uses hash() method to get the bucket location then saves the data as Map.Entry (which contains both key-value pair).

```
static final int hash(Object key) {
        int h;
        return (key == null) ? 0 : (h = key.hashCode()) ^ (h >>> capacity);
}
```

--> From Java8, LinkedLists are dynamically replaced with "balanced binary search trees" in collision resolution after the number of collisions in a given bucket location exceed a certain threshold. This change offers a performance boost, since, in the case of a collision, storage and retrieval happen in O(log n).
-->

11. Hash Collision
--> https://www.baeldung.com/java-hashmap-advanced

11. Difference or correlation b/w hashKey() and equals() method.
--> equals() --> Determines if two objects are equal. The equals() method should check the equality of objects as precisely as possible.
--> hashKey() --> Returns an integer hash code value for an object, which is used to optimize performance when storing objects in hash-based data structures. The hashCode() method should return the same value each time it's called, unless the object property used in the equals() method is modified.
--> These methods are declared in Object class with its default implementation. Class which uses hashing based data structures should define its implementation.

12. HashMap vs LinkedHashMap implementation and internal working.
--> HashMap is array of LinkedList but LinkedHashMap is LinkedList of LinkedList.

13. map vs flatmap in Java streams
--> List<Integer> flatList
= number.stream()

```
                .flatMap(list -> list.stream())
                .collect(Collectors.toList());
```
14. Why Java Stream are called as Lazy loading?
    --> The Java 8 Streams API is lazy because it's based on a "process-only, on-demand" strategy, which means that intermediate operations are not evaluated until a terminal operation is invoked. This allows for more efficient processing of data by only computing the elements that are necessary to produce the final result.
14. Fetch duplicates in a list of Integers using Java Stream API
    --> Set<Integer> set = new HashSet<>();
    --> List<Integer> duplicates = list.stream().filter(l -> !set.add(l)).collect(Collectors.toList());

15. Fetch unique values in a list.
    --> list.stream().distinct().collect(Collectors.toList());

15. Intermediate and terminal operations in Stream API
16. What is thread safety
    --> To executed by a single thread at a time.
    --> Use synchronized keyword for thread safety.
17. Can we break singleton design pattern? If yes, how??
    --> Using serialization or cloning of the singleton object.
    --> We can also use Reflection.
18. Functional Interface and its types. Where are we using it??
    --> Interface having single abstract method and can have any number of static and default methods.
    e.g., Runnable, Callable, Comparator, ActionListener, etc.
19. BiPredicate function and its example.
    --? Function take two parameters and return result as boolean. e.g., Filtering Map object using Java stream.
20. Use of static and default methods in Functional interface.
    --> These have been introduced in Java8 which contains implementation of the method.
21. Checked and Unchecked exceptions.
22. Garbage collection and its algorithms.
    --> Two Types:
        --> Minor Garbage Collection
        --> Major Garbage collection
    --> Two methods System.gc() or Runtime.getRuntime().gc() can be used for Garbage collection.
    --> The call System.gc() is effectively equivalent to the call : Runtime.getRuntime().gc()
    --> Just before destroying an object, Garbage Collector calls finalize() method on the object to perform cleanup activities. Once finalize() method completes, Garbage Collector destroys that object.
    --> Algorithm Used: Mark and Sweep algorithm.
    --> Garbage Collector is an example of Daemon thread.

24. How to do garbage collection in Java.
    --> Use finalize() method of Object class.
25. Thread Pool in java and its types.
    --> SingleThreadPoolExecutor()
    --> FixedThreadPool(n)
    --> CachedThreadPool()

--> ScheduledThreadPool(n)
where n is initial number of threads to be created in pool.

26. How to execute multiple threads sequentially?
--> Use join() method.
--> Use SingleThreadPoolExecutor

27. Difference b/w java.lang.Thread.join() and java.lang.Thread.yield()
--> If a thread A calls join() method then current thread will be waiting till thread A is completing its task.
--> If any executing thread t1 calls join() on t2 i.e; t2.join() immediately t1 will enter into waiting state until t2 completes its execution.
--> Giving a timeout within join(), will make the join() effect to be nullified after the specific timeout.
--> If a thread A calls yield() method then thread A will wait until same or higher priority threads complete their task.

public static void yield() [Not throwing any checked exception, also not final]
public final void join() throws InterruptedException [join() method is not static so can be called using Thread object]
public final void join(long millis) throws InterruptedException
public final void join(long millis, int nanos) throws InterruptedException
public static void sleep(long millis) throws InterruptedException
public static void sleep(long millis, int nanos) throws InterruptedException

28. Thread Life Cycle.
--> New
--> Active
--> Blocked/Waiting
--> Timed Waiting
--> Terminated

29. Why wait(), notify() and notifyAll() methods are in Object class.
--> wait() - Tells the current thread to release the lock and go to sleep until some other thread enters the same monitor and calls notify().
The wait() method is used to make a thread voluntarily give up its lock on an object, allowing another thread to execute code within a synchronized block. The thread that calls wait() will enter a waiting state until another thread calls notify() or notifyAll() on the same object, allowing it to resume execution.
--> notify() - Wakes up the single thread that is waiting on this object's monitor.
The notify() method wakes up one of the waiting threads on the same object. If multiple threads are waiting, it is not specified which one will be awakened. The awakened thread will then compete for the lock on the object.
--> notifyAll() - It wakes up all the threads that called wait() on the same object.
The notifyAll() method wakes up all waiting threads on the same object. This can be useful when multiple threads are waiting, and you want all of them to be notified simultaneously.
--> wait() and notify() work at the monitor level and monitor is assigned to an object not to a particular thread. Hence, wait() and notify() methods are defined in Object class rather than Thread class.

