OPPs Concepts:-

1.) The difference between method overloading and overriding?
Answer: Several differences but the most important one is that method overloading is resolved at compile time and method overriding is resolved at runtime. The compiler only used the class information for method overloading, but it needs to know object to resolved overridden method calls.

2.) Can we override a private method in Java?
Answer: No, you cannot. Since the private method is only accessible and visible inside the class they are declared, it's not possible to override them in subclasses. Though, you can override them inside the inner class as they are accessible there.

3.) Can we override a method which throws runtime exception without throws clause?
Answer: Yes, there is no restriction on unchecked exception while overriding. On the other hand, in the case of checked exception, an overriding exception cannot throw a checked exception which comes higher in type hierarchy e.g. if original method is throwing IOException than overriding method cannot throw java.lang.Exception or java.lang.Throwable.

4.) Can we have a non-abstract method inside interface?
Answer: From Java 8 onward you can have a non-abstract method inside interface, prior to that it was not allowed as all method was implicitly public abstract. From JDK 8, you can add static and default method inside an interface.

5.) What is difference between Abstraction and Encapsulation in Java?
Answer: Encapsulation is wrapping, just hiding properties and methods. Encapsulation is used for hide the code and data in a single unit to protect the data from the outside the world. Class is the best example of encapsulation. Abstraction refers to showing only the necessary details to the intended user.

6.) What is the 5 objects oriented design principle from SOLID?
Answer: In SOLID each character stands for one design principle:

- **S for Single Responsibility Principle**- a class should have only a single responsibility (i.e. changes to only one part of the software's specification should be able to affect the specification of the class).
- **O for Open closed design principle-** Objects or entities should be open for extension, but closed for modification.
- **L for Liskov substitution principle-** "objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program
- **I for Interface segregation principle**- "many client-specific interfaces are better than one general-purpose interface. (A client should never be forced to implement an interface that it doesn't use or clients shouldn't be forced to depend on methods they do not use.)
- **D for Dependency inversion principle**- one should "depend upon abstractions, [not] concretions

Serialization and Deserialization

1.) What will happen if we try to serialize List, Set and Map as member of class in java?
a. compilation error
b. They will get serialized
c. Throw Exception

Answer: B- They will get serialized

2.) What will happen if we try to serialize primitive types in java?
a. They will get serialized
b. compilation error
c.) Throw Exception

Answer: A-They will get serialized

3.) Suppose Parent Class is not serializable but Child Class is Serializable. Will there be any problem If we try to serialize the object of Child class
a.) Won't Allow Serialization
b.) No Problem
c.) Throw Exception

Answer: No problem

Multithreading

1.) How can we create a Thread in Java?
Answer: There are two ways to create Thread in Java
1.) First by implementing Runnable interface and then creating a Thread object from it
2.) Extend the Thread Class. Read this post to learn more about creating threads in java.

2.) What are different states in lifecycle of Thread?
Answer: When we create a Thread in java program, its state is New, then we start the thread that change its state to Runnable. Thread Scheduler is responsible to allocate CPU to threads in Runnable thread pool and change their state to Running. Other Thread states are Waiting, Blocked and Dead. Read this post to learn more about life cycle of thread.

3.) How does thread communicate with each other?
Answer: Object class wait(), notify() and notifyAll() methods allows threads to communicate about the lock status of a resource.

4.) Why thread communication methods wait(), notify() and notifyAll() are in Object class?

Answer: In Java every Object has a monitor and wait, notify methods are used to wait for the Object monitor or to notify other threads that Object monitor is free now. There is no monitor on threads in java and synchronization can be used with any Object, that's why it's part of Object class so that every class in java has these essential methods for inter thread communication

5.) How can we achieve thread safety in Java?

Answer: There are several ways to achieve thread safety in java –

- Synchronization,
- Atomic concurrent classes,
- Implementing concurrent Lock interface,
- Using volatile keyword,
- Using immutable classes and Thread safe classes etc.

6.) What is volatile keyword in Java

Answer: The value of this variable will never be cached thread-locally: all reads and writes will go straight to "main memory"; Access to the variable acts as though it is enclosed in a synchronized block, synchronized on itself.

7.) What would be output if call a wait on an object Without taking a lock

Answer: it will through Runtime Exception.

