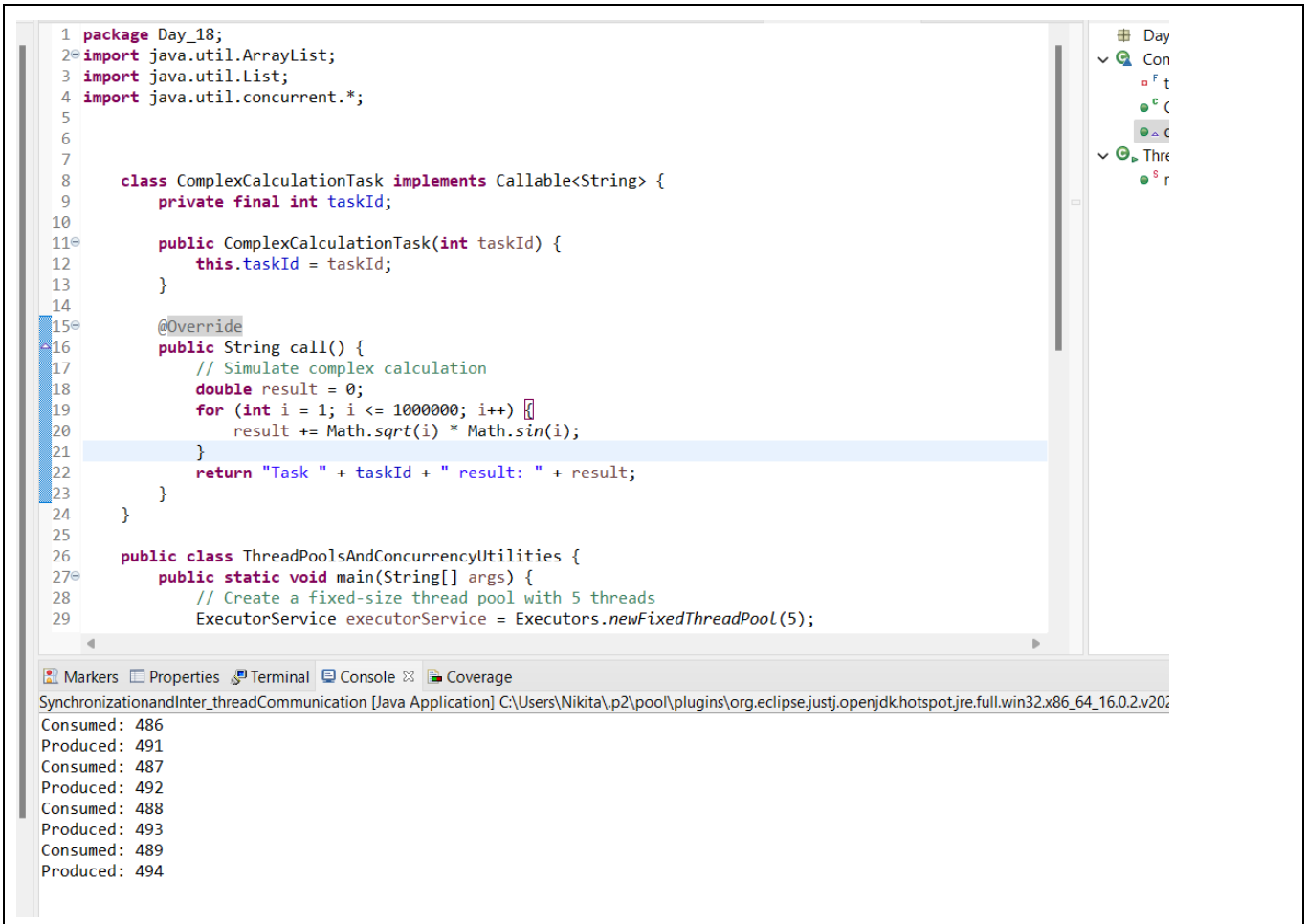
Thread Pools and Concurrency Utilities

Create a fixed-size thread pool and submit multiple tasks that perform complex calculations or I/O operations and observe the execution.

```java
package Day_18;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.*;



	class ComplexCalculationTask implements Callable<String> {
	    private final int taskId;

	    public ComplexCalculationTask(int taskId) {
	        this.taskId = taskId;
	    }

	    @Override
	    public String call() {
	        // Simulate complex calculation
	        double result = 0;
	        for (int i = 1; i <= 1000000; i++) {
	            result += Math.sqrt(i) * Math.sin(i);
	        }
	        return "Task " + taskId + " result: " + result;
	    }
	}

	public class ThreadPoolsAndConcurrencyUtilities {
	    public static void main(String[] args) {
	        // Create a fixed-size thread pool with 5 threads
	        ExecutorService executorService = Executors.newFixedThreadPool(5);

	        // List to hold Future objects representing submitted tasks
	        List<Future<String>> futures = new ArrayList<>();

	        // Submit 10 tasks to the thread pool
	        for (int i = 1; i <= 10; i++) {
	            ComplexCalculationTask task = new ComplexCalculationTask(i);
	            Future<String> future = executorService.submit(task);
	            futures.add(future);
	        }

	        // Retrieve and print the results of the tasks
	        for (Future<String> future : futures) {
	            try {
	                System.out.println(future.get());
	            } catch (InterruptedException | ExecutionException e) {
	                e.printStackTrace();
	            }
	        }

	        // Shutdown the executor service
	        executorService.shutdown();
	    }
	}
```



```java
package Day_18;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.*;



    class ComplexCalculationTask implements Callable<String> {
        private final int taskId;

        public ComplexCalculationTask(int taskId) {
            this.taskId = taskId;
        }

        @Override
        public String call() {
            // Simulate complex calculation
            double result = 0;
            for (int i = 1; i <= 1000000; i++) {
                result += Math.sqrt(i) * Math.sin(i);
            }
            return "Task " + taskId + " result: " + result;
        }
    }

    public class ThreadPoolsAndConcurrencyUtilities {
        public static void main(String[] args) {
            // Create a fixed-size thread pool with 5 threads
            ExecutorService executorService = Executors.newFixedThreadPool(5);
```

```
Markers  Properties  Terminal  Console  Coverage
SynchronizationandInter_threadCommunication [Java Application] C:\Users\Nikita\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.2.v202
Consumed: 486
Produced: 491
Consumed: 487
Produced: 492
Consumed: 488
Produced: 493
Consumed: 489
Produced: 494
```