--> wait(), notify() and notifyAll() methods can be called from synchronized blocks in Java.

https://medium.com/@reetesh043/java-wait-notify-and-notifyall-methods-3d3b511bd3ae
27. How to do serialization in Java and its interface?
28. How to block serialization in child class if parent class implements Serialization?
--> Override the writeObject(ObjectOutputStream stream) and readObject(ObjectInputStream stream) methods and throw NotSerializationException. Also make these methods private in child class.
29. Generics in Java. What wildcards are used in it?
--> Generics are parameterized types. It adds the type safety feature.
--> There are three types of wildcards in Java:
- Unbounded Wildcard (?) --> Used when you don't know or care about the specific type of the generic.
- Upper Bounded Wildcard (? extends T) --> Used when you want to restrict the type to a specific class or its subclasses.
- Lower Bounded Wildcard (? super T) --> Used when you want to restrict the type to a specific class or its supertypes.
30. Memory Management in String in Java.
--> Java uses String pools.
31. Priority Queue and Blocking Queue and its internal working.

Data Structures
---
1. Which sorting algorithm you have used?
2. What type of algorithm used in Merge Sort?
--> Divide and Conquer

Problems:
---
1. There is a list of Integer array. Find duplicates in it using Stream API
--> Set<Integer> set = new HashSet<>();
--> List<Integer> duplicates = list.stream().filter(l -> !set.add(l)).collect(Collectors.toList());

2. There is a String. We have to arrange it using its anagram. Do it either using Stream API or data structure. Choose Stream API at first.
e.g., dog fog god gof god fool gof loof --> dog god god fog gof gof fool loof
--> We can use LinkedHashMap directly
-->

Spring Boot
---
1. How to exclude a particular package or class dependency.
--> @ComponentScan(excludeFilters = @ComponentScan.Filter(value = {KafkaProducer.class})) // Similar for includeFilters
--> @ComponentScan(excludeFilters = @ComponentScan.Filter(value = {KafkaProducer.class}), basePackages = "com.*") // won't have include
2. Spring profiles
--> spring.profiles.active = dev

```
        application-dev.properties
--> To create profiles in same file
        #---
        spring.config.activate.on-profile=dev
        <List the dev properies>
        #---
        spring.config.activate.on-profile=test
        <List the test properies>
3. How to return two different format of response like XML format or JSON format
from same REST API?
-->     @RequestMapping(produces = {MediaType.APPLICATION_JSON_VALUE,
MediaType.APPLICATION_XML_VALUE})
        Also add Accept header while hitting API as {"Accept":
"application/json"}
4. Exception Handling in Spring Boot
--> For Global exception handling, use @ControllerAdvice and @ExceptionHandler
        @ControllerAdvice
        public class GlobalExceptionHandler {

                @ExceptionHandler(value = NoSuchCustomerExistsException.class)
                @ResponseStatus(HttpStatus.NOT_FOUND)
                public @ResponseBody ErrorResponse
handleException(NoSuchCustomerExistsException ex) {
                        return new ErrorResponse(HttpStatus.NOT_FOUND.value(),
ex.getMessage());
                }
        }

5. Lazy and Eager loading
6. @Primary, @Qualifier and @Required annotations
        --> @Qualifier --> Used for selecting bean in case of ambiguity
        --> @Primary --> Used for defining primary bean if multiple beans are
defined of same type. By default primary bean gets injected. if @Qualifier is
also present, it will be taken into consideration.
        --> @Required --> Used with Setter methods to inject dependency
7. Spring Boot internal working
        --> It uses spring.factories files present in META-INF folder
8. How to create custom exception class?
9. Caching in Spring boot. Num cache and Redis cache.
10. LRU and LFR in cache.
        --> LRU : Least Recently Used
        --> LFU : Least Frequently Used
11. Logging in spring boot.
        --> Using Slf4j and Log4j
        --> Logger log = LoggerFactory.getLogger(A.class);

12. How to create Git CI/CD pipeline?
        --> Using .gitlab-ci.yml file.
13. Any recent POC done in SpringBoot?
        --> Have implemented multitenancy in SpringBoot using Hibernate
AbstractMultitenancyConnectionProvider class which implements
MultitenantConnectionProvider interface.
14. How to implement Primary key and Composite key in JPA
15. Generation strategy in primary key.
```

	-->
16. Which Repository interface you have used?
	--> Both JPARepository<EntityName, PrimaryKeyDatatype> & CrudRepository<EntityName, PrimaryKeyDatatype>
17. @ConfigurationProperties in SpringBoot
	--> 1. Create a class to hold your configuration properties. Annotate it with @ConfigurationProperties and specify the prefix that will be used to match properties from your application.properties or application.yml file
	--> 2. In your application.properties or application.yml file, add the properties that you want to bind to your configuration class:
	--> 3. In your main application class, add the @EnableConfigurationProperties annotation to enable the binding of properties to your class.
	--> 4. Now you can inject your configuration properties class into any other Spring bean and use the values:
	https://www.baeldung.com/configuration-properties-in-spring-boot
17. Entity relationship in JPA
18. How JPA assures ACID principle?
	--> JPA used @Transactional annotation
	--> Atomicity: Ensures that either all operations in a transaction are completed or none are. If any part of the transaction fails, the entire transaction is rolled back.
	--> Consistency: Ensures that the database remains in a consistent state before and after a transaction.
	--> Isolation: Ensures that transactions are isolated from each other until they are complete.
	--> Durability: Ensures that changes made by a transaction are permanent and survive system failures.