8.) Is it possible to start a thread twice?

Answer: No, after having started a thread by invoking its start() method, a second invocation of start() will throw an IllegalThreadStateException.

9.) What is Deadlock? How to analyze and avoid deadlock situation?

Deadlock is a programming situation where two or more threads are blocked forever, this situation arises with at least two threads and two or more resources.

- **To analyze a deadlock**, we need to look at the java thread dump of the application, we need to look out for the threads with state as BLOCKED and then the resources it's waiting to lock, every resource has a unique ID using which we can find which thread is already holding the lock on the object.
- **Avoid deadlock situation**
  Avoid Nested Locks, Lock Only What is Required and Avoid waiting indefinitely are common ways to avoid deadlock situation, read this post to learn how to analyze deadlock in java with sample program.

10.) What is Thread Pool? How can we create Thread Pool in Java?

A thread pool manages the pool of worker threads, it contains a queue that keeps tasks waiting to get executed.

A thread pool manages the collection of Runnable threads and worker threads execute Runnable from the queue.

Creating Thread Pool-Its basically the ExecutorService (it's an interface) which is used to create Threadpools. There are implementations of ExecutorService which are used to create ThreadPools like:-

- newFixedThreadpools,
- newCacheThreadPools,
- Executors.newSingleThreadExecutor()

If you want to schedule a task to run with delay or periodically then you can use ScheduledThreadPoolExecutor class.

11.)What is Lock interface in Java Concurrency API? What are its benefits over synchronization?
Lock interface provide more extensive locking operations than can be obtained using synchronized methods and statements. They allow more flexible structuring, may have quite different properties, and may support multiple associated Condition objects.
The advantages of a lock are
it's possible to make a thread responsive to interruption while waiting on a Lock object.
it's possible to try to acquire the lock, but return immediately or after a timeout if the lock can't be acquired
it's possible to acquire and release locks in different scopes, and in different orders

12.) What is Executors Framework?

- In Java 5, Executor framework was introduced with the java.util.concurrent.Executor interface.
- The Executor framework is a framework for standardizing invocation, scheduling, execution, and control of asynchronous tasks according to a set of execution policies.
- Creating a lot many threads with no bounds to the maximum threshold can cause application to run out of heap memory. So, creating a ThreadPool is a better solution as a finite number of threads can be pooled and reused. Executors framework facilitate process of creating Thread pools in java.

13.) What is the difference between the Runnable and Callable interfaces? How are they used?
Answer: The Runnable interface has a single run method. It represents a unit of computation that has to be run in a separate thread. The Runnable interface does not allow this method to return value or to throw unchecked exceptions.
The Callable interface has a single call method and represents a task that has a value. That's why the call method returns a value. It can also throw exceptions. Callable is generally used in ExecutorService instances to start an asynchronous task and then call the returned Future instance to get its value.

14.) What are some of the improvements in Concurrency API in Java 8?
Answer: Some important concurrent API enhancements are:

- ConcurrentHashMap compute(), forEach(), forEachEntry(), forEachKey(), forEachValue(), merge(), reduce() and search() methods.
- CompletableFuture that may be explicitly completed (setting its value and status).
- Executors newWorkStealingPool() method to create a work-stealing thread pool using all available processors as its target parallelism level.

Design Patterns (Covering 2-3 basic Patterns)

1.) Check What all design the patterns that the candidate may have implemented in his projects

Answer: There are 3 categories of design patterns:-

- creational,
- structural,
- behavioral

Creational Patterns	Structural Patterns	Behavioral patterns
• Singleton Pattern • Factory Pattern • Abstract Factory Pattern • Builder Pattern • Prototype Pattern	• Adapter Pattern • Composite Pattern • Proxy Pattern • Flyweight Pattern • Facade Pattern • Bridge Pattern • Decorator Pattern	• Template Method Pattern • Mediator Pattern • Chain of Responsibility Pattern • Observer Pattern • Strategy Pattern • Command Pattern • State Pattern • Visitor Pattern • Interpreter Pattern • Iterator Pattern • Memento Pattern

2.) What Is Singleton Pattern?

Answer

- This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.
- This pattern involves a single class which is responsible to create an object while making sure that only single object gets created.
- This class provides a way to access its only object which can be accessed directly without need to instantiate the object of the class.

