Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

**Комп’ютерний практикум № 4**

з дисципліни «Технології паралельних обчислень»

**Тема:** «Розробка паралельних програм з використанням пулів потоків, екзекьюторів та ForkJoinFramework»

|  |  |
| --- | --- |
| **Виконав:**  студент групи ІТ-04  Чапча Святослав  Дата здачі \_\_\_\_\_\_\_\_  Захищено з балом \_\_\_\_\_\_\_ | **Перевірила:**  ас. кафедри ІПІ  Дифучина О.Ю. |

Київ 2023

**Завдання:**

1. Побудуйте алгоритм статистичного аналізу тексту та визначте характеристики випадкової величини «довжина слова в символах» з використанням ForkJoinFramework.**20 балів.**Дослідіть побудований алгоритм аналізу текстових документів на ефективність експериментально. **10 балів.**

Скріншоти програми:

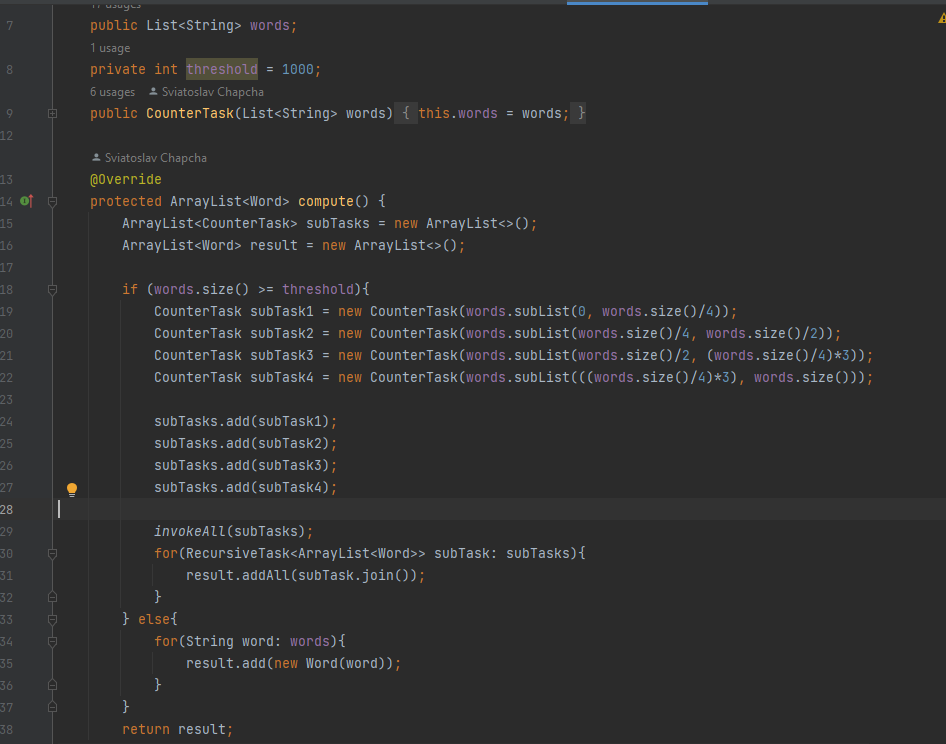
****

Рис. 1. CounterTask.java

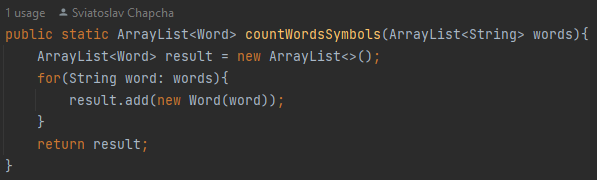


Рис. 2. Метод countWordsSymbols()

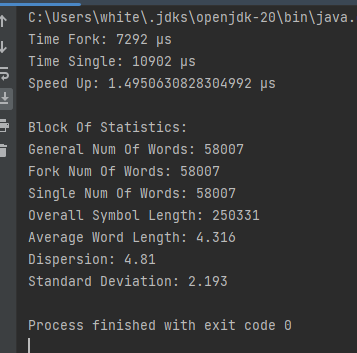


Рис. 3. Результат виконання програми

Для виконання цього завдання ми використаємо абстрактний клас RecursiveTask, щоб мати змогу повертати список слів, а також поставимо трешхолд. Якщо розмір слів буде більшим за нього, то ми поділимо роботу на 4 таски та запустимо їх у цей же метод.

Як бачимо за результатом виконання, прискорення ForkJoin ~ у 1.495 разів. Оскільки виконання є доволі програми є доволі швидким, то ми зафіксували швидкість програми у мікросекундах. Також ми визначили кілька характеристик випадкової величини, а саме дисперсію, середня довжина слова та стандартне відхилення.