19. How to handle transaction in SpringBoot?
--> @Transactional annotation is used
18. How to handle latency issue for some API in SpringBoot?
19. 401 and 403 error codes.
--> 200 : OK
--> 201 : Created
--> 202 : Accepted
--> 204 : No Content
--> 301 : Moved Permanently
--> 302 : Found
--> 400 : Bad Request
--> 401 : Unauthorized
--> 402 : Payment Required
--> 403 : Forbidden
--> 404 : Not Found
--> 405 : Method Not Allowed
--> 408 : Request Timeout
--> 409 : Conflict
--> 500 : Internal Server Error
--> 501 : Not Implemented
--> 502 : Bad Gateway
--> 503 : Service Unavailable
--> 504 : Gateway Timeout

20. What API headers you have used??

"Content-Type", "Accept", "Authorization"

21. Log Tracing mechanism
22. NFR (Non-Functional Requirements)
23. Any production support you have worked on
24. FetchType and its multiple values and the default value
--> Used in Entity Relationship mapping
        FetchType.LAZY --> Associative entity will not be loaded with main entity fields. A separate call would be required to fetch the data if needed.
        FetchType.EAGER --> Associative entity will be loaded alongwith main entity fields.

25. Transaction handling and its attribute
        --> @Transactional can only work when SpringBoot main class is annotated with @EnableTransactionManagement
26. Test Driven Design pattern
27. JUnit annotations
28. Difference b/w @Mock and @Spy
30. Use of @InjectMocks
31. How to test Rest APIs?
        --> JUnit for Unit testing. We can also use Postman
32. Any code quality tool used?
        --> Sonar Lint (SonarQube)
33. Reactive Programming
34. Spring Cloud Gateway
35. Hibernate First and Second Level Cache

SQL
---
1. Write a query to fetch 3rd max salary form Employee table.

2. We have Employee and Department table. Write a query to find the count of employees in a particular department.

Microservices
---
1. API Gateway
2. Service Registry
3. Circuit Breaker
4. Transaction handling in microservices.
5. Inter-service communication
6. Design patterns in Microservices
7. Saga and Circuit break design patterns
8. Log tracing using Grafana, Splunk or Cloudwatch
9. Fault tolerance
10. How to know if any service is malfunctioning. How do you fix it??
11. Time To Live (TTL)
12. Orchestration

13. GraphQL
14. gRPC

Spring Security
---
1. Which mechanism you have used?
2. Explain the flow
3. JWT and its parameters
4. How do you implement Spring Security. Which dependency is required?
	--> spring-boot-starter-security

Cloud
---
1. List the AWS services you have used?
2. Any serverless service like Lamdba
3. EKS, Storage service, RDS
4. Cloud watch

Miscellaneous
---
1. Have you participated in code review, Technical design
2. Have you created HLD or LLD designs
3. Types of diagrams you have created for designing --> Sequence diagram and Flow charts
4. Explain any end-to-end flow in your project.
5. NFR (Non Functional Requirements like Vulnerability fixes, Performance, etc.)
6. What tools have you used for Vulnerability analysis --> Aquasec (SCA), CAST Highlight (SAST)
7. How do you conduct performance testing? --> Using Apache JMeter
8. Have you done any production deployment?
9. How do you implement security in microservices?
10. How do you test the application performance?

HCLTech Interview Questions
---
1. If one service expects XML response from API and another service expects JSON response. How to implement that?
--> @RequestMapping(produces = {MediaType.APPLICATION_JSON_VALUE, MediaType.APPLICATION_XML_VALUE})
	Also add Accept header while hitting API as {"Accept": "application/json"}
2. Program: WAP using Stream API to sum of the digits of an Integer
	--> int num = 12345;
	--> int digitSum =

```
String.valueOf(num).chars().stream().mapToInt(Character::getNumericValue).sum();

3. What is API Gateway in microservices and how do you implement it?
4. How to implement security in microservices?
```

Accenture Interview Questions

```
1. What is actuator in Spring Boot?
2. How to reload beans on runtime?
        --> @RefreshScope and call /actuator/refresh API (POST)
3. How to execute beans in a specific order?
        --> @Order(<Integer value>)
4. @DependsOn annotation
        --> To make bean creation dependent on some other bean to be loaded into
the context.
5. @ConditionalOnMissingBean
        --> The @ConditionalOnBean and @ConditionalOnMissingBean annotations let
a bean be included based on the presence or absence of specific beans
6. How do you handle performance in your application?
        --> We're using Apache JMeter to run the PTs.
        --> Also to improve performance, we generally reduce database hits,
adding index in database columns,
        --> optimizing code by reducing multiple loops
        --> Choosing right data structure,
        --> Use caching
```

