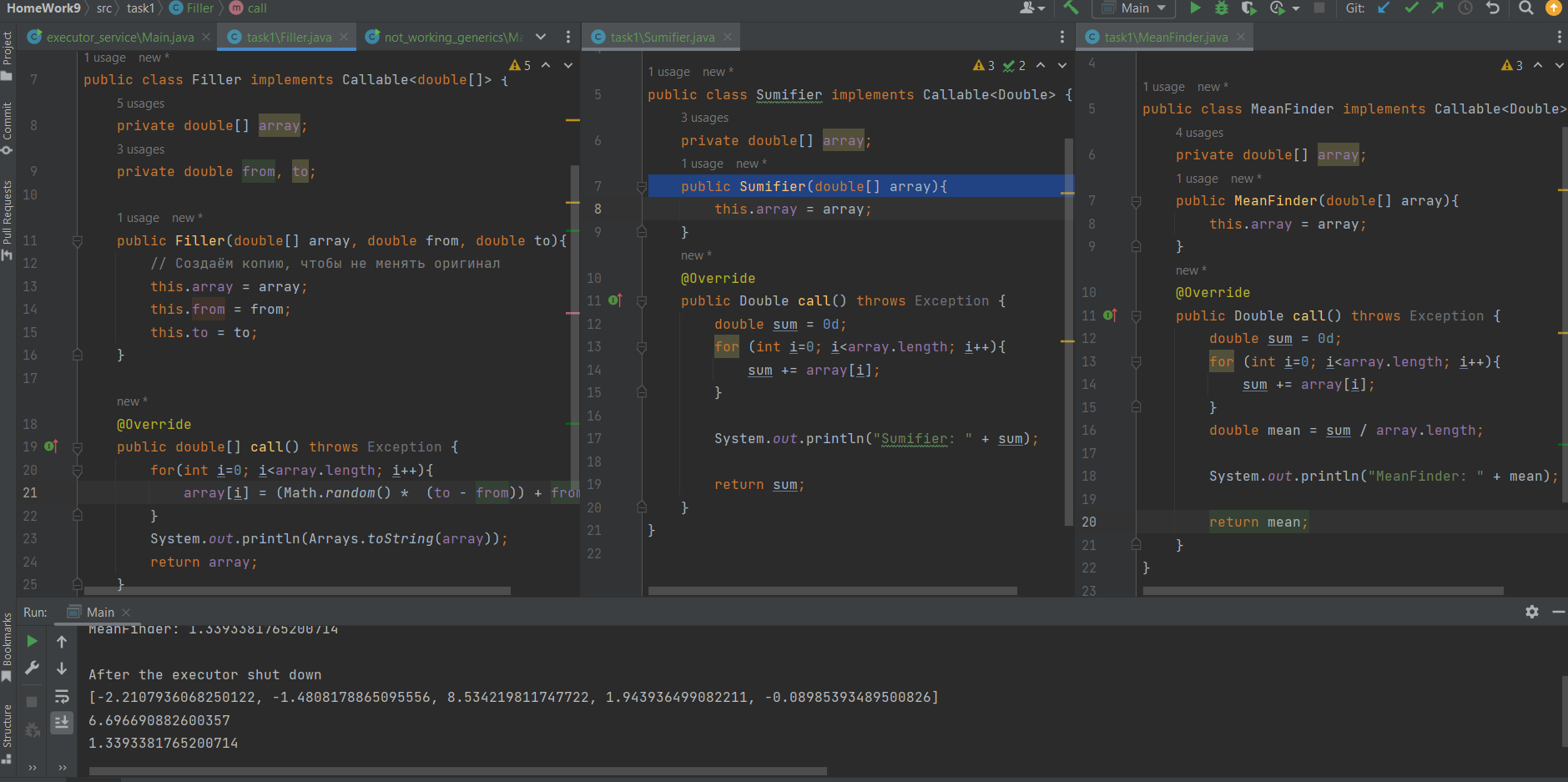
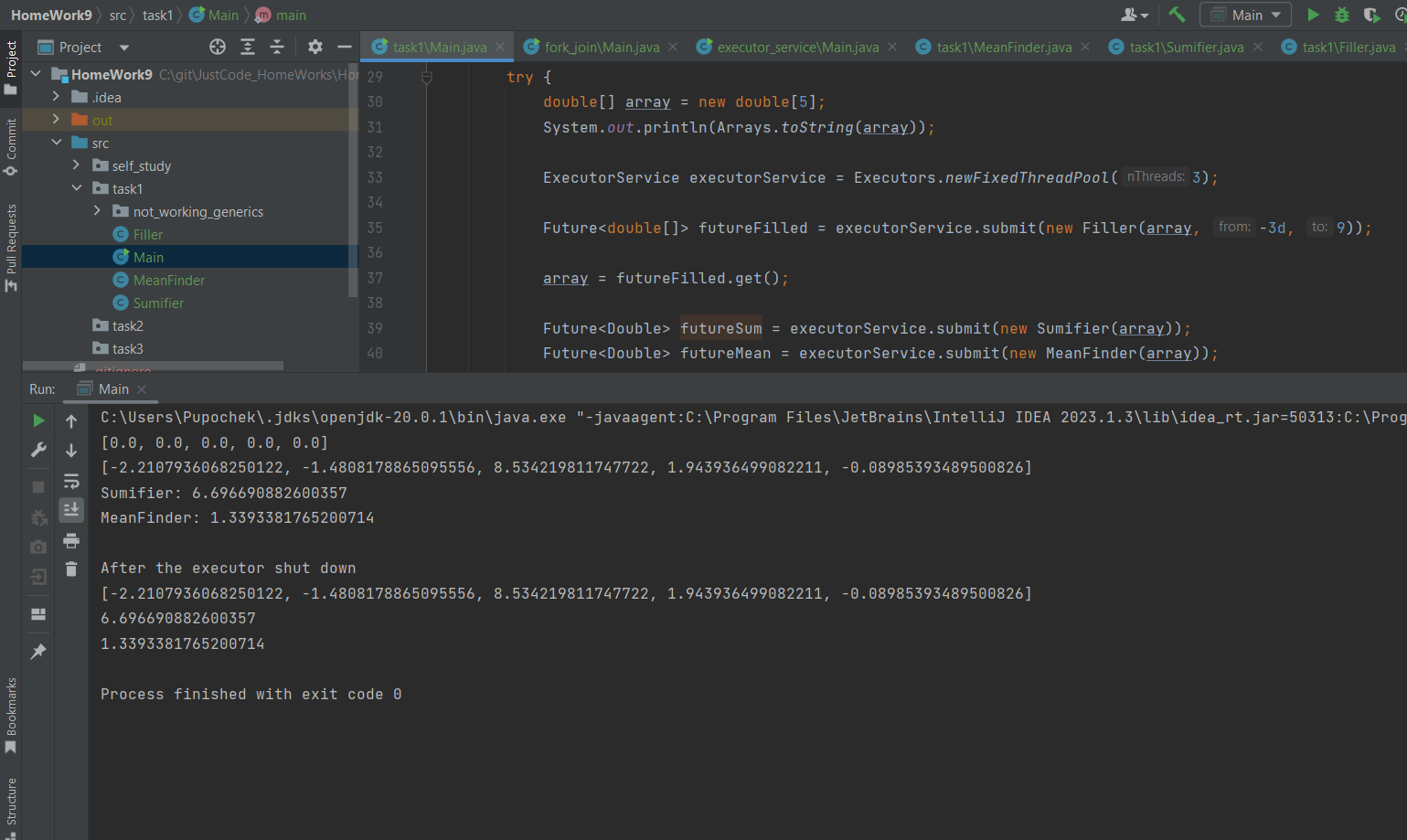
**Домашнее задание 9  
Акильбеков Алар  
GitHub:** <https://github.com/Alar-q/JustCode_HomeWorks>

