13. Паралельне виконання. Багатопоточність

Мета: Ознайомлення з моделлю потоків Java.

Організація паралельного виконання декількох частин програми.

1 ВИМОГИ

1.1 Розробник

Інформація про розробника:

- Кабак Олександр Русланович
- НТУ “ХПІ” 1.КІТ102.8а
- Варіант 5

1.2 Загальне завдання

- Використовуючи програми рішень попередніх задач, продемонструвати можливість паралельної обробки елементів контейнера: створити не менше трьох додаткових потоків, на яких викликати відповідні методи обробки контейнера.
- Забезпечити можливість встановлення користувачем максимального часу виконання (таймаута) при закінченні якого обробка повинна припинятися незалежно від того знайдений кінцевий результат чи ні.
- Для паралельної обробки використовувати алгоритми, що не змінюють початкову колекцію.
- 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.

1.3 Задача

5. Прикладна галузь: Довідник покупця. Торговельна точка: назва; адреса; телефони (кількість не обмежена); спеціалізація; час роботи (з зазначенням днів тижня).

2 ОПИС ПРОГРАМИ

2.1 Засоби ООП

У даній програмі присутні:

1) Двозв'язний список, що параметризується;
2) Збереження та відновлення об'єктів без протоколу серіалізації та з ним;
3) Спілкування з користувачем за допомогою меню та автоматичний режим для перегляду;
4) Регулярні вирази для коректного запису даних в базу;
5) Пошук елементів за допомогою регулярних виразів;
6) Багатопоточність та паралельне виконання завданнь.

2.2 Важливі фрагменти програми

```
static class Thread1 implements Runnable {
    public void run() {
        int count = 0;
        System.out.println("Первий поток запущен");
        try {
            for (Store elem : Menu.stores) {
                if (!Thread.currentThread().isInterrupted()) {
                    for (int i = 0; i < elem.getTelephone().size(); i++) {
                        if (elem.getTelephone().get(i).toString().matches( regex: "(\\+*)\\d{6}")) {
                            count++;
                        }
                    }
                } else {
                    throw new InterruptedException();
                }
            }
            System.out.println("6-значних номеров: " + count);
        } catch (InterruptedException e) {
            System.err.println(e.getMessage());
        }
    }
}
```

Рис.1 - Клас Thread1, як приклад створеної нитки(потоку)

```java
public static void startThreads() {
    int timer = 100000;
    System.out.println("Задан таймер ["+(timer/1000)+"]: ");

    System.out.println("Запуск потоков...");

    Thread1 first = new Thread1();
    Thread t1 = new Thread(first,  name: "Первий поток");

    Thread2 second = new Thread2();
    Thread t2 = new Thread(second,  name: "Второй поток");

    Thread3 third = new Thread3();
    Thread t3 = new Thread(third,  name: "Третий поток");

    t1.start();
    t2.start();
    t3.start();
    Timer timerTmp = new Timer(timer, new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent event) {
            System.out.println("Потоки прерваны...");
            t1.interrupt();
            t2.interrupt();
            t3.interrupt();
        }
    });
    timerTmp.setRepeats(false);
    timerTmp.start();
    try {
        t1.join();
        t2.join();
        t3.join();
        timerTmp.stop();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    System.out.println("Завершены все потоки...");
}
```

Рис. 2 – Метод для запуску потоків

3 ВАРІАНТИ ВИКОРИСТАННЯ

Програма дозволяє створювати об’єкт – список магазинів, що заносяться у запис каталогу. Користувач може додавати інші магазини до списку, видаляти

елементи вибірково, а також очистити весь масив одним викликом відповідної кнопки меню, переглядати магазини за номерами телефону(короткими або українські). Також присутня можливість серіалізувати /десеріалізувати об’єкти з файлу, використовувати багатопоточність для вирішення певних задач.