Indusind Interview Questions

```
1. Do we require same key for encryption and decryption?
        --> No, encryption and decryption should use different key as public
private key combination.
2. Difference b/w Encryption and Hashing.
3. Possible ways to create Thread in Java.
        --> Using Thread class and Runnable interface.
4. How to create Thread object using Runnable interface?
5. What would occur while executing the following code?
class ThreadEx extends Thread
{
    public void run()
    {
        System.out.print("Hello...");
    }
    public static void main(String args[])
    {
        ThreadEx T1 = new ThreadEx();
```

```
        T1.start();
        T1.stop();
        T1.start();
    }
}
        --> It will throw java.lang.IllegalThreadStateException exception.

6. What is Executor framework and CompletableFuture future. Write a program for
it.
7. How to execute threads sequentially?
-->
8. WAP using Java Stream API to find the second largest salary of the employee.
        -->
list.stream().map(Employee::getSalary).sorted(Comparator.reverseOrder()).skip(1)
.findFirst();
        -->
list.stream().sorted(Comparator.comparingInt(Employee::getSalary).reversed()).sk
ip(1).map(Employee::getSalary).findFirst().get();



------------------------------------------------------------------------
Wipro Interview Questions
------------------------------------------------------------------------
1. What is Oauth2?
2. JWT token and its parts.
3. How Spring Security is implemented?
4. Have you also done basic authentication with Spring Security?
5. What is Functional interface? List few functional interfaces.
6. What is lambda function?
7. Difference b/w Object Oriented and Functional programming.
8. What is Thread safety?
9. Is ConcurrentHashMap thread safe? --> Yes, it is thread safe because it uses
synchronization.
10. ConcurrentHashMap internal working
        --> As opposed to the HashTables where every read/write operation needs
to acquire the lock, there is no locking at the object level in CHM and locking
is much granular at a hashmap bucket level.
        --> CHM never locks the whole Map, instead, it divides the map into
segments and locking is done on these segments. CHM is separated into different
regions(default-16) and locks are applied to them. When setting data in a
particular segment, the lock for that segment is obtained. This means that two
updates can still simultaneously execute safely if they each affect separate
buckets, thus minimizing lock contention and so maximizing performance.
        --> No lock is applied on READ operations.
        -->
https://anmolsehgal.medium.com/concurrenthashmap-internal-working-in-java-b2a1a4
8c7289
10. What are the new changes done in HashMap implementation in Java8?
        --> From Java8, LinkedLists are dynamically replaced with "Balanced
Binary Search Trees" in collision resolution after the number of collisions in a
given bucket location exceed a certain threshold. This change offers a
performance boost, since, in the case of a collision, storage and retrieval
happen in O(log n).
```

11. How do you write Runnable interface before and after Java8 using Functional programming?

```
        # Using anonymous inner class
        --> class MyTask implements Runnable {
                        @Override
                        public void run() {
                                System.out.println("Executing task...");
                        }
                }

                // Usage
                Thread thread = new Thread(new MyTask());
                thread.start();


        --> Runnable task = new Runnable() {
                        @Override
                        public void run() {
                                System.out.println("This is a task running on a separate thread");
                        }
                };

                Thread thread = new Thread(task);
                thread.start();

        --> After Java8 using Lamda function.
        --> Runnable task = () -> System.out.println("Executing task with lambda...");
                Thread thread = new Thread(task);
                thread.start();
```

12. Difference b/w Callable vs Runnable interfaces.
13. Methods of Object class.
14. Difference b/w RestTemplate and WebClient.
15. Multiple ways for inter service communication.
        --> RestTemplate, WebClient, Messaging queue, etc.
16. What is API Gateway? How do you implement an API Gateway?
17. Design patterns in Microservices you have used.
18. What is Saga Design pattern?
        --> Choreography: Used with Message broker like Kafka.
        --> Orchestration: A centralized service which controlls all the services.
19. What is Circuit Breaker design pattern? What annotations we have to use to implement it?
20. What is CQRS Design pattern?
        --> Command Query Responsibilty Segregation.
21. You have Employee class having four fields Id, Name, Gender, Age. You have to find the count of Male and Female employees.

Wipro Interview Questions - Phase 2

---
1. Rate yourself with Java, SpringBoot and Microservices out of 5.
2. Java 8 features.
3. What is Java stream?
4. Difference b/w HashMap and Hash Table.
5. What is ConcurrentHashMap?
6. What is Singleton design pattern?
7. What do you mean by immutability? How do you implement it at various level?
6. Explain SpringBoot features and how it works internally?
7. What is Spring actuator?
8. What is POM file?
	--> Project Object Model
9. Spring profiles.
10. How do handle exception in SpringBoot?
10. What are microservices?
11. API Gateway, Service discovery.
12. How are two services interacting?
	--> Using RestTemplate
13. How do you handle transactions in microservices?
14. What is circuit breaker? How do you implement it?
15. Difference b/w final, finally and finalize.

Admiral Group Interview Questions Round-1
---
1. What are spring.factories? Explain Spring Boot autoconfiguration.
2. What is Spring JPA? How do we add it into Spring Boot project?
	--> I missed @EnableJpaRepositories annotation
3. What is Factory Interface?
4. How to create prototype bean in Spring Boot even if the controller is Singleton?
5. How to configure SSL in RestTemplate bean?
6. Kubernetes basic commands.
7. Basic cloud concepts, Availability zones, Region, VPC, Subnet.
8. How do you communicate to other service either through HTTP or HTTPS?
9. How do you handle transaction in SpringBoot?
10. How much deep testing you have done with JUnit?
11. Suppose there's a loop created in your service to add 5 employees in database which hits EmployeeRepository.save() method 5 times.
	How do you verify the count of calling EmployeeRepository.save() method if we Mock the repository object?
	--> Mockito.verify() [check more on Google]
12. Event driven approach.
13. Which VCS service you're using? Where do you store your JARS and Docker images? Azure Container Registry or something else.
14. Microservices Design pattern. What is orchestration and Choreography?
15. Do the following code:
	We have one root dictionary and other is statement. Wherever statement starts with root, replace the word with root. If multiple root matches, pick the one which is smallest in length.
	e.g., String[] dict = {"catt", "cat", "dog", "fat"};

String sentence = "fatttt dogggg hello doggtfsg fathjdb catthg fathd the";
Output = "fat dog hello dog fat cat fat the"

Admiral Group Interview Questions Round-2
---
1. Why do you want to join Admiral?
2. What are your strengths and weakness?
3.