**Задание 1**

При старте приложения запускаются три потока. Первый поток заполняет массив случайными числами. Два других потока ожидают заполнения. Когда массив заполнен оба потока запускаются. Первый поток находит сумму элементов массива, второй поток среднеарифметическое значение в массиве. Полученный массив, сумма и среднеарифметическое возвращаются в метод main, где должны быть отображены.

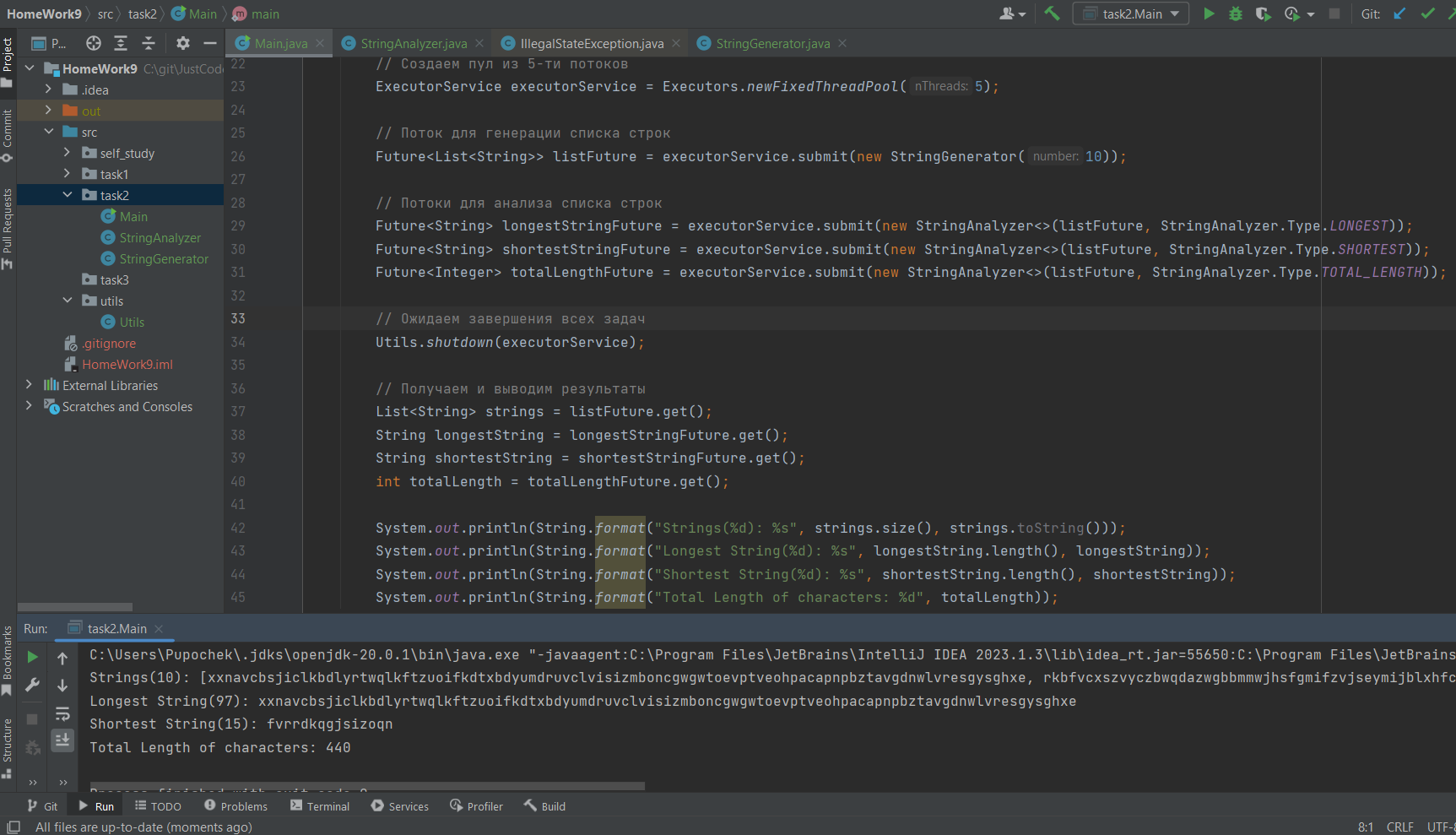
Заданиея выполнил с помощью **ExecutorService**. Я создал 3 класса, реализовывающих **Callable** интерфейс: **Filler**, **Sumifier** и **MeanFinder**. 

