300+ [LATEST] Java Concurrency Interview Questions and Answers

Q1. Why Is It Named Reentrantlock?

It is called ReentrantLock as there is an acquisition count associated with the lock which me when you use lock() method to acquire a lock and you get it then the acquisition count is 1.

A Reentrant lock will also allow the lock holder to enter another block of code with the same lock object as thread already owned it. In that case, if a thread that holds the lock acquires it again, the acquisition count is incremented and the lock then needs to be released twice to truly release the lock.

Q2. What Is The Difference Between Arraylist And Copyonwritearraylist In Java?

ArrayList is not thread-safe whereas CopyOnWriteArrayList is thread-safe and fit for use in multi-threaded environment.

65W SuperVOOC Fast Charging

Get ₹1500 instant discount or selected cards.

OnePlus India

Iterator returned by ArrayList is fail-fast. Iterator returned by CopyOnWriteArrayList is fail-safe.

Performance wise ArrayList is faster as it is not synchronized and there is no added burden of thread-safety. CopyOnWriteArrayList is comparatively slower and if there are lots of writes by various threads that will degrade the performance of the CopyOnwriteArrayList as there will be copies made per mutation.

Q3. What Is Trferqueue In Java?

TrferQueue interface, added in Java 7, extends BlockingQueue interface. The extra functionality provided by TrferQueue interface is that it provides blocking method which will wait until other thread receives your element.

That's how it differs from BlockingQueue where you can only put element into queue or retrieve element from queue and block if queue is full (while you are putting elements) or block if queue is empty (while you are retrieving elements).

TrferQueue has a blocking method trfer(E e) which will ensure that the element is trferred to the consumer, it will wait if required to do so.

Q4. What Is Readwritelock In Java?

In a multi-threading application multiple reads can occur simultaneously for a shared resource. It is only when multiple writes happen simultaneously or intermix of read and write that there is a chance of writing the wrong value or reading the wrong value.

ReadWriteLock uses the same idea in order to boost the performance by having separate pair of locks. A ReadWriteLock maintains a pair of associated locks –

- One for read-only operations;
  and

- One for writing.

The read lock may be held simultaneously by multiple reader threads, so long as there are no writers. The write lock is exclusive.

Q5. What Is Threadpool In Java?

In a large scale application if each task uses its own thread then allocating and deallocating many thread objects creates a significant memory management overhead.

Thread pool as the name suggests provides a set of threads, any task which has to be executed get a thread from this pool.


```
// creating executor with pool of 2 threads
 ExecutorService ex = Executors.newFixedThreadPool(2);
 // running 6 tasks
 ex.execute(new Task());
 ex.execute(new Task());
 ex.execute(new Task());
 ex.execute(new Task());
 ex.execute(new Task());
 ex.execute(new Task());
 //shutting down the executor service
 ex.shutdown();
```

Even if we are running 6 tasks here, all these tasks would be run using the 2 threads from the pool.

Q6. What Is The Difference Between Reentrantlock And Synchronized?

- When you use a synchronized block or method you just need to write synchronized keyword (and provide associated object) acquiring lock and releasing it is done implicitly.

With ReentrantLock acquiring and releasing lock is done by user using lock() and unlock() methods.

- Synchronized forces all lock acquisition and release to occur in a block-structured way which me when multiple locks are acquired they must be released in the opposite order, and all locks must be released in the same lexical scope in which they were acquired.
  ReentrantLock provides more flexibility, it allows a lock to be acquired and released in different scopes, and allowing multiple locks to be acquired and released in any order.

- ReentrantLock provides additional functionality over the use of synchronized methods and statements by providing an option for fairness, providing a non-blocking attempt to acquire a lock (tryLock()), an attempt to acquire the lock that can be interrupted (lockInterruptibly(), and an attempt to acquire the lock that can timeout (tryLock(long, TimeUnit)).

Q7. What Is Priorityblockingqueue In Java Concurrency?

PriorityBlockingQueue class implements the BlockingQueue interface. The elements of the PriorityBlockingQueue are ordered according to their natural ordering, or by a Comparator provided at queue construction time, depending on which of the following constructor is used.