1. Реалізуйте один з алгоритмів комп’ютерного практикуму 2 або 3 з використанням ForkJoinFramework та визначте прискорення, яке отримане за рахунок використання ForkJoinFramework.**20 балів.**

Скріншоти програми:

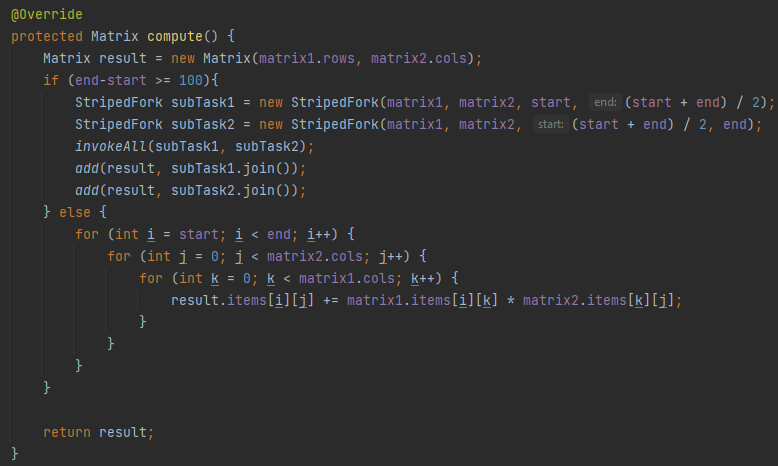
****

Рис. 4. StripedFork.java



Рис. 5. Striped.java

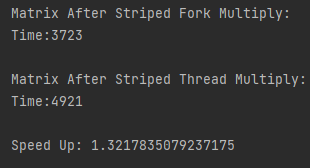
****

Рис. 6. Результат виконання програми

Для виконання цього завдання я обрав стрічковий алгоритм з другої практичної роботи. Суть полягає у тому, що якщо рядок стрічкового алгоритму довший за 100 елементів, то задача ділиться на дві підзадачі. І завдяки можливостям абстрактного класу RecursiveTask, ми повертаємо результуючу матрицю.

За результатами замірів, можемо побачити, що прискорення також є, у 1.32 рази.

1. Розробіть та реалізуйте алгоритм пошуку спільних слів в текстових документах з використанням ForkJoinFramework. **20 балів.**

Скріншоти програми:

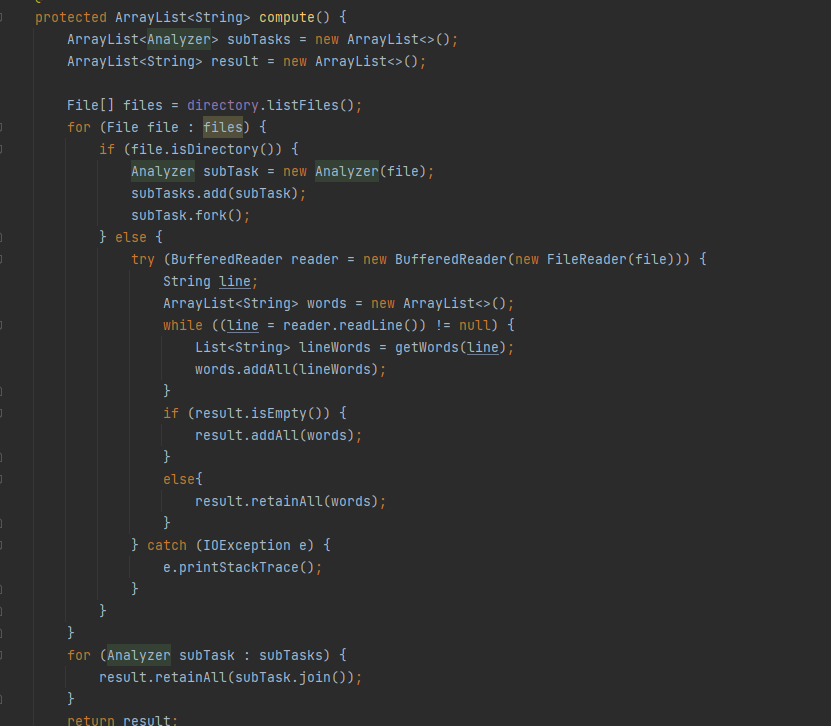
****

Рис. 7. Analyzer.java

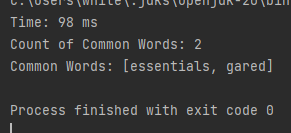
****

Рис. 8. Результат виконання програми

Для виконання цього завдання, ми використовуємо ForkJoin та RecursiveTask, щоб ділити задачу на підзадачі при наявності у директорії інших директорій. Ми перевіряємо, чи є файл – директорією і запускаємо підзадачу, якщо результат позитивний. Потім через метод retainAll, ми зберігаємо спільні слова і збираємо результати через метод join().

1. Розробіть та реалізуйте алгоритм пошуку текстових документів, які відповідають заданим ключовим словам (належать до області «Інформаційні технології»), з використанням ForkJoinFramework. **30 балів.**

Скріншоти програми:



Рис. 9. Analyzer.java

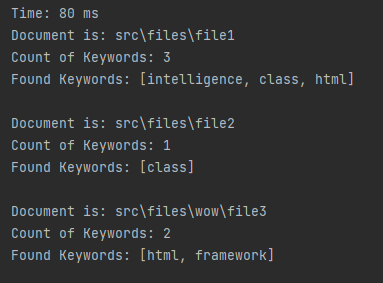


Рис. 10. Результат виконання програми

Завдання виконано практично за тією ж самою схемою, що і третє, і тут також використовується метод retainAll().

Програмний код до всіх завдань наведений в репозиторії:

<https://github.com/whitetark/multithreading/tree/main/lab04>

**Висновок**Під час виконання практичної роботи під номером 4, я дізнався про пул потоків ForkJoin, можливістю розділяти задачу на підзадачі. Легко зрозуміти, що такий спосіб гарно використовувати, коли підзадачі незалежні один від одного. Ставлячи обмеження у вигляді трешхолду, ми можемо розбити одну затратну задачу на декілька незатратних підзадач, що будуть йти паралельно. Також на результати виконання впливають кількість процесорів, доступних для використання в поточній системі.