Создал фиксированный пул из трех потоков. Интересный момент блокирующей функции **get**, то есть главный поток ожидает его и только после этого запускаются остальные два. Потоки работают с самим массивом, поэтому в случае его изменения, нужно дописывать синхронизацию.  


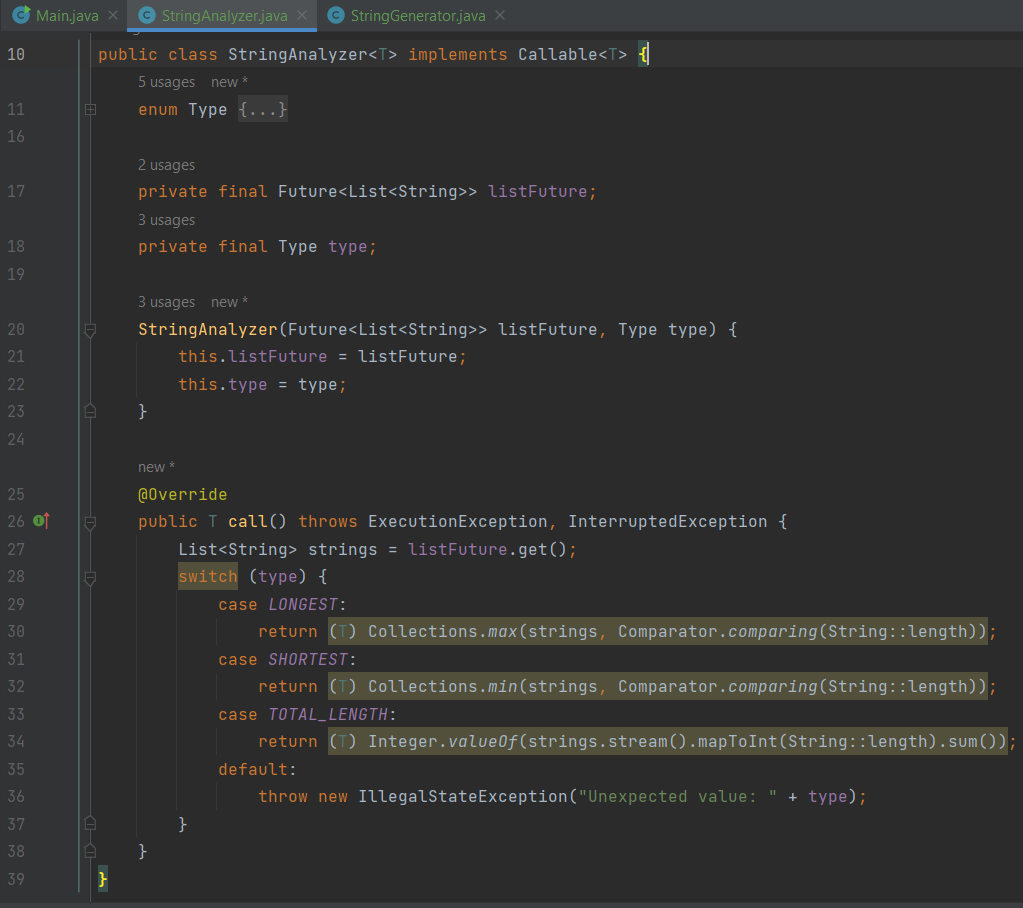
Дополнительно, я хотел написать обобщенные классы Filler, Sumifier и MeanFinder, но не получается приводить значения при суммировании (например, (T) Number), поэтому основная реализация основана на типе Double.