3.) How Can You Create Singleton Class In Java? –

Answer: It is two-step process:

- First, make the constructor private so that new operator cannot be used to instantiate the class.
- Return an object of the object if not null otherwise create the object and return the same via a method.

How to create thread safe singleton Class

- The easier way to create a thread-safe singleton class is to make the global access method synchronized, so that only one thread can execute this method at a time, but it reduces the performance because of cost associated with the synchronized method, although we need it only for the first few threads who might create the separate instances.

- To avoid this extra overhead every time, double checked locking principle is used. In this approach, the synchronized block is used inside the if condition with an additional check to ensure that only one instance of singleton class is created.

4.) What Is Factory Pattern?

Answer:

- Factory pattern is one of most used design pattern in Java.
- This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.
- In Factory pattern, we create object without exposing the creation logic to the client and refer to newly created object using a common interface.

5.) What Is Abstract Factory Pattern?

Answer:

- Abstract Factory patterns work around a super-factory which creates other factories.
- This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.
- In Abstract Factory pattern an interface is responsible for creating a factory of related objects without explicitly specifying their classes.
- Each generated factory can give the objects as per the Factory pattern.

Collections

1.) What is difference between HashMap and Hashtable?

Answer: HashMap and Hashtable both implements Map interface and looks similar, however there are following difference between HashMap and Hashtable.

HashMap	HashTable
• HashMap allows null key and values • Hashtable is synchronized, So HashMap is better for single threaded environment. • HashMap provides Set of keys to iterate and hence it's fail-fast • Hashtable is considered to be legacy class	• HashTable doesn't allow null key and value. • HashMap is not synchronized, So Hashtable is suitable for multi-threaded environment. • Hashtable provides Enumeration of keys that doesn't support this feature. • if you are looking for modifications of Map while iterating, you should use ConcurrentHashMap.

2.) What is difference between Array and ArrayList? When will you use Array over ArrayList?

- Arrays can contain primitive or Objects whereas ArrayList can contain only Objects.
- Arrays are fixed size whereas ArrayList size is dynamic.
- Arrays doesn't provide a lot of features like ArrayList, such as addAll, removeAll, iterator etc.

Although ArrayList is the obvious choice when we work on list, there are few times when array are good to use.

- If the size of list is fixed and mostly used to store and traverse them.
- For list of primitive data types, although Collections use autoboxing to reduce the coding effort but still it makes them slow when working on fixed size primitive data types.
- If you are working on fixed multi-dimensional situation, using [][] is far more easier than List<List<>>

3.) What is difference between ArrayList and LinkedList?

ArrayList and LinkedList both implement List interface but there are some differences between them.

ArrayList is an index-based data structure backed by Array, so it provides random access to its elements with performance as O(1) but LinkedList stores data as list of nodes where every node is linked to its previous and next node. So even though there is a method to get the element using index, internally it traverse from start to reach at the index node and then return the element, so performance is O(n) that is slower than ArrayList.

Insertion, addition or removal of an element is faster in LinkedList compared to ArrayList because there is no concept of resizing array or updating index when element is added in middle.
LinkedList consumes more memory than ArrayList because every node in LinkedList stores reference of previous and next elements.

4.) When do you use ConcurrentHashMap in Java?

Answer:

- You should use ConcurrentHashMap when you need very high concurrency in your project.
- It is thread safe without synchronizing the whole map.
- Reads can happen very fast while write is done with a lock.
- There is no locking at the object level.
- The locking is at a hashmap bucket/segment level.
- ConcurrentHashMap doesn't throw a ConcurrentModificationExceptionif one thread tries to modify it while another is iterating over it.
- If you have an equal number of reader and writer than ConcurrentHashMap will perform in the line of Hashtable or synchronized HashMap

5.) What will happen if 2 threads simultaneously tries to write data to the same bucket in the ConcurrentHashMap?

Answer: The 2nd thread will be blocked till 1st thread completes

6.) If I will try to add a custom object as a key which is already present in ConcurrentHashMap, what will happen?

Answer: It will override the old value

7.) Which of these collections provides the slowest search time?

1. CopyOnWriteArrayList
2. TreeMap
3. Hashmap

Answer: CopyOnWriteArrayList

8.) Which of these collections will allow removal in FIFO order?

