

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТУ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Інститут комп’ютерних наук та інформаційних технологій
Кафедра систем штучного інтелекту

Лабораторна робота №6

з дисципліни

«Об’єктно-орієнтоване програмування»

Виконала:

студентка групи КН-108

Семич Тамара

Прийняла:

Грабовська Н.Р.

Львів – 2019 р.

Зміст

1. Тема лабораторної роботи.
2. Мета роботи.
3. Вимоги.
4. Висновок.

Тема : Паралельне виконання. Багатопоточність. Ефективність використання.

Мета:

- Ознайомлення з моделлю потоків Java.
- Організація паралельного виконання декількох частин програми.
- Вимірювання часу паралельних та послідовних обчислень.
- Демонстрація ефективності паралельної обробки.

Вимоги:

1. Використовуючи програми рішень попередніх задач, продемонструвати можливість паралельної обробки елементів контейнера: створити не менше трьох додаткових потоків, на яких викликати відповідні методи обробки контейнера.

2. Забезпечити можливість встановлення користувачем максимального часу виконання (таймаута) при закінченні якого обробка повинна припинятися незалежно від того знайдений кінцевий результат чи ні.

3. Для паралельної обробки використовувати алгоритми, що не змінюють початкову колекцію.

4. Кількість елементів контейнера повинна бути досить велика, складність алгоритмів обробки колекції повинна бути зіставна, а час виконання приблизно однаковий, наприклад:

- пошук мінімуму або максимуму;
- обчислення середнього значення або суми;
- підрахунок елементів, що задовольняють деякій умові;
- відбір за заданим критерієм;
- власний варіант, що відповідає обраній прикладної області.

5. Забезпечити вимірювання часу паралельної обробки елементів контейнера за допомогою розроблених раніше методів.

6. Додати до алгоритмів штучну затримку виконання для кожної ітерації циклів поелементної обробки контейнерів, щоб загальний час обробки був декілька секунд.

7. Реалізувати послідовну обробку контейнера за допомогою методів, що використовувались для паралельної обробки та забезпечити вимірювання часу їх роботи.

8. Порівняти час паралельної і послідовної обробки та зробити висновки про ефективність розпаралелювання:

- результати вимірювання часу звести в таблицю;
- обчислити та продемонструвати у скільки разів паралельне виконання швидше послідовного.

Розробник: Семич Тамара, КН-108, номер варіанту індивідуального завдання- 4.

