НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра «ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ ТА ПРОГРАМУВАННЯ»

«Об’єктно-орієнтоване програмування»

Звіт з лабораторної роботи №13
Тема: «Паралельне виконання. Багатопоточність»

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

Перевірив:
Пугачов Р.В.

Харків – 2019

Лабораторна робота №13
Паралельне виконання. Багатопоточність

Мета

- Ознайомлення з моделлю потоків Java.
- Організація паралельного виконання декількох частин програми.

1 ВИМОГИ

1.1 Розробник

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

1.2 Загальне завдання

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.

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

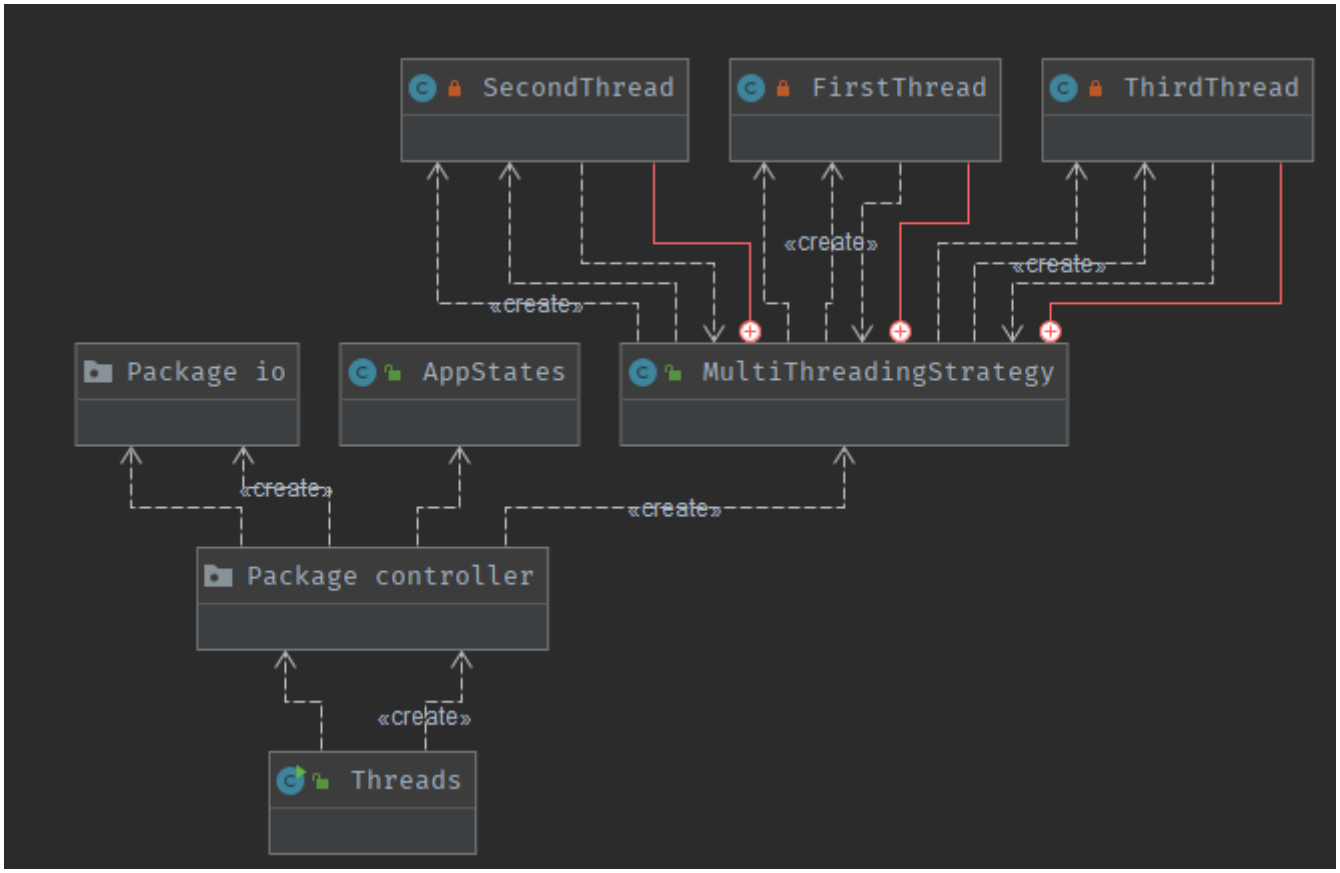

Рисунок 1 – Ієрархія классів

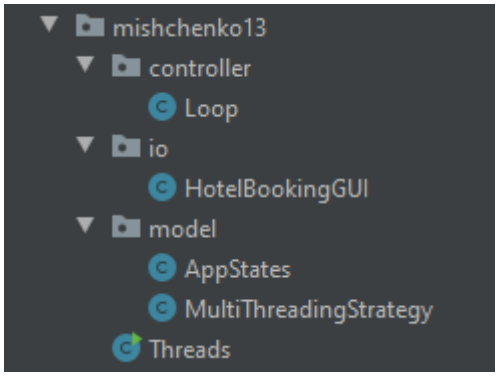

Рисунок 2 – Структура пакету

2.2 Важливі фрагменти програми

```
private class FirstThread implements Runnable {
    public void run() {
        int count = 0;
        logger.info("First Thread started");
        try {
            for (HotelBooking client : list) {
                if (!Thread.currentThread().isInterrupted()) {
                    if (client.getPerson().getName().equals("Charles")) {
                        count++;
                    }
                } else {
                    throw new InterruptedException();
                }
            }
            logger.info("First Thread finished");
            output.println("People with name Charles : " + count);
        } catch (InterruptedException e) {
            logger.info("First Thread is interrupted");
        }
    }
}
```

Рисунок 3 – Одна з реалізованих ниток

```
public void start() {
    output.println("Set the timer [0 - 100 000 ms]: ");
    int timer_num = HotelBookingInput.inInt(scanner);
    output.println("Starting all threads ... ");
    FirstThread first = new FirstThread();
    Thread t1 = new Thread(first, name: "FirstThread");
    SecondThread second = new SecondThread();