- PriorityBlockingQueue() – Creates a PriorityBlockingQueue with the default initial capacity (11) that orders its elements according to their natural ordering.
- PriorityBlockingQueue(int initialCapacity, Comparator comparator) – Creates a PriorityBlockingQueue with the specified initial capacity that orders its elements according to the specified comparator.

Q8. What Is Atomic Variable In Java?

In Java concurrency classes like AtomicInteger, AtomicLong are provided with a int, long value respectively that may be updated atomically.

These atomic variable classes in Java concurrency like AtomicInteger, AtomicLong uses non-blocking algorithm. These non-blocking algorithms use low-level atomic machine instructions such as compare-and-swap instead of locks to ensure data integrity under concurrent access.

Q9. What Is Cyclicbarrier In Java Concurrency?

CyclicBarrier is useful in scenarios where you want set of threads to wait for each other to reach a common barrier point. When each thread reaches the barrier (common point) you need to call await() method on the CyclicBarrier object. This will suspend the thread until all the thread also call the await() method on the same CyclicBarrier object.

Once all the specified threads have called await() method that will trip the barrier and all threads can resume operation.

The barrier is called cyclic because it can be re-used after the waiting threads are released.

Q10. What Is Countdownlatch In Java Concurrency?

CountDownLatch can be visualized as a latch that is released only after the given number of events occur. CountDownLatch is initialized with that count (given number of events).

Each time one of those events occur count is decremented, for that countdown() method is used. Thread(s) that are waiting for the latch to release (current count reaches zero due to invocations of the countDown()method) are blocked using await() method.

It is useful in the scenario when you want one or more threads to wait until one or more events being performed in other threads complete.

Q11. What Is Linkedtrferqueue In Java?

LinkedTrferQueue, is an implementation of the TrferQueue. It is an unbounded queue and stores elements as linked nodes.

Q12. What Is Linkedblockingdeque In Java?

LinkedBlockingDeque is an implementation of the BlockingDeque interface and it was added in Java @LinkedBlockingDeque is an optionally bounded deque and it stores its elements as linked nodes.

Q13. What Is Arrayblockingqueue In Java Concurrency?

Methods which want to execute the task assigned without relinquishing control to other thread are called blocking methods.

A very relevant example of blocking methods, which most of you would have encountered is read() method of theInputStream class. This method blocks until input data is available, the end of the stream is detected, or an exception is thrown.

Q14. Why Concurrenthashmap Is Faster Than Hashtable In Java?

In HashTable each method is synchronized on a single lock which me at any given time only one thread can enter any method.

ConcurrentHashMap uses separate lock for separate buckets thus locking only a portion of the Map. By default there are 16 buckets and also separate locks for separate buckets. So the default concurrency level is @Thus theoretically at any given time 16 threads can access separate buckets without blocking which improves the performance of the ConcurrentHashMap.

In ConcurrentHashMap performance is further improved by providing read access concurrently without any blocking. Retrieval operations (including get) generally do not block, so may overlap with update operations (including put and remove).

Q15. What Is Exchanger In Java Concurrency?

Exchanger makes it easy for two threads to exchange data between themselves.

Exchanger provides a synchronization point at which two threads can pair and swap elements. Exchanger waits until two separate threads call its exchange() method. When two threads have called the exchange() method, Exchanger will swap the objects presented by the threads.

Q16. If A Countdownlatch Is Initialized With Some Count Let's Say 3 (new Countdownlatch(3)). Do We Need To Have 3 Threads For Countdown In That Case?

No. Same number of threads are not required. A CountDownLatch initialized to N can be used to make one thread wait until N threads have completed some action, or some action has been completed N times.

Q17. What Is Blockingqueue In Java Concurrency?

BlockingQueueinterface is added in Java 5 with in the java.util.concurrent package.

BlockingQueue is a queue that can block the operations. Which me BlockingQueue supports operations that wait for the queue to become non-empty when retrieving an element, and wait for space to become available in the queue when storing an element.

BlockingQueue provides following blocking methods –

- put(E e) – Inserts the specified element into this queue, waiting if necessary for space to become available.
- take() – Retrieves and removes the head of this queue, waiting if necessary until an element becomes available.

Q18. What Is Busy Spinning? When Will You Use Busy Spinning As Waiting Strategy?

