300+ [REAL TIME] Executors Java Interview Questions

1. What Do You Understand By Executor Framework In Java?

Executor Framework in java has been introduced in JDK 5. Executor Framework handles creation of thread, creating the thread pool and checking health while running and also terminates if needed.

2. What Is The Role Of Executorservice In Java?

ExecutorService provides different methods to start and terminate thread. There are two methods execute() and submit() in ExecutorService. Execute() method is used for threads which is Runnable and submit() method is used for Callable threads.

Adv Java Interview Questions

3. What Is Executors In Java Executor Framework?

Executors is a factory that provides the methods to return ExecutorService, ScheduledExecutorService, ThreadFactory. Find some method details.

newFixedThreadPool():
It returns the pool with fixed number of size. We need to pass the number of threads to this method. If concurrently task are submitted more than the pool size, then rest of task need to wait in queue. It returns ExecutorService.

newScheduledThreadPool:
This also creates a fixed size pool but it can schedule the thread to run after some defined delay. It is useful to schedule the task. It returns ScheduledExecutorService.

newCachedThreadPool():
There is no fixed size of this pool. Thread will be created at run time and if there is no task it will alive for 60 second and then

die. For short lived threads this pool works good. It returns ExecutorService.

4. What Is The Role Of Futuretask And Future In Java?

FutureTask is a cancellable asynchronous computation in java. It can cancel the task which is running. Once the FutureTask will be cancelled, it cannot be restarted. Future is result of asynchronous computation. Future checks if task is complete and if completed it gets the output.

Adv Java Tutorial

5. What Is Difference Between Shutdownnow() And Shutdown() In Executor Framework In Java?

shutdown() and shutdownNow() methods belongs to ExecutorService. shutdown() method tries to stop the threads and do not accept new task to execute but it completes the execution which has been submitted. shutdownNow() methods also tries to stop the running threads and will not execute any task which has been submitted but not started.

J2EE Interview Questions

6. How To Terminate A Thread In Executor Framework In Java?

ExecutorService provides a method awaitTermination(long timeout, TimeUnit unit) that takes time and unit of time as an arguments. After that time thread pool is terminated. Suppose we need to terminate a task just now, then we can do as

ExecutorService.awaitTermination(0, TimeUnit.SECONDS)

7. What Is The Role Of Executors.privileged Threadfactory() In Executor Framework?

privileged ThreadFactory returns a thread factory that creates thread with same permission as main thread.

J2EE Tutorial
Core Java Interview Questions

8. What Is The Role Of Executors.unconfigurable Executorservice In Executor Framework?

unconfigurable ExecutorService returns an object that delegates all methods of ExecutorService to the given executor so that any other method cannot accessed by cast.

9. What Are The Different Policy In Executor Framework?

There are different policy within ThreadPoolExecutor in java.

ThreadPoolExecutor.AbortPolicy :
AbortPolicy is a handler for rejected task. It handles those task which has been rejected.

ThreadPoolExecutor.CallerRunsPolicy :
This also handles the rejected task and runs the rejected task directly.

ThreadPoolExecutor.DiscardOldestPolicy :
This handles those rejected task that is oldest and unhandled. It discards those that oldest task.

ThreadPoolExecutor.DiscardPolicy :
This is the handler for those rejected task that are rejected silently.

JSP Interview Questions

10. How To Get Return Value Of A Callable Thread In Java Executor Framework?

Using Future, we can get the return value of callable thread.

```
ExecutorService exService =
Executors.newCachedThreadPool();

Future future=exService.submit(new CallableThread());

int val=future.get();
```

Core Java Tutorial

11. Write A Program Using Executor In Java Or Example Of Thread Pool In Java?

```
public class EmailSender  implements Runnable {

    String message;

    EmailSender  (String message) {

        this.message = message;

    }

    public void run() {

        try {

            sendEmail(message);

        } catch (Exception e) {

            e.printStackTrace();

        }

    }
```

```
    private void sendEmail(String message2)  {

      System.out.println("Sending Email" + message);

   }

}

public class TestThreadPool {

   public static void main(String[] args) {

 //Let us start the Worker Threads

    static final Integer NTHREDS=5;

          ExecutorService executor =
Executors.newFixedThreadPool(NTHREDS);

for (int i = 0; i < 6; i++) {

   Runnable worker = new EmailSender("Hi Email from java");

               executor.execute(worker);

          }

          // This will make the executor accept no new threads

          // and finish all existing threads in the queue

          executor.shutdoOWN();

 // Wait until all threads are finish

          while (!executor.isTerminated()) {

}
```

```
}

}
```

Java-Springs Interview Questions

12. What Is Semaphore?

Semaphore can be used to create a set of Children Threads even when the size of the Threads to be created is not known fore hand. This is because a Semaphore can wait until a number of releases have been made but that number is not required to initialize the Semaphore. Semaphores can be used in other scenarios such as Synchronizing between different threads such as Publisher, Subscriber scenario. Semaphore is a technique used to control access to common resource for competeing multiple processes. Semaphore maintains a counter which keeps track of the number of resources available. When a process requests access to resource, semaphore checks the variable count and if it is less than total count then grants access and subsequently reduces the available count.

Semaphore is just a gatekeeper guarding the resources. If available grants access and otherwise asks the processes to wait. When the resource count is arbitrary then this is called counting semaphore. If resource count is only one and the state value is restricted to on/off, then it is called binary semaphore.