Moglix Interview Questions Round-1
---
1. What is abstraction??
2. Constructor in Java. If abstract class is having constructor, why couldn't we create object from it?
	--> Abstract class constructor is being used by its child class for getting the object. The purpose of an abstract class is to act as a blueprint for other classes to inherit from.
3. Hash Collision and how HashMap handles it?
4. Difference b/w array and ArrayList.
5. Is ArrayList thread safe. What error it will throw if I make changes while iterating? How can I do use concurrency feature in it?
	--> Use CopyOnWriteArrayList
	--> To prevent unsynchronized access to the list, you can use the Collections.synchronizedList method when creating the list.
6. How do we handle Exception in Java?
7. If try, catch and finally are returning some result and error is throwing, which block will be executed and if there's no error what will be executed?
	--> If error is thrown, catch and finally both will be executed.
	--> If error is not thrown, only finally will be executed.
8. Use of final keyword in multiple context. Class, method and variable.
	--> final at class level stops inheritance.
	--> final at method level stops overriding.
	--> final at variable level makes it constant.
9. If we declare StringBuiilder variable as final, can we use append or delete operations?
	--> yes, we can do because variable will be pointing to the same object.
10. How to create our own Immutable and Singleton class?
11. Write a program to find the first non-repeating character in String without using Collections and Map.

```
--> StringBuilder sb = new StringBuilder();
	for(int i=0; i<s.length(); i++){
		int index = sb.indexOf(String.valueOf(s.charAt(i)));
		if(index != -1){
			sb.deleteCharAt(index);
		} else{
			sb.append(s.charAt(i));
```

```
                }
        }
        return sb.charAt(0);
```

12. Given a array of Integers having numbers from 1 to 100 in sorted order and all are unique. But one number is missing. Find that missing number.

```
--> for(int i=1; i<arr.size(); i++){
                if(arr.get(i-1) != i){
                        return i;
                }
        }
        return 100;
```

        Note: This can also be done using Binary search.

13. Difference b/w unique and primary key.
14. Indexing in SQL.
15. If we have an employee table having id, name, age, gender and salary. Write a query to get employee id and name having the second max salary.

```
--> SELECT emp.id, emp.name from employee emp order by salary desc offset 1
limit 1;
```

16. Write a query to fetch the list of employees having duplicate name and email combination.

```
--> SELECT emp.name, emp.email from employee emp group by emp.name, emp.email
having count(*) > 1;
```

17. Embedded Servers in SpringBoot.
18. What is auto configuration in SpringBoot?
19. @Qualifier annotation.
20. Default bean scope in SpringBoot.
21. Difference b/w Put and Post. Can we replace POST with PUT and vice versa.
22. Dependency Injection and Inversion of Control.
23. Any other way to get the object instead of @Autowired annotation?
24. If the bean is created, where its obect is stored. Either in classpath or context?
25. n+1 problem in Hibernate.
26. @Service annotation is used for.

Moglix Interview Questions Round-2

---

1. What is index in database? Its pros and cons.
2. Triggers in database.
3. Create two tables CUSTOMER and ORDER and write following queries.

```
        Order
        ----------------
        order_id --> PK
        purchase_date
        items
        cust_id --> FK
```

```
        Customer
        ----------------
        id --> PK
        name
        address
        contact
        email
        status -> 0,1

i. Write a query to fetch the list of customers who don't purchase any order on
17th Sept.
        -->
ii. Write a query to fetch the duplicate email_id in Customer table.
        --> select email_id from customer group by email having count(*) > 1;

iii. Write a query to deactivate duplicate records by updating the status field.
        --> UPDATE customer as c1 set c1.status = 0
                WHERE EXISTS(
                        SELECT 1 FROM customer AS c2 WHERE c1.email = c2.email
AND c1.id <> c2.id
                        GROUP BY c2.email HAVING count(*) > 1
                );

4. Write a program to reverse the String in the provided integer value chunks.
e.g.,   String s = "abcdefg"
                int chunk = 2
                Output --> fgdebca

                If chunk = 3, output --> efgbcda




-----------------------------------------------------------------------
Hughes Systique Interview Questions
-----------------------------------------------------------------------
1. How HashMap stores data? Explain its internal working.
2. What is hashkey() and equals() method?
3. How hashKey() and equals() method are getting calculated?
4. HashMap vs ConcurrentHashMap. Explain internal working of ConcurrentHashMap
and which one is thread safe.
5. What is deadlock? Create one example to represent deadlock situation.
6. Write a program to reverse the order of words in an statement.
        e.g., My name is Vishwas Maheshwari
        Output --> Maheshwari Vishwas is name My




-----------------------------------------------------------------------
Wissen Interview Questions
-----------------------------------------------------------------------
```