An algorithm is called non-blocking if it doesn't block threads in such a way that only one thread has access to the data structure and all the other threads are waiting. Same way failure of any thread in a non-blocking algorithm doesn't mean failure or suspension of other threads.

Implementation of non-blocking data structures in Java like atomic variables or ConcurrentLinkedQueue use an atomic read-modify-write kind of instruction based on compare-and-swap.

Q19. What Is Concurrentlinkeddequeue In Java?

ConcurrentLinkedDeque is an unbounded thread-safeDeque which stores its elements as linked nodes. Since it implements deque interface ConcurrentLinkedDequesupports element insertion and removal at both ends.

ConcurrentLinkedDequeue is thread safe and it doesn't block operations.

Q20. What Is Executors Class?

Executors class provide factory and utility methods for Executors framework classes like Executor, ExecutorService, ScheduledExecutorService, ThreadFactory, and Callable.

Though you can use ThreadPoolExecutor and ScheduledThreadPoolExecutor directly, but the best way to get an executor is to use one of the static factory methods provided by the Executors utility class.

Some of the factory methods –

- static ExecutorService newCachedThreadPool() – Creates a thread pool that creates new threads as needed, but will reuse

previously constructed threads when they are available.

- static ExecutorService newFixedThreadPool(int numThreads) – Creates a thread pool that reuses a fixed number of threads.
- static ScheduledExecutorService newScheduledThreadPool(int numThreads) – Creates a thread pool that can schedule commands to run after a given delay, or to execute periodically.
- newSingleThreadExecutor() – Creates an Executor that uses a single worker thread operating off an unbounded queue.

As example –

ExecutorService ex = Executors.newFixedThreadPool(2);

Q21. What Is Executor In Java Concurrency?

The concurrent API has a feature called executors that provides an alternative to managing threads through the Thread class. At the core of the executors is the Executor interface – An object of type Executor can execute runnable tasks. An Executor is normally used instead of explicitly creating threads.

For example If r is a Runnable object, and e is an Executor object you can replace

(new Thread(r)).start();
with
e.execute(r);

The Executor interface provides a single method, execute –

void execute(Runnable command)

Q22. What Do You Mean By Non-blocking Algorithm/data Structure?

An algorithm is called non-blocking if it doesn't block threads in such a way that only one thread has access to the data structure and all the other threads are waiting. Same way failure of any thread in a non-

blocking algorithm doesn't mean failure or suspension of other threads.

Implementation of non-blocking data structures in Java like atomic variables or ConcurrentLinkedQueue use an atomic read-modify-write kind of instruction based on compare-and-swap.

Q23. What Is Blockingdeque In Java Concurrency?

BlockingDeque interface (added in Java 6) is a Deque that provides additional support for blocking operations. Blocking methods of BlockingDeque interface come in four forms.

- Throw exception – Methods falling in this category will throw exception if blocked.
- Return special value – This type of methods will return some value if need to wait, like false.
- Blocks – This type of methods will wait if necessary for space to become available.
- Times out – This type of methods will block for only a given maximum time limit before giving up.

BlockingDeque is thread safe, does not permit null elements, and may (or may not) be capacity-constrained.

Q24. What Is Concurrenthashmap In Java?

ConcurrentHashMap is also a hash based map like HashMap, how it differs is the locking strategy used by ConcurrentHashMap. Unlike HashTable (or synchronized HashMap) it doesn't synchronize every method on a common lock. ConcurrentHashMap uses separate lock for separate buckets thus locking only a portion of the Map.

That way ConcurrentHashMap depite being a thread safe alternative to HashTable gives much better performance.

Q25. What Is Copyonwritearrayset In Java?

CopyOnWriteArraySet is a thread-safe collection and it internally uses CopyOnWriteArrayList for all of its operations.

Since it uses CopyOnWriteArrayList internally so thread-safety is achieved in the same way in CopyOnwriteArraySet as in CopyOnWriteArrayList – all mutative operations (add, set, and so on) are implemented by making a fresh copy of the underlying array.

The iterator returned by CopyOnwriteArraySet is fail-safe which me any structural modification made to the CopyOnwriteArraySet won't throw ConcurrentModificationException.

Q26. What Is Phaser In Java Concurrency?