**Задание 2**

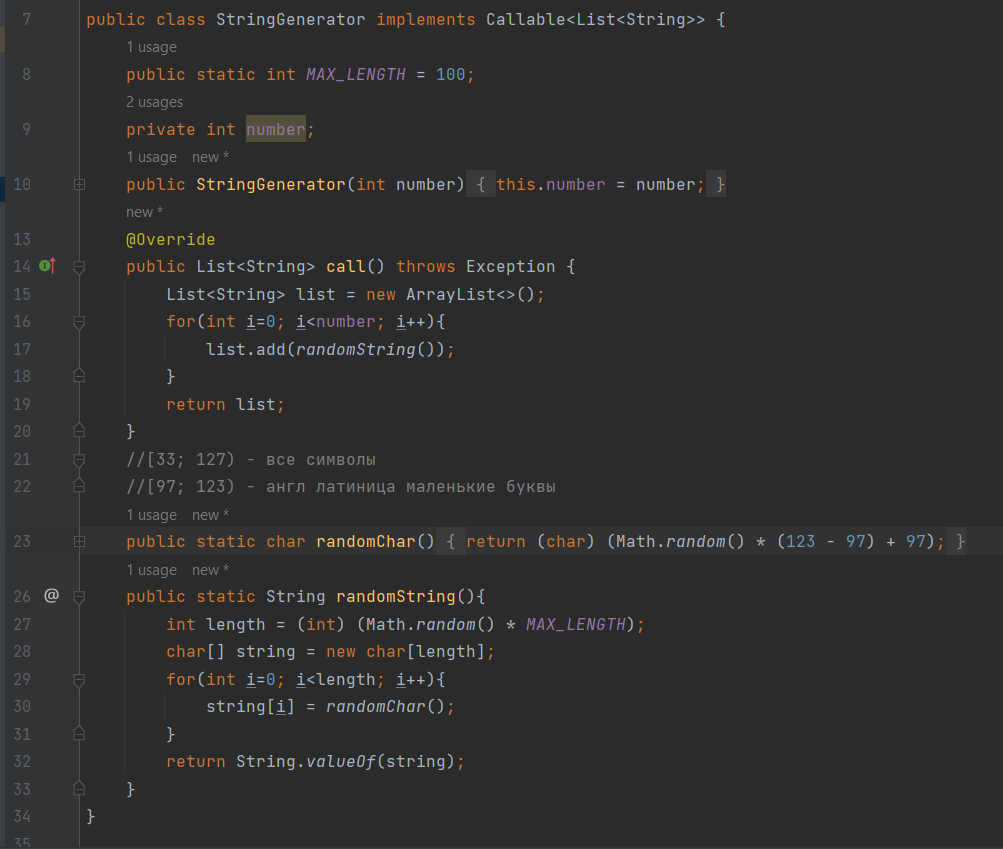
При инициализации программы создаются **пять(?)** потоков. Первый поток генерирует список из случайных строк. **Три других(?)** потока ожидают, пока список строк будет полностью сгенерирован. Как только список создан, все три потока запускаются. Первый поток находит строку с наибольшей длиной, второй поток находит строку с наименьшей длиной, а третий поток считает общее количество символов во всех строках. Полученный список строк, самая длинная строка, самая короткая строка и общее количество символов возвращаются в метод main, где должны быть выведены на экран.

Главный метод запуска программы. Создаем пул из пяти потоков, **запускаем четыре потока, три из которых работают с аргументом типа Future**, то есть они ожидают результата выполнения генератора строк. 

StringAnalyzer обобщен и меняет свои функции в зависимости от переданного аргумента (inner-enum)