1. Design your product architecture.
2. Design an architecture for the food delivery app.
3. What is builder design pattern?
4. What is immutability? Make the following class immutable.
```
class Employee{
int id; String name; List<Department> departments;
}
```
5. How to add multiple database in SpringBoot?
Suppose we have five endpoints and 5 databases. Implement the SpringBoot project such that when request comes to endpoint 1, call goes to DB1, similar for others.
6. Write a program to get the output string by removing the characters in the original string with the following logic.
if A&B are adjacent or C&D are adjacent, remove the combination. Return the result when no such combination left after multiple iterations.
e.g., "AABCCDABDB" --> A[AB]C[CD][AB]DB --> "ACDB" --> A[CD]B --> "AB" --> ""
7. How spring security works?
8. Write a program using Java 8 to find the 3rd largest number.
9. Write a program using Java 8 to find the count of each character in String.

EPAM Interview Questions - Round 1

---

1. What is the use of hashCode() and equals() method?
2. How to implement class level and object level locks in Java?
3. Difference b/w Synchronization and concurrency.
2. How HashMap works internally?
3. List the collections you have worked on.
4. Difference b/w HashMap and ConcurrentHashMap.
5. SpringBoot actuator.
6. There is a list of integers. You have to find the numbers having 2nd digit is 1. You can't convert this to String and don't use any other collection.
7. There is a String of words. You have to get the fourth longest word in which there's a possibility of words with same length and in such case same length word will be treated at same level. Solve using stream API.
```
--> String s = "Hello every two nine seven five nineteen";
String[] splitted = s.split(" ");
Map<Integer, List<String>> map = Arrays.stream(splitted)
                            .collect(Collectors.groupingBy(String::length, Collectors.toList()));
                List<String> fourthLongest = map.get(map.entrySet().stream().map(Map.Entry::getKey)

.sorted(Comparator.reverseOrder()).skip(3).findFirst().get());
                System.out.println(fourthLongest);
```
8. What is API Gateway?
9. Circuit design pattern and how do you impalement it?
10. What is spring boot starter?
11. Scopes in java from lowest visibility to highest.
12. Default scope of bean in SpringBoot.
13. Why non-static methods can't be called from static method?

14. Any JVM level changes you have done?
	--> --xnx256 (OutOfMemoryError - to increase heap memory size, default is 256MB)
	--> --xss256 (StackOverflow - to increase stack overflow size, default is 256MB)

EPAM Interview Questions - Round 2
---
1. Can a function interface have methods of object class like hashCode(), equals() and toString()?
2. List the functional interfaces in Java before Java 8. Also what type of functional interfaces introduced in Java 8?
3. Java design patterns.
4. Give some example of Flyweight design pattern in Java.
	--> String pool
	--> Integer Cache (-128 to 127)
	--> Enum
5. What is Optional class? What its purpose? Can we inherit this class?
6. What happen if a public method is getting ovverriden by protected method?
	--> When overriding a method, the access modifier in the child class can be the same or more accessible than the parent class.
	--> This will give compile time error.
7. What are SOLID priciples?
8. What are DRY principles?
9. If we have common default methods in two interfaces A and B and class implementing both, what would happen?
	--> Compile time error will be thrown. To handle the same we have to implement the method in class and can use interface implementation by using A.super.grow() or B.super.grow();
10. If we have common static methods in two interfaces A and B and class implementing both, what would happen?
	--> It will process without any issue as static methods can be accessed using interface name.

11. If a Singleton scope class A have used a class B of prototype scope. If we @Autowire class A in some other class, what will happen?
12. How to execute some code after the bean creation and before the bean deletion?
13. By default JPA executed queries are accessible in console. How can you disable this?
	--> spring.jpa.show-sql=false
14. Scenarios when finally block won't get executed?
15. Have you accessed any conditional annotations?
16. Any NoSQL databases you have used like MongoDB?
17. By default spring boot uses tomcat server as its embedded server. If we want to use some other server like Jetty as embedded server, how to do that?
18. If a request is passed through multiple services. At some point of time, request got failed, how would you know that at which service this request got

failed?
19. If a service is down or going through some failure. Another service is continuously hitting it, how would you prevent the hits to the service if it is down?
	--> By implementing Circuit breaker.
20. Are you aware of TDD (Test Driven Design)? How do you implement it?
21. How would you do performance testing and integration testing?
22. Are you aware of ACID properies of database?
23. Which cloud services you have used?
24. If EC2 gets down, how do we make sure its data would be preserved?
	--> Use EBS (Elastic Block Service)
25. What's the max file size we can upload on Amazon S3?
26. if we want to run the entire project on AWS Lamdba. How much max size we can upload on it?
	-->
27. How to create CI/CD pipeline on Jenkins?

UST Global Interview Questions
---
1. Adapter Design pattern
2. Bridge design pattern
3. Diff b/w == and equals.
4. CompletableFuture and ExecutorService
5. Program: You have a list of numbers. Find the numbers which starts with 1. Use stream API.
6. How do you configure multiple databases in SpringBoot?
7. Why String is immutable in Java?
8. Difference b/w StringBuilder and StringBuffer.
9. Difference b/w Collections and streams.

Cognizant Interview Questions
---
1. What is the default capacity of ConcurrentHashMap?
2. List the design patterns you've used.
3. What is Oauth2. How is it working?
4. What is JWT?
5. Any microservices design pattern you've used?
6. Write a program to get the frequency of numbers in the list. Use stream API.
7. Stored procedure.
8. What is indexing?
9. What is CompletableFuture?
10. What is Executor framework and difference b/w ExecutorService and CompletableFuture.
11. How @Transactional works internally in SpringBoot?