Phaser is more suitable for use where it is required to synchronize threads over one or more phases of activity. Though Phaser can be used to synchronize a single phase, in that case it acts more like a CyclicBarrier.

Phaser is reusable (like CyclicBarrier) and more flexible in usage.

The number of parties registered to synchronize on a phaser may vary over time. Tasks may be registered at any time (using methods register(), bulkRegister(int), or by specifying initial number of parties in the constructor). Tasks may also be optionally deregistered upon any arrival (using arriveAndDeregister()).

Q27. What Is Blocking Method In Java?

Methods which want to execute the task assigned without relinquishing control to other thread are called blocking methods.

A very relevant example of blocking methods, which most of you would have encountered is read() method of theInputStream class. This method blocks until input data is available, the end of the stream is detected, or an exception is thrown.

Q28. How To Construct A Thread Pool With 2 Threads That Executes Some Tasks That Return A Value?

You can create a fixed thread pool using the newFixedThreadPool() method of the Executors class.

```
// creating executor with pool of 2 threads
 ExecutorService ex = Executors.newFixedThreadPool(2);
 // running tasks
 Future f1 = ex.submit(new Task());
 Future f2 = ex.submit(new Task());
 try {
  // getting the future value
  System.out.println("Future f1 " + f1.get());
  System.out.println("Future f1 " + f1.get());
 } catch (InterruptedException e) {
  // TODO Auto-generated catch block
  e.printStackTrace();
 } catch (ExecutionException e) {
  // TODO Auto-generated catch block
  e.printStackTrace();
 }
 ex.shutdown();
```

Q29. What Is Executorservice In Java Concurrency?

ExecutorService interface extends Executor interface and provides methods to manage termination and methods that can produce a Future for tracking progress of one or more asynchronous tasks.

ExecutorService has more versatile submit method. Like execute, submit accepts Runnable objects, but also accepts Callable objects, which allow the task to return a value. The submit method returns a Future object, which is used to retrieve the Callable return value and to manage the status of both Callable and Runnable tasks.

Q30. What Is The Difference Between A Countdownlatch And Cyclicbarrier?

- When you are using a CountDownLatch, you specify the number of calls to the countdown() method when creating a CountDownLatch object. What this me is you can use CountDownLatch with only a single thread and using countdown() to decrement as and when the specified even occur.
  When you are using CyclicBarrier you specify the number of threads that should call await() method in order to trip the barrier. That me if you have a CyclicBarrier initialized to 3 that me you should have at least 3 threads to call await().

- CountDownLatch can't be reused, when count reaches zero it cannot be reset.
  CyclicBarrier can be reused after the waiting threads are released.

Q31. What Is Callable And Future In Java Concurrency?

Callable, an interface, was added in Java @It allows you to define a task to be completed by a thread asynchronously. The Callable interface has a call() method, since it is a generic interface so it can return any value (Object, String, Integer etc.) based on how it is initialized. Main feature of the call() method provided by Callable interface is that it can return value.

Future interface – A Future represents the result of an asynchronous computation. When you submit a callable task using the submit() method of the ExecutorService, Future object is returned.

Future provides methods to check if the computation is complete, to wait for its completion, and to retrieve the result of the computation.

get() – get() method retrieves the result of the computation, blocking if necessary for the computation to complete.

Q32. What Is Difference Between Submit() And Execute() Method Of Executor And Executorservice In Java?

- execute() method is provided by Executor interface where as submit() method is provided by ExecutorService.
- execute() method only takes Runnable as argument – execute(Runnable command) and does not return any value.
- ExecutorService has more versatile submit() method. submit() method is overloaded and accepts both Runnable objects and Callable objects, submit also allows the task to return a value (an object of type Future). The Future's get method will return the given result upon successful completion.

Q33. What Is Concurrentlinkedqueue In Java?

ConcurrentLinkedQueue is an unbounded thread-safe queue which stores its elements as linked nodes. This queue orders elements FIFO (first-in-first-out).

It doesn't block operations as it is done in the implementations of BlockingQueue interface like ArrayBlockingQueue.

Q34. What Is A Scheduledexecutorservice?

ScheduledExecutorService extends ExecutorService and provides methods that can schedule commands to run after a given delay, or to execute periodically.

It has methods that execute a Runnable or Callable task after a specified delay.