Adv Java Interview Questions

13. What Is Cyclicbarrier And Countdownlatch?

CyclicBarrier and CountDownLatch in Java is a synchronizer introduced in JDK 5 on java.util.Concurrent package.Both CyclicBarrier and CountDownLatch are used to implement a scenario where one Thread waits for one or more Thread to complete there job before starts processing but there is one Difference between CountDownLatch and CyclicBarrier in Java

which separates them apart and that is, you can not reuse same CountDownLatch instance oncecount reaches to zero and latch is open, on the other hand CyclicBarrier can be reused by resetting Barrier, Once barrier is broken.

JSP Tutorial

14. How Countdownlatch Work In Java?

Any thread, usually main thread of application, which calls CountDownLatch.await() will wait until count reaches zero or its interrupted by another Thread. All other thread are required to do count down by calling CountDownLatch.countDown() once they are completed or ready to the job. as soon as count reaches zero, Thread awaiting starts running. One of the disadvantage of CountDownLatch is that its not reusable once count reaches to zero

you can not use CountDownLatch any more, but don't worry Java concurrency API has another concurrent utility called CyclicBarrier for such requirements Java program requires 3 services namely CacheService, AlertService and ValidationService to be started and ready before application can handle any request and this is achieved by

using CountDownLatch in Java.

```
import java.util.Date;

import java.util.concurrent.CountDownLatch;

import java.util.logging.Level;

import java.util.logging.Logger;

public class CountDownLatchDemo {

 public static void main(String args[]) {
```

```
final CountDownLatch latch = new CountDownLatch(3);

Thread cacheService = new Thread(new
Service("CacheService", 1000, latch));

Thread alertService = new Thread(new
Service("AlertService", 1000, latch));

Thread validationService = new Thread(new
Service("ValidationService", 1000, latch));

cacheService.start(); //separate thread will initialize
CacheService

alertService.start(); //another thread for AlertService
initialization

validationService.start();

// application should not start processing any thread until
all service is up

// and ready to do there job.main thread will start with
count 3

// and wait until count reaches zero. Each thread once up
and read will do

// a count down.

//count is 3 since we have 3 Threads

try{

    latch.await();  //main thread is waiting on
CountDownLatch to finish
```

```
            System.out.println("All services are up, Application is
starting now");

        }catch(InterruptedException ie){

            ie.printStackTrace();

        }

}

}

/**

 * Class executed by Thread using CountDownLatch
synchronizer.

 */

class Service implements Runnable{

    private final String name;

    private final int timeToStart;

    private final CountDownLatch latch;

    public Service(String name, int timeToStart,
CountDownLatch latch){

        this.name = name;

        this.timeToStart = timeToStart;

        this.latch = latch;

    }
```

```
  @Override
   public void run() {
      try {
         Thread.sleep(timeToStart);
      } catch (InterruptedException ex) {

Logger.getLogger(Service.class.getName()).log(Level.SEVERE,
null, ex);
      }
      System.out.println( name + " is Up");
      latch.countDown(); //reduce count of CountDownLatch by
1
   }
}
```

Output of Program:

```
ValidationService is Up
AlertService is Up
CacheService is Up
All services are up, Application is starting now
```

Points To Remember

1. You can not reuse CountDownLatch once count is reaches to zero, this is the main difference between CountDownLatch and CyclicBarrier.
2. Main Thread wait on Latch by calling CountDownLatch.await() method while other thread calls CountDownLatch.countDown() to inform that they have completed.

15. What Is Difference Between Lock Interface And Synchronized Keyword?

The main differences between a Lock and a synchronized block are:

1. Having a timeout trying to get access to a synchronized block is not possible. Using Lock.tryLock(long timeout, TimeUnit timeUnit), it is possible.
2. The synchronized block must be fully contained within a single method. A Lock can have it's calls to lock() and unlock() in separate methods.

JMS(Java Message Service) Interview Questions

16. What Is Condition?

Condition objects are similar to Object wait-notify model with additional feature to create different sets of wait. A Condition object is always created by Lock object. Some of the important methods are await() that is similar to wait() and signal(), signalAll() that is similar to notify() and notifyAll() methods.

Java-Springs Tutorial

17. What Is Readwritelock?

It contains a pair of associated locks, one for read-only operations and another one for writing. The read lock may be held simultaneously by multiple reader threads as long as there are no writer threads. The write lock is exclusive.

Java applet Interview Questions

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

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

1. **300+ [LATEST] Java Concurrency Interview Questions and Answers**
2. **300+ [LATEST] Java Threads Interview Questions and Answers**
3. **300+ TOP Java Multithreading Interview Questions and Answers**
4. **300+ [LATEST] Java-multithreading Interview Questions and Answers**
5. **300+ [LATEST] Java.lang Interview Questions and Answers**
6. **250+ TOP MCQs on Java.lang – ThreadGroup class & Runnable Interface and Answers**
7. **300+ [LATEST] Java String Interview Questions and Answers**
8. **300+ [LATEST] Java Developer Interview Questions and Answers**
9. **250+ TOP MCQs on Multithreading and Answers**
10. **300+ [LATEST] Ericsson Java Interview Questions and Answers**
11. **300+ [MOSK ASKED] Deloitte Java Interview Questions and Answers**
12. **300+ [LATEST] Aricent Java Interview Questions and Answers**
13. **250+ TOP MCQs on Thread Pools and Answers**
14. **300+ [MOSK ASKED] Wipro Java Interview Questions and Answers**
15. **300+ [MOSK ASKED] Infosys Core Java Interview Questions and Answers**
16. **250+ TOP MCQs on Multithreaded Programming and Answers**

17. **300+ [LATEST] Java J2ee Technical Support Engineer Interview Questions and Answers**
18. **300+ TOP ADVANCED JAVA Interview Questions and Answers**
19. **Which category the java thread do fall in?**
20. **300+ [MOSK ASKED] Birlasoft Java Interview Questions and Answers**