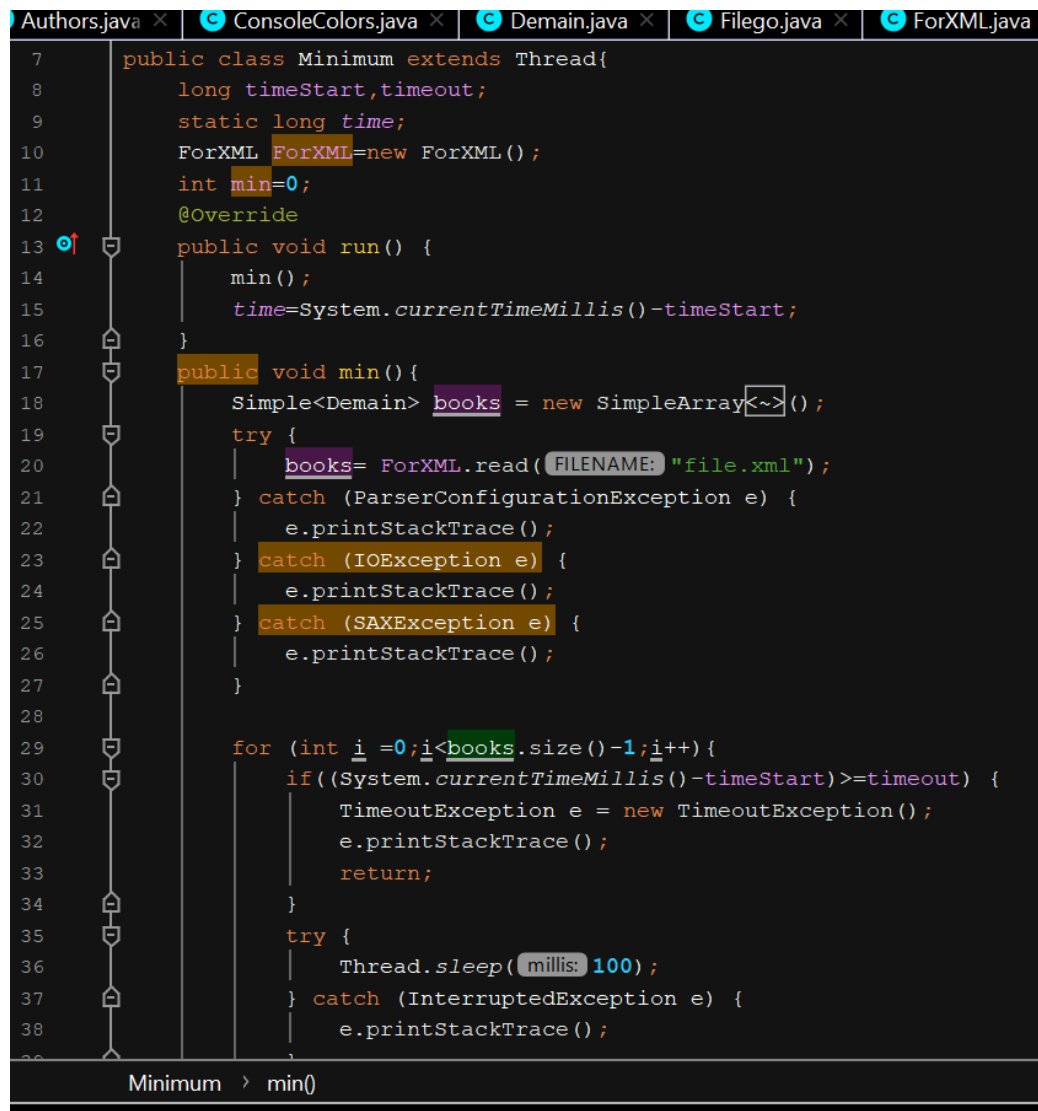
Завдання: Бібліотека. Дані про книгу: ISBN ; назва; автори (кількість не обмежена); видавництво; жанр; дата видання.

Ієрархія та структура об'єктів:

1. Клас Main, який містить функцію main, doexample.
2. Клас Filego, який містить функції, doFile та make_info.
3. Клас Demain, який містить поля numberISBN, title, genre, edition, date та їхні гетери / сетери.
4. Клас Authors, який містить поля author, surname.
5. Клас FoXML, який містить функції WriteParamXML, read.
6. Клас ConsoleColors,
7. Клас SimpleArray – контейнер.
8. Інтерфейс Simple
9. Клас ArrayIterator, який містить функції hasNext, next.
10. Клас Search, який містить функції search, run.
11. Клас Max, який містить функції max, run.
12. Клас Minimum, який містить функції minimum, run

Важливі фрагменти коду:

Клас Minimum:



```
7 public class Minimum extends Thread{
8     long timeStart,timeout;
9     static long time;
10    ForXML ForXML=new ForXML();
11    int min=0;
12    @Override
13    public void run() {
14        min();
15        time=System.currentTimeMillis()-timeStart;
16    }
17    public void min() {
18        Simple<Demain> books = new SimpleArray<>();
19        try {
20            books= ForXML.read( FILENAME: "file.xml");
21        } catch (ParserConfigurationException e) {
22            e.printStackTrace();
23        } catch (IOException e) {
24            e.printStackTrace();
25        } catch (SAXException e) {
26            e.printStackTrace();
27        }
28
29        for (int i =0;i<books.size()-1;i++){
30            if((System.currentTimeMillis()-timeStart)>=timeout) {
31                TimeoutException e = new TimeoutException();
32                e.printStackTrace();
33                return;
34            }
35            try {
36                Thread.sleep( millis: 100);
37            } catch (InterruptedException e) {
38                e.printStackTrace();
39            }
40        }
41    }
42 }
```

Minimum > min()

Висновок: на цій лабораторній роботі я дізналася про багатопоточність, для чого її використовувати і як це правильно робити. Переконалася що паралельна обробка даних швидша за послідовну .