Executor

1. ForkJoinPool

All threads in the pool attempt to find and execute tasks submitted to the pool and/or created by other active tasks (eventually blocking waiting for work if none exist). It take advantage of the multiple processor and use all the available processing power to enhance performance.

Links:
https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/ForkJoinPool.html
https://docs.oracle.com/javase/tutorial/essential/concurrency/forkjoin.html

2. AbstractExecutorService (provide default implementations of executor service)

This method helps to manage or produce a Future for tracking the progress of 1 or more asynchronous tasks. The main reason of using this method is so that it can create and return a Future that can be used to cancel execution and/or wait for completion.

Links
https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/AbstractExecutorService.html

3. ScheduledThreadPoolExecutor

It helps to schedule the commands ad more flexible as compared to ThreadPoolExecutor as it helps to schedule commands by executing it periodically. However, once the task is being delayed, there is no guarantee when the task will be executed. A new thread will be created only if the queue is full.

Links
https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/ScheduledThreadPoolExecutor.html

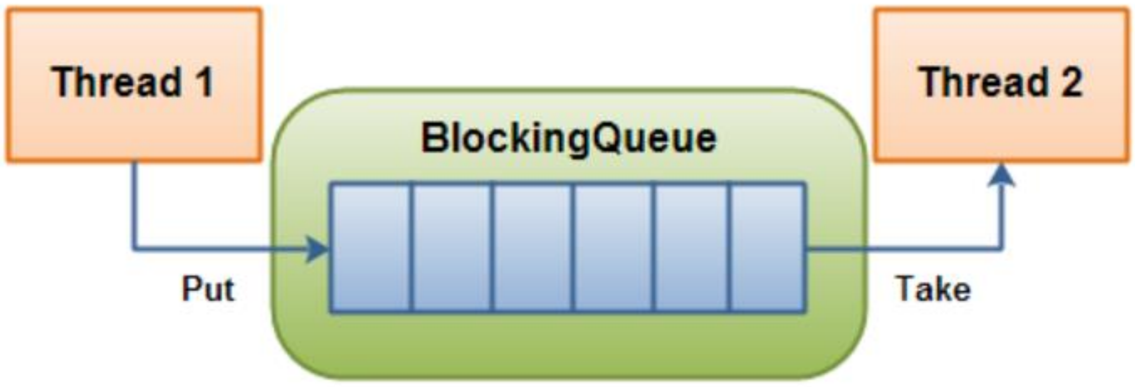
4. ThreadPoolExecutor (extends AbstractExecutorService)

It helps to improve the performance when handling the large number of asynchronous tasks and manage the thread when executing a collection of tasks.

Links
https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/ThreadPoolExecutor.html

Blocking Queue

1. Blocking queue



This thread will produce new objects and insert them into the queue until it reaches its upper limit based on what it has.

Link:
http://tutorials.jenkov.com/java-util-concurrent/blockingqueue.html

2. Synchronous Queue

This queue implements BlockingQueue that consist of only single element. Thread that insert the element into a queue is being block and can only be unblocked until another thread takes it from the queue.

Link:
http://tutorials.jenkov.com/java-util-concurrent/synchronousqueue.html

3. Priority Blocking Queue

This queue is unbounded concurrent queue which uses the same ordering rule as `java.util.PriorityQueue` class.

Link:
http://tutorials.jenkov.com/java-util-concurrent/priorityblockingqueue.html

4. Linked Blocking Queue

This queue keeps the elements in a linked nodes structure which have an upper bound or the integer max value if not specify. It stores it in FIFO (first in, first out) order.

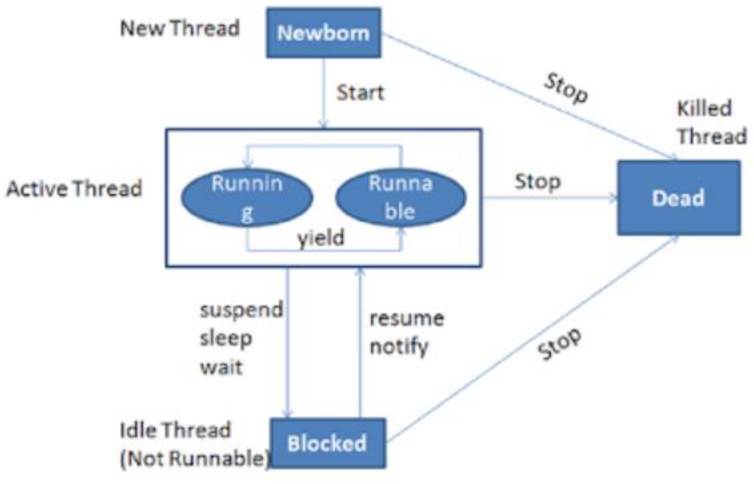
Link:
http://tutorials.jenkov.com/java-util-concurrent/linkedblockingqueue.html

5. Array Blocking Queue

This queue is a bounded queue which means that there is a limit amount of elements that it can store. You can set the limit as the upper bound and cannot be changed thereafter. It stores it in FIFO 9first in, first out) order.

Link:
http://tutorials.jenkov.com/java-util-concurrent/arrayblockingqueue.html

How To Kill A Thread Heading



As there is no stop method thus you will need to set it to Boolean = true in order to stop a running thread.

Link
https://stackoverflow.com/questions/5915156/how-can-i-kill-a-thread-without-using-stop
https://stackoverflow.com/questions/671049/how-do-you-kill-a-thread-in-java
https://stackoverflow.com/questions/23317332/alternate-to-thread-stop-in-java-8
https://www.quora.com/How-do-you-destroy-a-thread-in-Java
http://www.java67.com/2015/07/how-to-stop-thread-in-java-example.html

Thread Dependency and Termination

*Research done to terminate the thread.

Link:
https://stackoverflow.com/questions/21826369/one-thread-dependent-on-the-other
http://www.baeldung.com/java-countdown-latch