



CompletionService Interface

This lesson talks about how to batch multiple tasks together

CompletionService Interface

In the previous lesson we discussed how tasks can be submitted to executors but imagine a scenario where you want to submit hundreds or thousands of tasks. You'll retrieve the future objects returned from the submit calls and then poll all of them in a loop to check which one is done and then take appropriate action. Java offers a better way to address this use case through the **CompletionService** interface. You can use the **ExecutorCompletionService** as a concrete implementation of the interface.

The completion service is a combination of a blocking queue and an executor. Tasks are submitted to the queue and then the queue can be polled for completed tasks. The service exposes two methods, one **poll** which returns null if no task is completed or none were submitted and two **take** which blocks till a completed task is available.

Below is an example program that demonstrates the use of completion service.

```
33  
34  
35  
36     }  
37
```



```
38      Exec
39      Exec
40
41
42      // S
43      for
44
45      }
46
47      // w
48      int
49      while
50
51
52
53
54
55      }
56
57      thre
58  }
59
60 }
```



Interviewing soon? We've partnered with Hired so that companies apply to you
[utm_source=educative&utm_medium=lesson&utm_location=CA&utm_campaign=hired](https://www.educative.io/courses/java-multithreading-for-senior-engineering-interviews/7n94JNyPOMw?utm_source=educative&utm_medium=lesson&utm_location=CA&utm_campaign=hired)

[← Back](#)[Next →](#)

Future Interface

ThreadLocal

☒ Mark as Completed[Report an Issue](#)[Ask a Question](#)https://discuss.educative.io/tag/completion-service-interface__java-thread-basics__java-multithreading-for-senior-engineering-interviews