- Treemap
- LinkedHashmap
- Concurrenthashmap

Answer: LinkedHashmap

Which of the following practices produces a good hashcode() implementation.

1. Use random function in hashcode formula.
2. Use multiples of hash size in hashcode formula.
3. Use prime numbers in hashcode formula.

Answer: Use prime numbers in hashcode formula.

Data Structure & Algorithms:

1.) What is the complexity of adding and deleting an element in TreeMap?

Answer: O(Log N)

2.) Why Red-black trees are preferred over hash tables though hash tables have constant time complexity?

a) no, they are not preferred
b) because of resizing issues of hash table and better ordering in redblack trees
c) because they can be implemented using trees
d) because they are balanced

Answer: B
Explanation: Redblack trees have O(log n) for ordering elements in terms of finding first and next elements. also, whenever table size increases or decreases in hash table you need to perform rehashing which can be very expensive in real time. also, red black stores elements in sorted order rather than input order.

3.) What is the complexity of adding an element to the heap.

a) O(log n)

b) O(h)
c) a and b
d) None of the mentioned

Answer: C
Explanation: The total possible operation in re locating the new location to a new element will be equal to height of the heap.

4.) Heap exhibits the property of a binary tree?
a) True
b) False

Answer: A
Explanation: Yes, Because the leaf nodes are present at height h or h-1, which is a property of complete binary tree.

Java Memory management & Garbage Collection

1.) Where an object of a class get stored?
a) Heap
b) Stack
c) Disk
d) File

Answer: A

2.) Which exception is thrown when java is out of memory?
a) MemoryFullException
b) MemoryOutOfBoundsException
c) OutOfMemoryError
d) MemoryError

Answer: C
Explanation: The Xms flag has no default value, and Xmx typically has a default value of 256MB. A common use for these flags is when you encounter a java.lang.OutOfMemoryError.

3.) Which class loader loads jar files from JDK directory?
a) Bootstrap
b) Extension
c) System
d) Heap

Answer: B
Explanation: Extension loads jar files from lib/ext directory of the JRE.This gives the basic functionality available.

4.) Classes and Methods are stored in which space?
a) Eden space

b) Survivor space
c) Tenured space
d) Permanent space

Answer: D
Explanation: The permanent generation holds objects which JVM finds convenient to have the garbage collector. Objects describing classes and methods, as well as the classes and methods themselves are a part of Permanent generation.

5.) Where is String Pool stored?

a) Java Stack
b) Java Heap
c) Permanent Generation
d) Metaspace

Answer: (C) Permanent Generation (this is part of Java Heap)
Explanation: When a string is created ; if the string already exists in the pool, the reference of the existing string will be returned, else a new object is created, and its reference is returned.

Spring Framework

1.) Ask to candidate That- What all Spring Framework modules the Candidate has used in project
Answer: Some of the important Spring Framework modules are:

- Spring Context – for dependency injection.
- Spring AOP – for aspect oriented programming.
- Spring DAO – for database operations using DAO pattern
- Spring JDBC – for JDBC and DataSource support.
- Spring ORM – for ORM tools support such as Hibernate
- Spring Web Module – for creating web applications.
- Spring MVC – Model-View-Controller implementation for creating web applications, web services etc.

2.) What are some of the important Spring annotations you have used?
Answer: Some of the Spring annotations are:

Spring MVC:-

- **@Controller** – for controller classes in Spring MVC project.
- **@RequestMapping** – for configuring URI mapping in controller handler methods. This is a very important annotation, so you should go through Spring MVC RequestMapping Annotation Examples
- **@ResponseBody** – for sending Object as response, usually for sending XML or JSON data as response.
- **@PathVariable** – for mapping dynamic values from the URI to handler method arguments.
- **@Service** – for service classes.

- **@Configuration, @ComponentScan**

Spring CORE

- **@Bean** – for java-based configurations.
- **@Autowired** – for autowiring dependencies in spring beans.
- **@Qualifier** – with @Autowired annotation to avoid confusion when multiple instances of bean type are present.
- **@Scope** – for configuring scope of the spring bean.
- AspectJ annotations for configuring aspects and advices,

Spring AOP:- @Aspect, @Before, @After, @Around, @Pointcut etc.