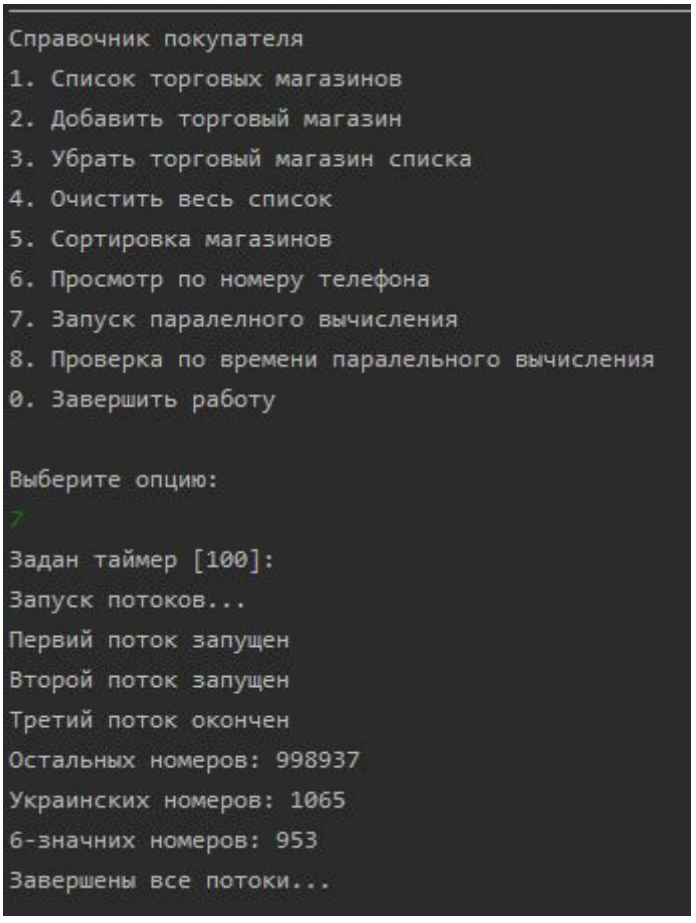

Рис. 3 – Результати пошуку коротких, українських та інших номерів у списку (задіяно 1 мільйон магазинів, приблизний час виконання 1 секунда)

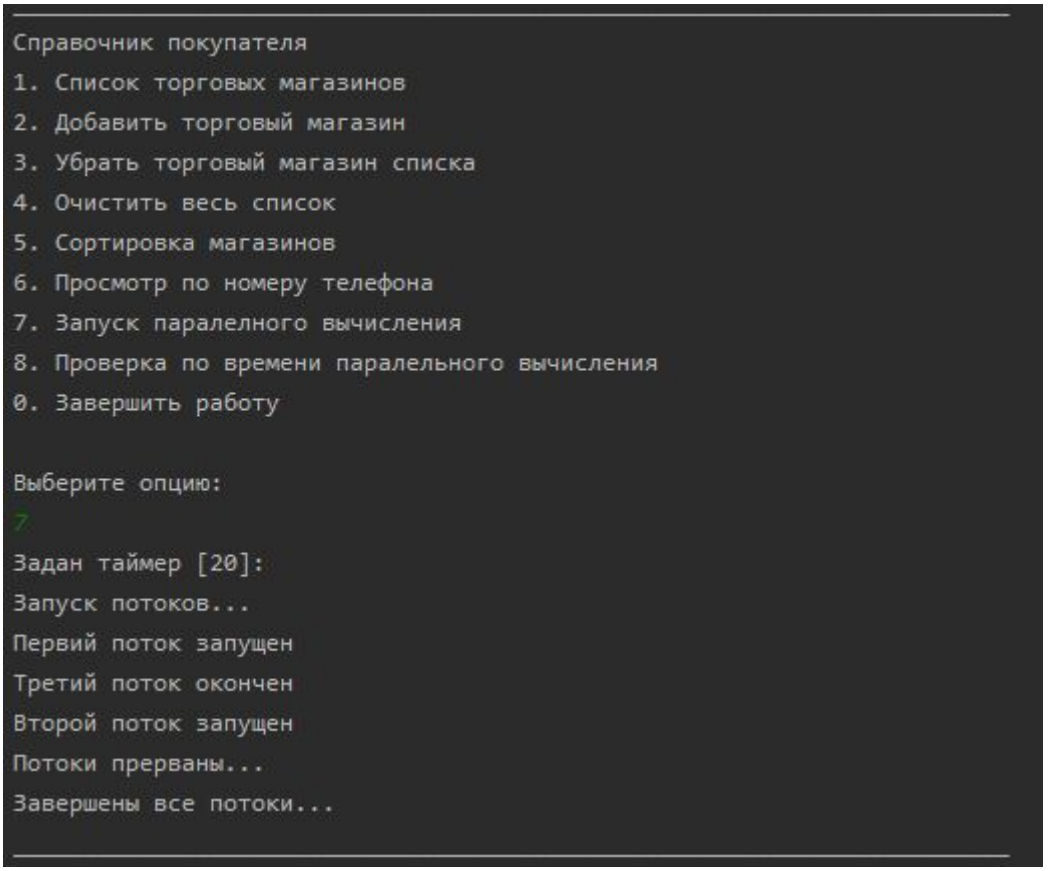

Рис. 4 - Результати пошуку коротких, українських та інших номерів у списку (задіяно 10 мільйон магазинів, приблизний час виконання 1 секунда)

Попри те, що к-сть була збільшена у 10 разів, процес підрахунку зайняв кілька хвилин, тому встановивши таймер на 20 секунд програма закінчила виконання.

Висновки: в даній лабораторній роботі створена та виконана паралельна обробка елементів контейнера за допомогою класу Thread та створеного ThreadHelper для виконання умов задачі.