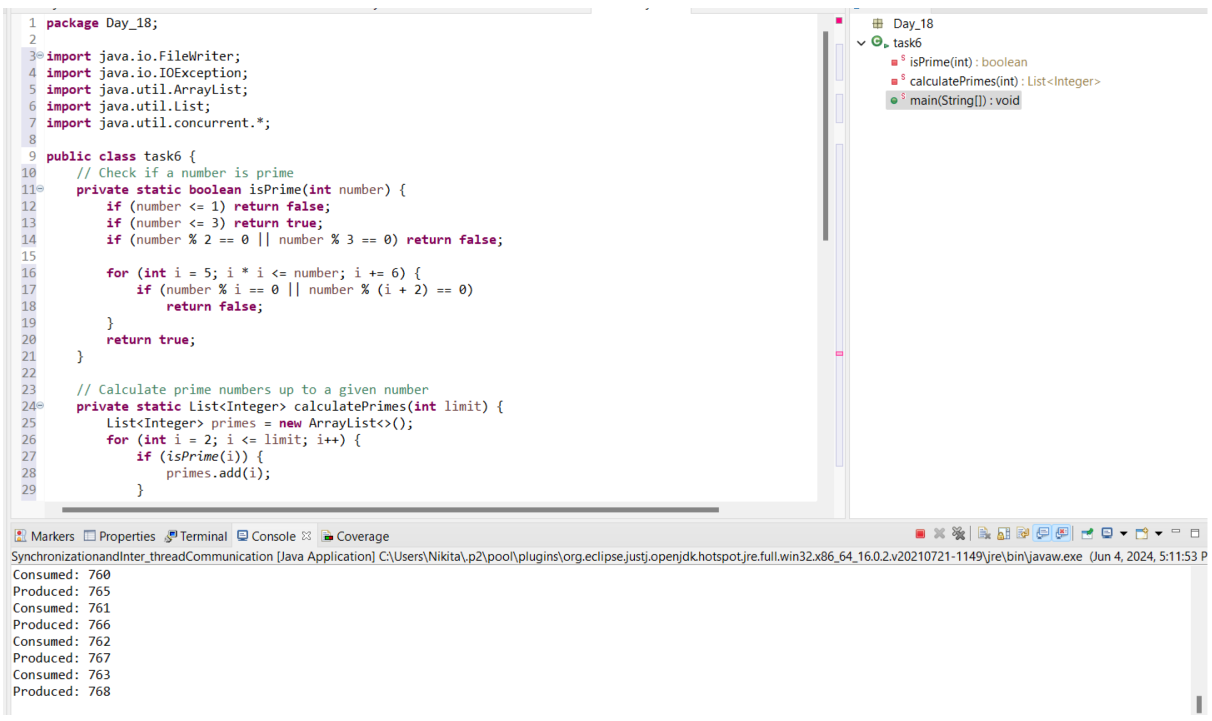
Executors, Concurrent Collections, CompletableFuture

Use an ExecutorService to parallelize a task that calculates prime numbers up to a given number and then use CompletableFuture to write the results to a file asynchronously.

```
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.*;

public class Task6 {
  // Check if a number is prime
  private static boolean isPrime(int number) {
    if (number <= 1) return false;
    if (number <= 3) return true;
    if (number % 2 == 0 || number % 3 == 0) return false;

    for (int i = 5; i * i <= number; i += 6) {
      if (number % i == 0 || number % (i + 2) == 0)
        return false;
    }
    return true;
  }

  // Calculate prime numbers up to a given number
  private static List<Integer> calculatePrimes(int limit) {
    List<Integer> primes = new ArrayList<>();
    for (int i = 2; i <= limit; i++) {
      if (isPrime(i)) {
        primes.add(i);
      }
    }
    return primes;
  }
```

```
    public static void main(String[] args) {
        int limit = 100; // Example limit for calculating primes
        String filename = "primes.txt"; // Output file name

        // Create an ExecutorService with a fixed thread pool
        ExecutorService executorService = Executors.newFixedThreadPool(Runtime.getRuntime().availableProcessors());

        // Submit the prime number calculation task
        Future<List<Integer>> primeTask = executorService.submit(() -> calculatePrimes(limit));

        // Use CompletableFuture to asynchronously write results to a file
        primeTask.thenAcceptAsync(primes -> {
            try (FileWriter writer = new FileWriter(filename)) {
                for (int prime : primes) {
                    writer.write(prime + "\n");
                }
                System.out.println("Primes written to " + filename);
            } catch (IOException e) {
                System.err.println("Error writing to file: " + e.getMessage());
            }
        }).exceptionally(ex -> {
            System.err.println("Error occurred during prime number calculation: " + ex.getMessage());
            return null;
        }).thenRun(() -> {
            // Shutdown the executor service after completion
            executorService.shutdown();
        });
    }
}
```



```java
package Day_18;

import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.*;

public class task6 {
    // Check if a number is prime
    private static boolean isPrime(int number) {
        if (number <= 1) return false;
        if (number <= 3) return true;
        if (number % 2 == 0 || number % 3 == 0) return false;

        for (int i = 5; i * i <= number; i += 6) {
            if (number % i == 0 || number % (i + 2) == 0)
                return false;
        }
        return true;
    }

    // Calculate prime numbers up to a given number
    private static List<Integer> calculatePrimes(int limit) {
        List<Integer> primes = new ArrayList<>();
        for (int i = 2; i <= limit; i++) {
            if (isPrime(i)) {
                primes.add(i);
            }
```

SynchronizationandInter_threadCommunication [Java Application] C:\Users\Nikita\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.2.v20210721-1149\jre\bin\javaw.exe (Jun 4, 2024, 5:11:53 P

```
Consumed: 760
Produced: 765
Consumed: 761
Produced: 766
Consumed: 762
Produced: 767
Consumed: 763
Produced: 768
```