Alok Sarkar

[Company name]  [Company address]

JAVA - THREADS

**What is CountDownLatch in Java?**

**Answer.** CountDownLatch in Java is like a synchronizer. It allows a thread to wait for one or more threads before starting the process. CountDownLatch is a very crucial requirement and we often need it in server-side core Java applications. Having this functionality built-in as CountDownLatch simplifies the development.

CountDownLatch in Java was introduced on Java 5 along with other concurrent utilities like CyclicBarrier, Semaphore, ConcurrentHashMap, and BlockingQueue. These are all present in the java.util.concurrent package.

**What is CyclicBarrier in Java?**

**Answer.** The CyclicBarrier class is present in the java.util.concurrent package. It is a synchronization mechanism that synchronizes threads progressing through some algorithm. CyclicBarrier class is a barrier at which all the threads until all threads reach it.

A CyclicBarrier is used when multiple threads carry out different subtasks and there is a need to combine the output of these subtasks to form the final output. After completing its execution, threads call the await() method and wait for other threads to reach the barrier.

**Differentiate between CountDownLatch and CyclicBarrier in Java?**

**Answer.** Both CyclicBarrier and CountDownLatch are useful tools for synchronization between multiple threads. However, they are different in terms of the functionality they provide.

CountDownLatch allows one or more than one thread to wait for a number of tasks to complete while CyclicBarrier allows a number of threads to wait on each other. In short, CountDownLatch maintains a count of tasks whereas CyclicBarrier maintains a count of threads.

When the barrier trips in the CyclicBarrier, the count resets to its original value. CountDownLatch is different because the count never resets to the original value.

**What is the difference between a synchronized method and a synchronized block?**

**Answer.** The differences between a synchronized method and a synchronized block are:

1. A synchronized method uses the method receiver as a lock. It uses ‘this’ for non-static methods and the enclosing class for static methods. Whereas, the synchronized blocks use the expression as a lock.

2. A synchronized method locks on that object only in which the method is present, while a synchronized block can lock on any object.

3. The synchronized method holds the lock throughout the method scope. While the lock is held only during that block scope, also known as the critical section in the synchronized block.

4. If the expression provided as parameter evaluates to null, the synchronized block can throw NullPointerException while this is not the case with synchronized methods.

5. The synchronized block offers granular control overlock because we can use any lock to provide mutual exclusion to critical section code. The synchronized method always locks either class level lock on the current object, if its static synchronized method.

**What is a ThreadFactory?**

**Answer.** A ThreadFactory is an interface in Java that is used to create threads rather than explicitly creating threads using the new Thread(). It is an object that creates new threads on demand. The Thread factory removes hardwiring of calls to new Thread and enables applications to use special thread subclasses, and priorities, etc.