    Thread t2 = new Thread(second, name: "SecondThread");
    ThirdThread third = new ThirdThread();
    Thread t3 = new Thread(third, name: "ThirdThread");
    t1.start();
    t2.start();
    t3.start();

    Timer timer = new Timer(timer_num, event -> {
        logger.info("Interrupting thread ... ");
        t1.interrupt();
        t2.interrupt();
        t3.interrupt();
    });

    timer.setRepeats(false);
    timer.start();
    try {
        t1.join();
        t2.join();
        t3.join();
        timer.stop();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    logger.info("Finishing all threads ... ");
    GUI.waitForEnter();
    GUI.cls();
}
```

Рисунок 4 – Метод, що контролює нитки

3 ВАРІАНТИ ВИКОРИСТАННЯ

```
17:51:23.580 [main] INFO  u.k.oop.mishchenko13.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. Generate random values
00. Exit
Choose option
```

Рисунок 5 – Головне меню програми

```
Choose option
11
Set the timer [0 - 100 000 ms]:
100
Starting all threads ...
17:25:56.263 [FirstThread] INFO  u.k.o.m.model.MultiThreadingStrategy - First Thread started
17:25:56.264 [SecondThread] INFO  u.k.o.m.model.MultiThreadingStrategy - Second Thread started
17:25:56.264 [ThirdThread] INFO  u.k.o.m.model.MultiThreadingStrategy - Third Thread started
17:25:56.264 [FirstThread] INFO  u.k.o.m.model.MultiThreadingStrategy - First Thread finished
17:25:56.265 [ThirdThread] INFO  u.k.o.m.model.MultiThreadingStrategy - Third Thread finished
People with name Charles : 0
People with birthday in April: 1
17:25:56.278 [SecondThread] INFO  u.k.o.m.model.MultiThreadingStrategy - Second Thread finished
People in business trip: 5
17:25:56.301 [main] INFO  u.k.o.m.model.MultiThreadingStrategy - Finishing all threads ...
Enter any key to continue ...
```

Рисунок 6 – Виконання ниток

ВИСНОВКИ

В результаті виконання лабораторної роботи було організоване паралельне виконання декількох частин програми за допомогою моделі багатопоточності в Java.