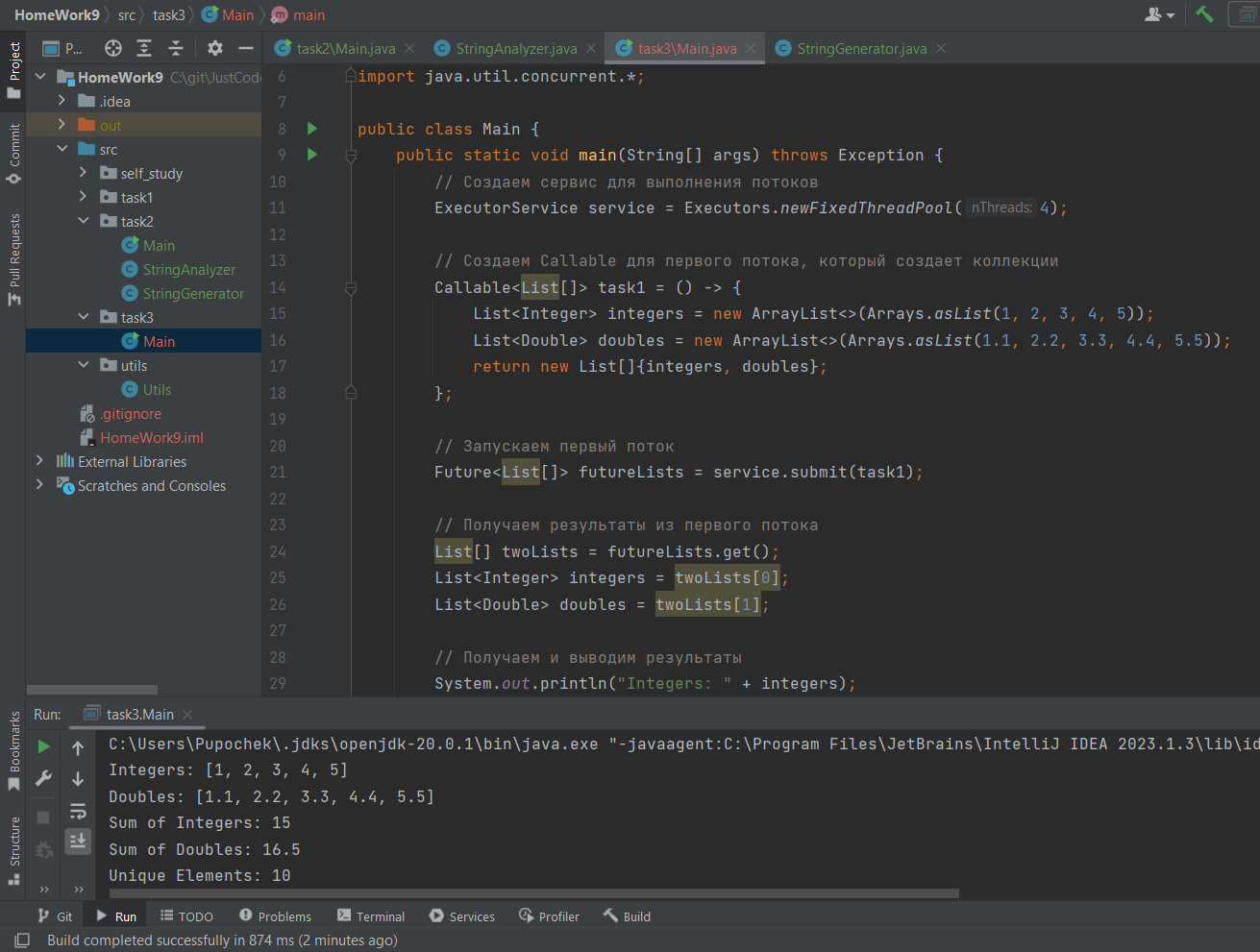
StringGenerator в примере выдает лист из 100 случайных строк состоящих из букв в нижнем регистре

