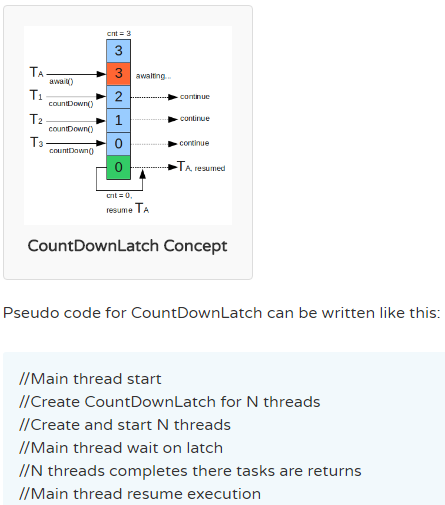
## **What is CountDownLatch?**

CountDownLatch was **introduced with JDK 1.5 along with other concurrent utilities like CyclicBarrier, Semaphore, [ConcurrentHashMap](https://howtodoinjava.com/core-java/collections/best-practices-for-using-concurrenthashmap/" \o "Best practices for using ConcurrentHashMap) and [BlockingQueue](https://howtodoinjava.com/java-5/how-to-use-blockingqueue-and-threadpoolexecutor-in-java/" \o "How to use BlockingQueue and ThreadPoolExecutor in java)** in java.util.concurrent package. This class **enables a java thread to wait until other set of threads completes** their tasks. e.g. Application’s main thread want to wait, till other service threads which are responsible for starting framework services have completed started all services.

CountDownLatch works by having a counter initialized with number of threads, which is decremented each time a thread complete its execution. When count reaches to zero, it means all threads have completed their execution, and thread waiting on latch resume the execution.



[Executor](http://docs.oracle.com/javase/7/docs/api/java/util/concurrent/Executor.html) just executes stuff you give it.

[ExecutorService](http://docs.oracle.com/javase/7/docs/api/java/util/concurrent/ExecutorService.html) adds startup, shutdown, and the ability to wait for and look at the status of jobs you've submitted for execution on top of Executor (which it extends).

## **Example application using CountDownLatch**

In this example, I have simulated an application startup class which starts N threads that will check for external systems and report back to latch, on which startup class is waiting. As soon as all services are verified and checked, startup proceeds.

**BaseHealthChecker.java** : This class is a Runnable and parent for all specific external service health checkers. This remove the code duplicacy and central control over latch.

|  |
| --- |
| public abstract class BaseHealthChecker implements Runnable {        private CountDownLatch \_latch;      private String \_serviceName;      private boolean \_serviceUp;        //Get latch object in constructor so that after completing the task, thread can countDown() the latch      public BaseHealthChecker(String serviceName, CountDownLatch latch)      {          super();          this.\_latch = latch;          this.\_serviceName = serviceName;          this.\_serviceUp = false;      }        @Override      public void run() {          try {              verifyService();              \_serviceUp = true;          } catch (Throwable t) {              t.printStackTrace(System.err);              \_serviceUp = false;          } finally {              if(\_latch != null) {                  \_latch.countDown();              }          }      }        public String getServiceName() {          return \_serviceName;      }        public boolean isServiceUp() {          return \_serviceUp;      }      //This methos needs to be implemented by all specific service checker      public abstract void verifyService();  } |

**NetworkHealthChecker.java** : This class extends BaseHealthChecker and needs to provide only implementation of verifyService() method. **DatabaseHealthChecker.java** and **CacheHealthChecker.java** are same as NetworkHealthChecker.java apart from their service names and sleep time.

|  |
| --- |
| public class NetworkHealthChecker extends BaseHealthChecker  {      public NetworkHealthChecker (CountDownLatch latch)  {          super("Network Service", latch);      }        @Override      public void verifyService()      {          System.out.println("Checking " + this.getServiceName());          try          {              Thread.sleep(7000);          }          catch (InterruptedException e)          {              e.printStackTrace();          }          System.out.println(this.getServiceName() + " is UP");      }  } |

**ApplicationStartupUtil.java** : This clas is main startup class which initilizes the latch and wait of this latch till all services are checked.

|  |
| --- |
| public class ApplicationStartupUtil  {      //List of service checkers      private static List<BaseHealthChecker> \_services;        //This latch will be used to wait on      private static CountDownLatch \_latch;        private ApplicationStartupUtil()      {      }        private final static ApplicationStartupUtil INSTANCE = new ApplicationStartupUtil();        public static ApplicationStartupUtil getInstance()      {          return INSTANCE;      }        public static boolean checkExternalServices() throws Exception      {          //Initialize the latch with number of service checkers          \_latch = new CountDownLatch(3);            //All add checker in lists          \_services = new ArrayList<BaseHealthChecker>();          \_services.add(new NetworkHealthChecker(\_latch));          \_services.add(new CacheHealthChecker(\_latch));          \_services.add(new DatabaseHealthChecker(\_latch));            //Start service checkers using executor framework          Executor executor = Executors.newFixedThreadPool(\_services.size());            for(final BaseHealthChecker v : \_services)          {              executor.execute(v);          }            //Now wait till all services are checked          \_latch.await();            //Services are file and now proceed startup          for(final BaseHealthChecker v : \_services)          {              if( ! v.isServiceUp())              {                  return false;              }          }          return true;      }  } |

Now you can write any test class to check the functionality of latch.

|  |
| --- |
| public class Main {      public static void main(String[] args)      {          boolean result = false;          try {              result = ApplicationStartupUtil.checkExternalServices();          } catch (Exception e) {              e.printStackTrace();          }          System.out.println("External services validation completed !! Result was :: "+ result);      }  }    Output in console:    Checking Network Service  Checking Cache Service  Checking Database Service  Database Service is UP  Cache Service is UP  Network Service is UP  External services validation completed !! Result was :: true |