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.

```
import java.util.Random;
import java.util.concurrent.ExecutorCompletionService;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.Future;


class Demonstration {

  static Random random = new Random(System.currentTimeMillis());

  public static void main( String args[] ) throws Exception {
        completionServiceExample();
    }
```

```
  static void completionServiceExample() throws Exception {

        class TrivialTask implements Runnable {

            int n;

            public TrivialTask(int n) {
                this.n = n;
            }

            public void run() {
                try {
                    // sleep for one second
                    Thread.sleep(random.nextInt(101));
                    System.out.println(n*n);
                } catch (InterruptedException ie) {
                    // swallow exception
                }
            }
        }

        ExecutorService threadPool = Executors.newFixedThreadPool(3);
        ExecutorCompletionService<Integer> service =
                new ExecutorCompletionService<Integer>(threadPool);

        // Submit 10 trivial tasks.
        for (int i = 0; i < 10; i++) {
            service.submit(new TrivialTask(i), new Integer(i));
        }

        // wait for all tasks to get done
        int count = 10;
        while (count != 0) {
            Future<Integer> f = service.poll();
            if (f != null) {
                System.out.println("Thread" + f.get() + " got done.");
                count--;
            }
        }

        threadPool.shutdown();
    }

}
```