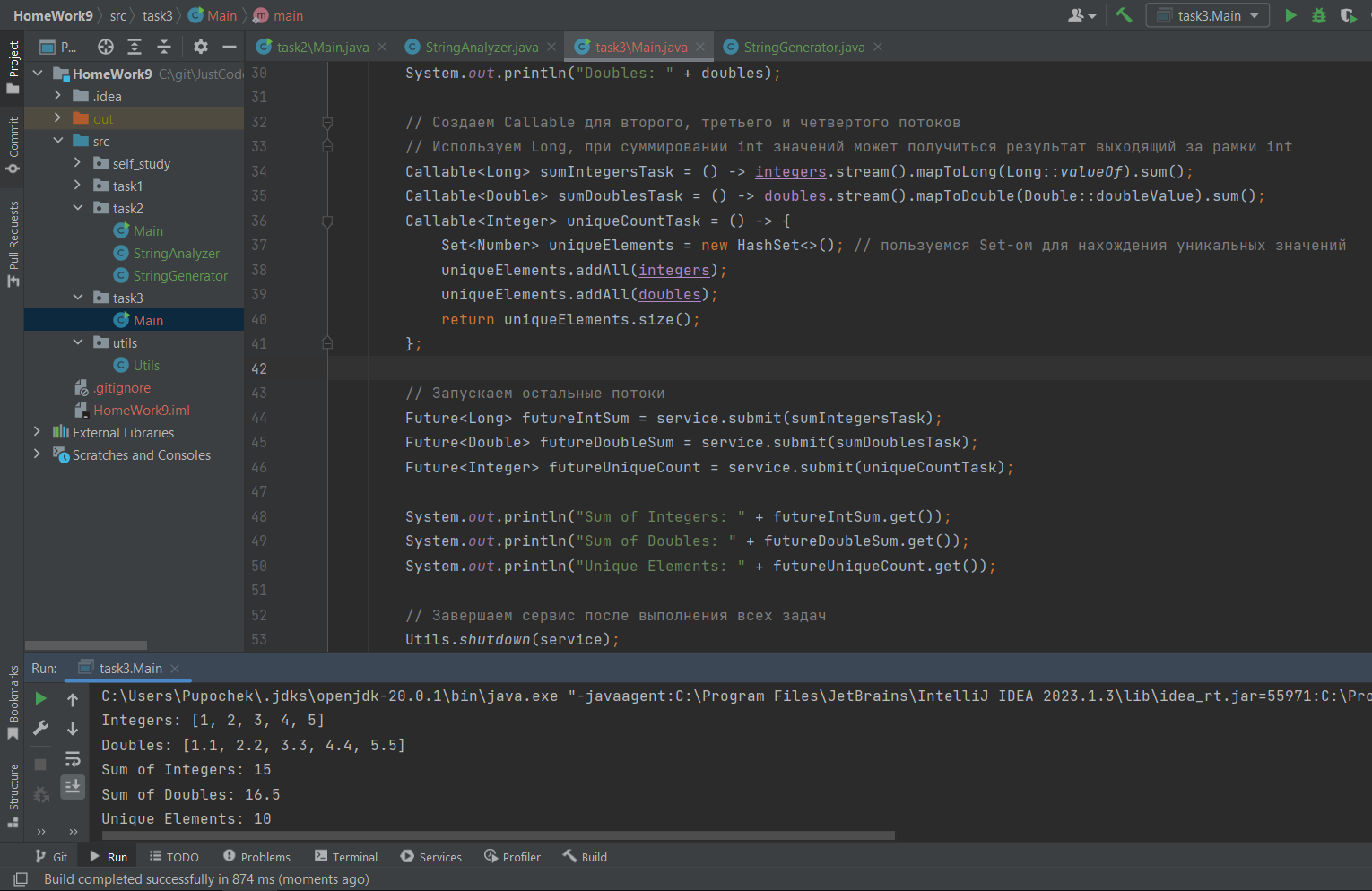


**Задание 3**

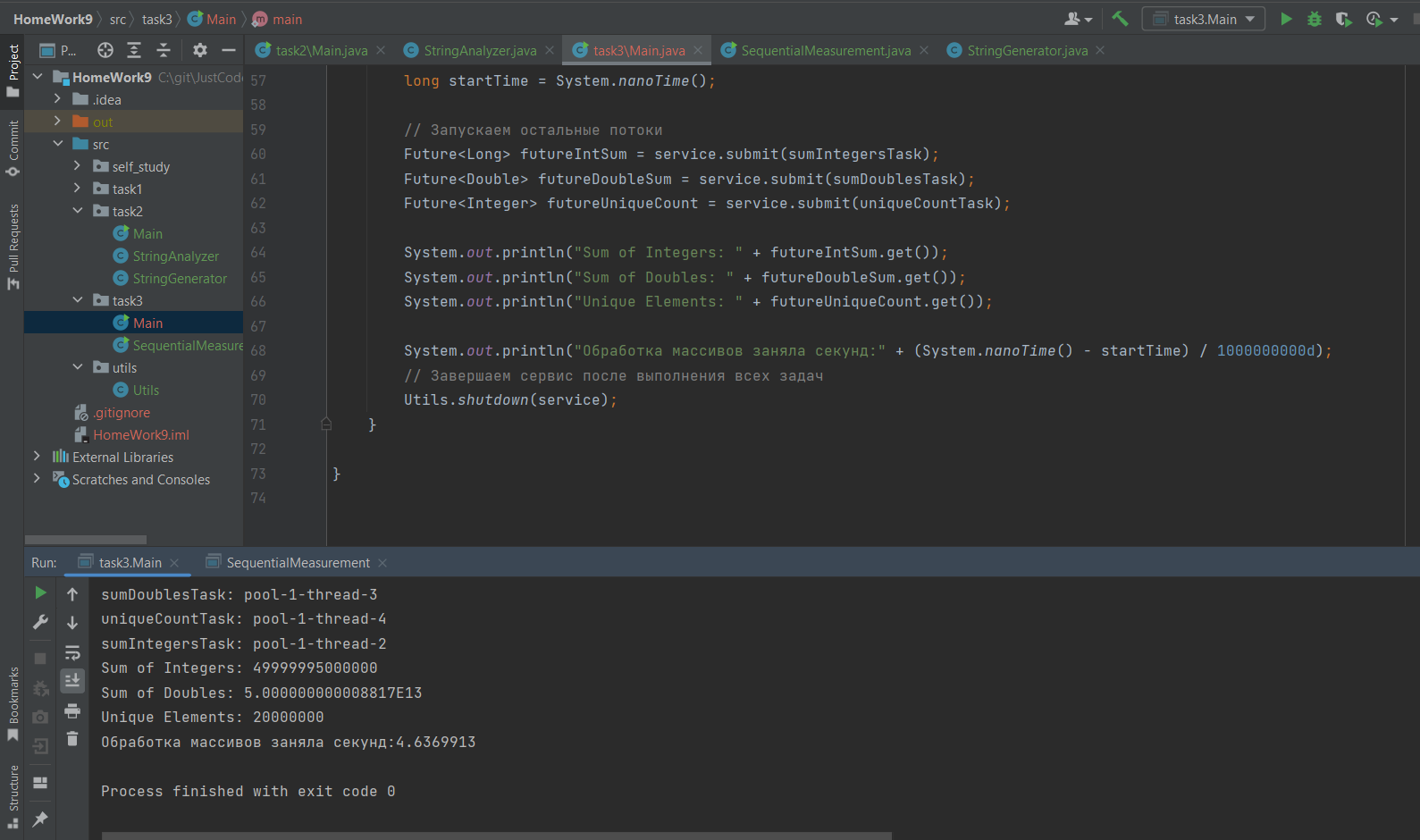
При старте приложения запускаются четыре потока. Первый поток создает две коллекции: одну с целыми числами и одну с вещественными числами. Три других потока ожидают завершения первого потока. Когда коллекции созданы, остальные три потока запускаются. Второй поток находит сумму всех целых чисел, третий поток находит сумму всех вещественных чисел, а четвертый поток находит количество уникальных элементов в обеих коллекциях. Полученные коллекции, суммы и количество уникальных элементов возвращаются в метод main, где должны быть отображены.

Решение для задания я написал, используя лямбда-выражения для реализации Callable классов.

Простой пример, 2 массива по 5 элементов.   




Так же я попытался измерить и **сравнить время выполнение** программы с использованием много поточности и без нее. Должно получиться что линейное выполнение программы работает в 3 раза медленнее, чем параллельное. Экспериментально получилось, что и та и та программа с массивами в 10млн элементов работает около 4 секунд. На скриншотах линейное даже работает чуть быстрее. Я измеряю только время выполнение анализа на заполненных массивах. Также я вывожу имена потоков, так что параллельная реализация точно работает параллельно.

Параллельное (4.6369913с):  
  
Линейное (4.2583039с):  