3.) Can we send an Object as the response of Controller handler method?

Answer: Yes we can, using @ResponseBody annotation. This is how we send JSON or XML based response in Restful web services.

4.) Does Spring Bean provide thread safety?

Answer: The default scope of Spring bean is singleton, so there will be only one instance per context. That means that all the having a class level variable that any thread can update will lead to inconsistent data. Hence in default mode spring beans are not thread-safe.
However we can change spring bean scope to request, prototype or session to achieve thread-safety at the cost of performance. It's a design decision and based on the project requirements.

What are different types of Spring Bean autowiring?

Answer:

There are four types of autowiring in Spring framework.

- autowire byName
- autowire byType
- autowire by constructor
- autowiring by @Autowired and @Qualifier annotations

Hibernate

1.) Name some important annotations used for Hibernate mapping?

Hibernate supports JPA annotations and it has some other annotations in org.hibernate.annotations package. Some of the important JPA and hibernate annotations used are:

- **javax.persistence.Entity:** Used with model classes to specify that they are entity beans.
- **javax.persistence.Table:** Used with entity beans to define the corresponding table name in database.
- **javax.persistence.Access:** Used to define the access type, either field or property. Default value is field and if you want hibernate to use getter/setter methods then you need to set it to property.

- **javax.persistence.Id:** Used to define the primary key in the entity bean.
- **javax.persistence.EmbeddedId:** Used to define composite primary key in the entity bean.
- **javax.persistence.Column:** Used to define the column name in database table.
- **javax.persistence.GeneratedValue:** Used to define the strategy to be used for generation of primary key. Used in conjunction with javax.persistence.GenerationType enum.
- **javax.persistence.OneToOne:** Used to define the one-to-one mapping between two entity beans. We have other similar annotations as OneToMany, ManyToOne and ManyToMany
- **org.hibernate.annotations.Cascade:** Used to define the cascading between two entity beans, used with mappings. It works in conjunction with org.hibernate.annotations.CascadeType
- **javax.persistence.PrimaryKeyJoinColumn:** Used to define the property for foreign key. Used with org.hibernate.annotations.GenericGenerator and org.hibernate.annotations.Parameter

2.) Hibernate SessionFactory is thread safe?
Answer: Internal state of SessionFactory is immutable, so it's thread safe. Multiple threads can access it simultaneously to get Session instances.

3.) What is Hibernate Session and how to get it?
Answer: Hibernate Session is the interface between java application layer and hibernate. This is the core interface used to perform database operations. Lifecycle of a session is bound by the beginning and end of a transaction.
Session provide methods to perform create, read, update and delete operations for a persistent object. We can execute HQL queries, SQL native queries and create criteria using Session object.

4.) Hibernate Session is thread safe?
Answer: Hibernate Session object is not thread safe, every thread should get its own session instance and close it after it's work is finished.

Webservices

1.) What are different HTTP Methods supported in Restful Web Services?
Answer: Restful web services supported HTTP methods are – GET, POST, PUT, DELETE and HEAD.

2.) Comparison in between SOAP & REST
Answer:

SOAP webservices	Rest Webserives
• SOAP is a standard protocol for creating web services.	• REST is an architectural style to create web services.
• SOAP is acronym for Simple Object Access Protocol.	• REST is acronym for REpresentational State Transfer.

• SOAP uses WSDL to expose supported methods and technical details.	• REST exposes methods through URIs, there are no technical details.
• SOAP web services and client programs are bind with WSDL contract	• REST doesn't have any contract defined between server and client
• SOAP web services and client are tightly coupled with contract.	• REST web services are loosely coupled.
• SOAP learning curve is hard, requires us to learn about WSDL generation, client stubs creation etc.	• REST learning curve is simple, POJO classes can be generated easily and works on simple HTTP methods.
• SOAP supports XML data format only	• REST supports any data type such as XML, JSON, image etc.
• SOAP web services are hard to maintain, any change in WSDL contract requires us to create client stubs again and then make changes to client code.	• REST web services are easy to maintain when compared to SOAP, a new method can be added without any change at client side for existing resources.
• SOAP web services can be tested through programs or software such as Soap UI.	• REST can be easily tested through CURL command, Browsers and extensions such as Chrome Postman.