TSYS Interview Questions
---
1. Thread Life Cycle.
2. How two threads communicate with each other?
3. What is Serialization?
4. What is the use of serialVersionUID variable?
	--> SerialVersionUID is used to ensure that during deserialization the same class (that was used during serialize process) is loaded.
	--> It is created by JVM for the class to validate while deserialization for confirming the class whether it is same or not. if its different, JVM will throw InvalidClassException.
	--> https://www.geeksforgeeks.org/serialversionuid-in-java/
5. What is immutability? How to achieve it both at object and class level?
6. What do you mean by unreachable catch block?
7. Can we write try without catch?
8. Is there any scenario where catch block won't get executed?
9. What design patterns you have used?
10. What do you mean by Lambda function?
11. How to create custom Functional interface? @FunctionalInterface is required for this?
12. Which version of Java you have worked on? Have you used Java 11 or 17 as well?
13. What is Optional class?
13. Write a program to write the numbers in an alternate fashion e.g., first positive then negative till all the possible combinations. Then add` the rest of them as it is.

```
	e.g.,	Input - [-1, 2, -3, 4, 5, 6, -7, 8, 9]
			Output - [9, -7, 8, -3, 6, -1, 5, 2, 0]
	-->	Arrays.sort(arr);
		int i= arr.length-1;
		int j = 0;
		int[] res = new int[arr.length];
		int k = 0;
		while(j < i) {
			res[k++] = arr[i--];
			res[k++] = arr[j++];
		}
		return res;
```

14. Write a program to convert the following String by adding frequency to each letter.

```
	e.g.,	Input string - abbbccddaaabbccceeff
			Output String - a1b3c2d2a3b2c3e2f2
	-->	String s = "abbbccddaaabbccceeff";
		StringBuilder sb = new StringBuilder();
		char[] ch = s.toCharArray();
		sb.append(ch[0]);
		int j = 0;
		for (int i = 1; i < ch.length; i++) {
			if (ch[i] != ch[i - 1]) {
				sb.append(String.valueOf(i - j));
				j = i;
```

```
                                sb.append(ch[i]);
                        }
                }
                sb.append(ch.length - j);
                return sb.toString();
```

15. Write a program to reverse the process in the last question. Also consider the frequency in double or triple digits.

e.g., Input string - a12b13c2d2a3b2c3e2f1
Output String - aaaaaaaaaaaabbbbbbbbbbbbbccddaaabbccceef

Tech Mahindra Interview Questions

---

1. Difference b/w Vector and HashMap.
2. ConcurrentHashMap.
3. Java 8 features.
4. Write a program to sort the integer array.
5. Actuator in spring boot.
6. Bean scopes in SpringBoot.
   --> singleton: (Default) Scopes a single bean definition to a single object instance for each Spring IoC container.
   --> prototype: Scopes a single bean definition to any number of object instances.
   --> request: Scopes a single bean definition to the lifecycle of a single HTTP request. That is, each HTTP request has its own instance of a bean created off the back of a single bean definition. Only valid in the context of a web-aware Spring ApplicationContext.
   --> session: Scopes a single bean definition to the lifecycle of an HTTP Session. Only valid in the context of a web-aware Spring ApplicationContext.
   --> application: Scopes a single bean definition to the lifecycle of a ServletContext. Only valid in the context of a web-aware Spring ApplicationContext.
   --> websocket: Scopes a single bean definition to the lifecycle of a WebSocket. Only valid in the context of a web-aware Spring ApplicationContext.
6. How do you configure datasource in spring boot?
7. How do you capture database password? Which type of vault you use?
8. What is docker? Why do we use it for Microservices?
9. How docker works?
10. How much you are aware of cloud services?
11. Which cloud platform do you use?
12. Difference b/w DROP, DELETE and TRUNCATE.
   --> DROP removes an entire table or database, including its structure.
   --> DELETE removes specific rows from a table while keeping its structure.
   --> TRUNCATE removes all rows from a table while keeping its structure.

13. What's the ideal percentage of Unit test coverage?
   --> 80%

Sigmoid Interview Questions - Round 1
---
1. Difference b/w Concurrency and synchronization.
2. Java 8 features.
3. Difference b/w Vector and ArrayList.
4. DB Query:
There is a customers and orders table. You have to write a query which fetch the list of customers with their total order value for the customers who have placed minimum of three orders in last year and have total order value greater than 5000$.

```
--> SELECT
      c.customer_id,
      c.customer_name,
      SUM(o.order_value) AS total_order_value
    FROM
      customers c
    INNER JOIN
      orders o ON c.customer_id = o.customer_id
    WHERE
      YEAR(o.order_date) >=
    GROUP BY
      c.customer_id
    HAVING
      COUNT(o.order_id) >= 3 AND
      SUM(o.order_value) > 5000;
```

5. There is an array representing length of ropes. We have to join all the ropes such that their cost of joining would be minimum. Cost of joining two ropes is the sum of their length. Write a Java program for the same.

```
--> PriorityQueue<Integer> minHeap = new PriorityQueue<>();
for (int rope : ropes) {
    minHeap.add(rope);
}

int totalCost = 0;
while (minHeap.size() > 1) {
    int rope1 = minHeap.poll();
    int rope2 = minHeap.poll();
    totalCost += rope1 + rope2;
    minHeap.add(rope1 + rope2);
}

return totalCost;
```

Sigmoid Interview Questions - Round 2
---
1. Difference b/w SQL and NoSQL databases.
2. If a e-commerce company having two types of records - Users and Products.

