НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра «ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ ТА ПРОГРАМУВАННЯ»

«Об’єктно-орієнтоване програмування»

Звіт з лабораторної роботи №14
Тема: «Паралельне виконання. Багатопоточність»

Виконав:
ст. гр. 1.КІТ102.8а
Міщенко Д.С.

Перевірив:
Пугачов Р.В.

Харків – 2019

Лабораторна робота №14
Паралельне виконання. Багатопоточність

Мета

- Вимірювання часу паралельних та послідовних обчислень.
- Демонстрація ефективності паралельної обробки.

1 ВИМОГИ

1.1 Розробник

1. Міщенко Дмитро Сергійович
2. КІТ-118в
3. Варіант 13

1.2 Загальне завдання

Вимоги

1. Забезпечити вимірювання часу паралельної обробки елементів контейнера за допомогою розроблених раніше методів.
2. Додати до алгоритмів штучну затримку виконання для кожної ітерації циклів поелементної обробки контейнерів, щоб загальний час обробки був декілька секунд.
3. Реалізувати послідовну обробку контейнера за допомогою методів, що використовувались для паралельної обробки та забезпечити вимірювання часу їх роботи.
4. 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.

Прикладні задачі

13. Готель. Замовлення номера: паспортні дані; дати поселення та виселення; номер (клас, число місць); причина поселення (кількість не обмежена).