- schedule(Callable callable, long delay, TimeUnit unit) – Creates and executes a ScheduledFuture that becomes enabled after the given delay.
- schedule(Runnable command, long delay, TimeUnit unit) – Creates and executes a one-shot action that becomes enabled after the given delay.

Q35. What Is Concurrentskiplistset In Java?

ConcurrentSkipListSet implements NavigableSet and it is a sorted set just like TreeSet with added feature of being concurrent.

The elements of the set are kept sorted according to their natural ordering, or by a Comparator provided at set creation time, depending on which constructor is used.

Q36. What Is Concurrentskiplistmap In Java?

ConcurrentSkipListMap implements ConcurrentNavigableMap and it is a sorted map just like TreeMap with the added feature of being concurrent.

ConcurrentSkipListMap is sorted according to the natural ordering of its keys, or by a Comparator provided at map creation time, depending on which constructor is used.

Q37. How Readwritelock Can Help In Reducing Contention Among Multiple Threads? Or What Is Benefit Of Using Readwritelock In Java?

ReadWriteLock provides separate set of locks for reading and writing operations. Where read lock may be held simultaneously by multiple reader threads, so long as there are no writers. The write lock is exclusive.

So read operations are not mutually exclusive. It exploits the fact that while only a single thread at a time (a writer thread) can modify the shared data, in many cases any number of threads can concurrently read the data (hence reader threads).

Thus in the applications where reads are more than writes or duration of reads is more the thread contention will be less as read lock is shared by many thread rather than being mutually exclusive. So you won't have a situation where only one thread is reading and other threads are waiting.

Q38. What Is Reentrantlock In Java?

ReentrantLock is a concrete implementation of the Lock interface which is present injava.util.concurrent.locks package.

Every object created in Java has one mutually exclusive lock associated with it. When you are using synchronized you are using that lock implicitly (with no other feature) whereas when you are using any of the lock implementation (like Reentrant lock) you are using that lock explicitly. Which me there are methods like lock() to acquire the lock and unlock() to release the lock. Along with that ReentrantLock provides many other features like fairness, ability to interrupt and a thread waiting for a lock only for a specified period.

Q39. How Can I Immediately Block A Thread Even If I Am Using Submit Method And Using A Callable Object?

If you would like to immediately block waiting for a task, you can use constructions of the form

result = exec.submit(aCallable).get();

Q40. What Is Synchronous Queue In Java?

SynchronousQueue is an implementation of the BlockingQueue interface. SynchronousQueue does not have any internal capacity, not even a capacity of one. In SynchronousQueue each insert operation must wait for a corresponding remove operation by another thread, and vice versa.

If you put an element in SynchronousQueue using put() method it will wait for another thread to receive it, you can't put any other element in the SynchronousQueue as it is blocked.

Q41. What Is Linkedblockingqueue In Java Concurrency?

LinkedBlockingQueue is an implementation of BlockingQueue interface.

LinkedBlockingQueue internally uses linked nodes to store elements. It is optionally bounded and that's where it differs from ArrayBlockingQueue which is bounded.

Q42. What Is Copyonwritearraylist In Java?

CopyOnWriteArrayList is also an implementation of the List interface but it is a thread safe variant. This thread safety is achieved by making a fresh copy of the underlying array with every mutative operations (add, set, and so on).

Using CopyOnWriteArrayList provides better performance in scenarios where there are more iterations of the list than mutations.

Q43. What Is Reentrantreadwritelock In Java Concurrency?

ReentrantReadWriteLock is an implementation of the ReadWriteLock interface which provides a pair of read-write lock.

- To get a read lock you need to use – rw.readLock().lock();
- To get a write lock you need to use – rw.writeLock().lock();

Where rw is an object of ReentrantReadWriteLock class.

ReentrantReadWriteLock also allows downgrading from the write lock to a read lock. You can first acquire a write lock, then the read lock and then release the write lock.

Q44. How To Shut Down An Executorservice?

An ExecutorService can be shut down, which will cause it to reject new tasks. Two different methods are provided for shutting down an ExecutorService.

The shutdown() method will allow previously submitted tasks to execute before terminating, while the shutdownNow() method prevents waiting tasks from starting and attempts to stop currently executing tasks. Upon termination, an executor has no tasks actively

executing, no tasks awaiting execution, and no new tasks can be submitted.