Where type of database you would prefer for them. Either same type of different.
	--> Users can be save in a structure way, so we can use RDBMS database.
	--> Products can be of any type like grocery, electronic, clothing, etc. So its an unstructured type data which should be stored in unstructured database like NoSQL.
3. What is Kafka? How it is different from other messaging queues?
4. What is a messaging queue? Where do we use it? What type of problem it is solving?
5. Write a query to find 3 minimum salaries of Employee.
	--> select salary form employee order by salary limit 3;
6. Write a program to solve the following problem.
	There is an array. You have to replace each of its value by its immediate greater number.
	How do we handle if this could be a circular array or list.

7. Internal working of Spring Boot.
8. How do request flows in Spring Boot controller. Can we make changes in the request before executing the controller? Also same for Post request
	--> Using Interceptor
		--> preHandle():
		--> afterCompletion():
		--> postHandle():
	--> In Spring Boot, firstly request comes to dispatcher, then it will redirect the request based on its path to specific controller and then specific API.

Global Logic Interview Questions
---
1. Working difference b/w PUT and PATCH
2. Can you write complete CRUD operations?
3. How @Transactional works in SpringBoot?
4. What are starter dependencies in Spring Boot?
5. How do we take our data from on-premises to S3? Explain the approach you will use for this.
6. Can we create thread pools on our own. If yes, what's the need of Executor framework?
	--> Executor Framework provides a predefined and optimized way of handling thread pools. We can also create our thread pools but we have to manage it on our own which requires extra effort. Also Executor framework uses Callable Interface which can return some result but thread is using Runnable interface which won't return anything.
7. Where do you use HashSet? Give some real life scenario.
8. When we get 500 error?
9. What is sharding?

Amerprise Interview Questions
---
1. Difference b/w Inheritance and Abstraction.

```
-----------------------------------------------------------------------
Others
-----------------------------------------------------------------------
1. Actuator in SpringBoot
2. Interceptor in SpringBoot
        --> Methods:
                --> preHandle():
                --> afterCompletion():
                --> postHandle():
                @Configuration
                public class RequestInterceptorConfig implements
WebMvcConfigurer {

                          // Register an interceptor with the registry,
Interceptor name : RequestInterceptor
                          @Override
                        public void addInterceptors(InterceptorRegistry
registry) {
                                registry.addInterceptor(new
RequestInterceptor());
                        }
                        //* We can register any number of interceptors with our
spring application context
                }
3. Microservices Architecture Playlist
        -->
https://www.youtube.com/playlist?list=PLSVW22jAG8pBnhAdq9S8BpLnZ0_jVBj0c
4. How to create your own spring boot starter?
5. Pub-Sub vs Messaing queue.
        --> Message queues consist of a publishing service and multiple consumer
services that communicate via a queue. This communication is typically one way
where the publisher will issue commands to the consumers. The publishing service
will typically put a message on a queue or exchange and a single consumer
service will consume this message and perform an action based on this.
        --> Conversely, to message queues, in a pub-sub architecture we want all
our consuming (subscribing) applications to get at least 1 copy of the message
that our publisher posts to an exchange.
        --> https://www.baeldung.com/pub-sub-vs-message-queues
6. Difference b/w Arrays.asList() and List.of()
        -->     Arrays.asList()                                 |       List.of()
                ------------------------------------
                        |   ----------------------------------------
                1. Introduced in Java 8.
                                        |       1. Introduced in Java 9.
                1. Can contains null values
                                        |       1. Can't contain Null values
                2. Elements can be modified but can't be added or removed.
        |       2. Immutable(no modifications allowed)
                3. Fixed Size
```

                                                | 3. Fixed size
                4. Backed by an array. Any changes made to the array or list
| 4. Not backed by any array.
                   affect the both

7. Circular Dependency In SpringBoot
        --> Exception occurred BeanCurrentlyInCreationException
        --> Use @Lazy with any of the bean while using Constructor injection. It will create a proxy bean to be loaded and later load the actual bean on its first call.
        --> Use Setter injection because circular dependency is occurring while using Constructor injection.
        --> Use @PostConstruct to initialize the dependent bean post the bean initialization.
        https://www.baeldung.com/circular-dependencies-in-spring
        --> We can also use the property to resolve the circular dependency.
                spring.main.allow-circular-references=true

8. Comparator vs Comparable
        Comparable --> new T1().compareTo(new T2());
        Comparator --> Comparator.compare(Object o1, Object o2);
9. Why is Enum required in Java?
        --> Type Safety: Enums provide type safety, ensuring that only valid values are assigned to a variable. This prevents accidental errors caused by using incorrect string or integer values.
        -->   Readability: Enums make code more readable and self-explanatory. Instead of using magic numbers or strings, you use meaningful names that represent the values.
        -->   Maintainability: Enums make code easier to maintain. If you need to add or modify a constant, you only need to change it in one place (the enum declaration) instead of searching and updating multiple occurrences throughout the code.
        --> Enhanced Functionality: Enums are more than just constants. They can have methods, constructors, and even fields. This allows you to associate additional behavior or data with each constant.
        --> Iteration: You can easily iterate over the values of an enum using a loop or the values() method.