2 ОПИС ПРОГРАМИ

2.1 Засоби ООП

Поліморфізм,інкапсуляція

2.2 Ієрархія та структура даних

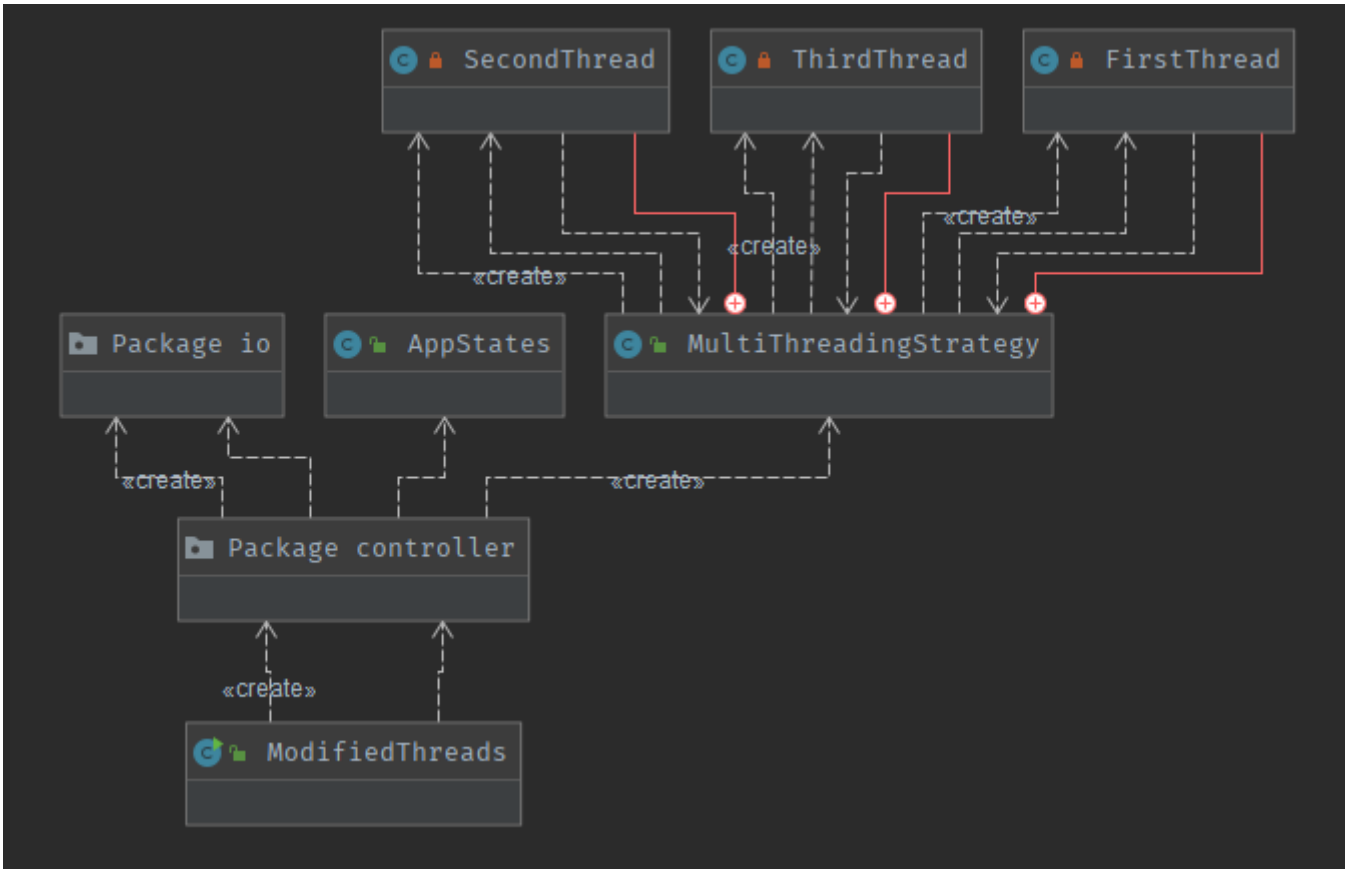

Рисунок 1 – Ієрархія класів

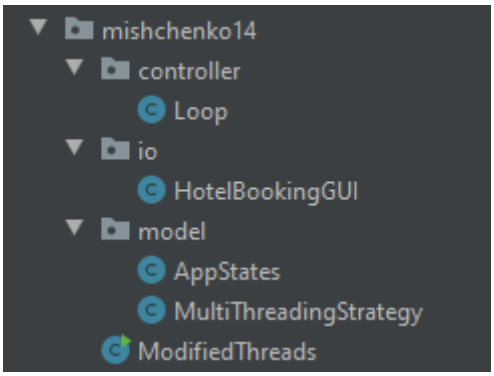

Рисунок 2 – Структура пакету

2.2 Важливі фрагменти програми

```java
public long runConsequentially() {
    long time_start = System.currentTimeMillis();
    logger.info("Starting consequential run ... ");
    try {
        getClientsWithNameCharlesCount();
        getWorkingClientsCount();
        getClientsWithBirthdaysInAprilCount();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    logger.info("Finishing consequential run ... ");
    return System.currentTimeMillis() - time_start;

}
```

Рисунок 3 – Виконання завдань послідовно в одній нитці

```java
public void getClientsWithNameCharlesCount() throws InterruptedException {
    int count = 0;
    for (HotelBooking client : list) {
        if (!Thread.currentThread().isInterrupted()) {
            if (client.getPerson().getName().equals("Charles")) {
                count++;
            }
        } else {
            throw new InterruptedException();
        }
    }
}
```

Рисунок 4 – Приклад завдання

```java
public void compare() {
    long parallel = runParallel();
    long consequential = runConsequentially();
    output.println("Time of parallel run → " + parallel);
    output.println("Time of consequential run → " + consequential);
    output.println("Parallel is faster in " + (double) consequential / parallel + " times");
    GUI.waitForEnter();
    GUI.cls();
}
```

Рисунок 5 – Порівняння паралельної обробки та послідовної

3 ВАРІАНТИ ВИКОРИСТАННЯ

```
17:52:54.178 [main] INFO  u.k.oop.mishchenko14.controller.Loop - App started
01. Add new client
02. Delete client
03. Clear list
04. Show list
05. Save
06. Load
07. Get list as string
08. Get list as array
09. Sort
10. Search by reason
11. Thread demo
12. Parallel vs consequential demo
13. Generate random values
00. Exit
```

Рисунок 6 – Головне меню програми

```
Choose option
13
Enter count of elements to generate
10000000
```

Рисунок 7 – Генерація великої кількості елементів

```
Choose option
12
17:54:20.609 [main] INFO  u.k.o.m.model.MultiThreadingStrategy - Starting parallel run ...
17:54:21.039 [ThirdThread] INFO  u.k.o.m.model.MultiThreadingStrategy - ThirdThread finished
17:54:21.143 [FirstThread] INFO  u.k.o.m.model.MultiThreadingStrategy - FirstThread finished
17:54:30.384 [SecondThread] INFO  u.k.o.m.model.MultiThreadingStrategy - SecondThread finished
17:54:30.384 [main] INFO  u.k.o.m.model.MultiThreadingStrategy - Finishing parallel run ...
17:54:30.384 [main] INFO  u.k.o.m.model.MultiThreadingStrategy - Starting consequential run ...
17:54:40.420 [main] INFO  u.k.o.m.model.MultiThreadingStrategy - Finishing consequential run ...
Time of parallel run → 9775
Time of consequential run → 10037
Parallel is faster in 1.0268030690537084 times
```

Рисунок 8 – Порівняння паралельного та послідовного виконання

ВИСНОВКИ

В результаті виконання лабораторної роботи була продемонстрована ефективність паралельної обробки. Було виявлено, що через імітовану багатопоточність, швидкість роботи зростає лише на дуже великій кількості даних.