Q45. Name Any Class That Implements Executor Or Executorservice Interface?

In the Java concurrency there are three pre defined executor classes that implement the Executor and ExecutorService interface.

- ThreadPoolExecutor – Implements the Executor and ExecutorService interfaces and executes the submitted task using one of the pooled thread.
- ScheduledThreadPoolExecutor – It extends ThreadPoolExecutor and also implements the ScheduledExecutorService. This class schedule commands to run after a given delay, or to execute periodically.
- ForkJoinPool – It implements the Executor and ExecutorService interfaces and is used by the Fork/Join Framework.

Q46. What Is Semaphore In Java Concurrency?

The Semaphore class present in java.util.concurrent package is a counting semaphore in which a semaphore, conceptually, maintains a set of permits. Thread that wants to access the shared resource tries to acquire a permit using acquire() method. At that time if the Semaphore's count is greater than zero thread will acquire a permit and Semaphore's count will be decremented by one. If Semaphore's count is zero, when thread calls acquire() method, then the thread will be blocked until a permit is available. When thread is done with the shared resource access, it can call the release() method to release the permit. That results in the Semaphore's count incremented by one.

Q47. What Is Delayqueue In Java Concurrency?

DelayQueue is an unbounded implementation of BlockingQueue interface. DelayQueue can store elements of type Delayed only and an

element can only be retrieved from DelayQueue when its delay has expired.

When you implement Delayed interface two methods have to be implementedgetDelay(TimeUnit unit) and compareTo(T o).

getDelay(TimeUnit unit) – Returns the remaining delay associated with this object, in the given time unit.

Q48. What Is Lock Striping In Concurrent Programming?

The concept of lock striping is to have separate locks for a portion of a data structure where each lock is locking on a variable sized set of independent objects.

That's how ConcurrentHashMap in Java provides synchronization. By default ConcurrentHashMap has 16 buckets and each bucket has its own lock so there are 16 locks too. So the threads which are accessing keys in separate buckets can access them simultaneously.

Q49. What Is The Difference Between Hashmap And Concurrenthashmap In Java?

- ConcurrentHashMap is thread safe and fit for use in a multi-threaded environment whereas HashMap is not thread safe.
- HashMap can be synchronized using the Collections.synchronizedMap() method but that synchronizes all the methods of the HashMap and effectively reduces it to a data structure where one thread can enter at a time.

In ConcurrentHashMap synchronization is done a little differently. Rather than locking every method on a common lock, ConcurrentHashMap uses separate lock for separate buckets thus locking only a portion of the Map.

---- >> Related Posts Of Above Questions :::

------>>[MOST IMPORTANT]<<------

1. 300+ [LATEST] Java Developer Interview Questions and Answers
2. 300+ [LATEST] Java-multithreading Interview Questions and Answers
3. 300+ [LATEST] Java Threads Interview Questions and Answers
4. 300+ TOP Java Multithreading Interview Questions and Answers
5. 300+ [REAL TIME] Executors Java Interview Questions
6. 300+ [LATEST] Ericsson Java Interview Questions and Answers
7. 300+ [LATEST] Aricent Java Interview Questions and Answers
8. 300+ [LATEST] Java.lang Interview Questions and Answers
9. 300+ [LATEST] Java Collections Interview Questions and Answers
10. 300+ [LATEST] Java Collections Framework Interview Questions and Answers
11. 300+ [LATEST] Java Hadoop Developer Interview Questions and Answers
12. 300+ [MOSK ASKED] Wipro Java Interview Questions and Answers
13. 300+ [LATEST] C# Developer Multi Threading Interview Questions and Answers
14. 300+ TOP ADVANCED JAVA Interview Questions and Answers
15. 300+ [LATEST] Java String Interview Questions and Answers
16. 300+ [LATEST] Java J2ee Technical Support Engineer Interview Questions and Answers
17. 300+ [LATEST] Java Constructor Interview Questions and Answers
18. 300+ [LATEST] Java & Ejb & Spring Interview Questions and Answers

19. **300+ [MOSK ASKED] Deloitte Java Interview Questions and Answers**
20. **300+ [LATEST] Java Abstraction Interview Questions and Answers**