

13. Паралельне виконання. Багатопоточність

Мета: Ознайомлення з моделлю потоків *Java*.

Організація паралельного виконання декількох частин програми.

1 ВИМОГИ

1.1 Розробник

Інформація про розробника:

- Кулик Данііл Ігорович
- НТУ “ХПІ” КІТ-118в
- Варіант 11

1.2 Загальне завдання

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 (тайм-аута) при закінченні якої обробка повинна припинитися 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.

2 ОПИС ПРОГРАМИ

2.1 Засоби ООП

Були розроблені додаткові функції для класу-контейнера. Розроблене діалогове меню та можливість зчитування даних з файлу.

2.2 Ієрархія та структура даних

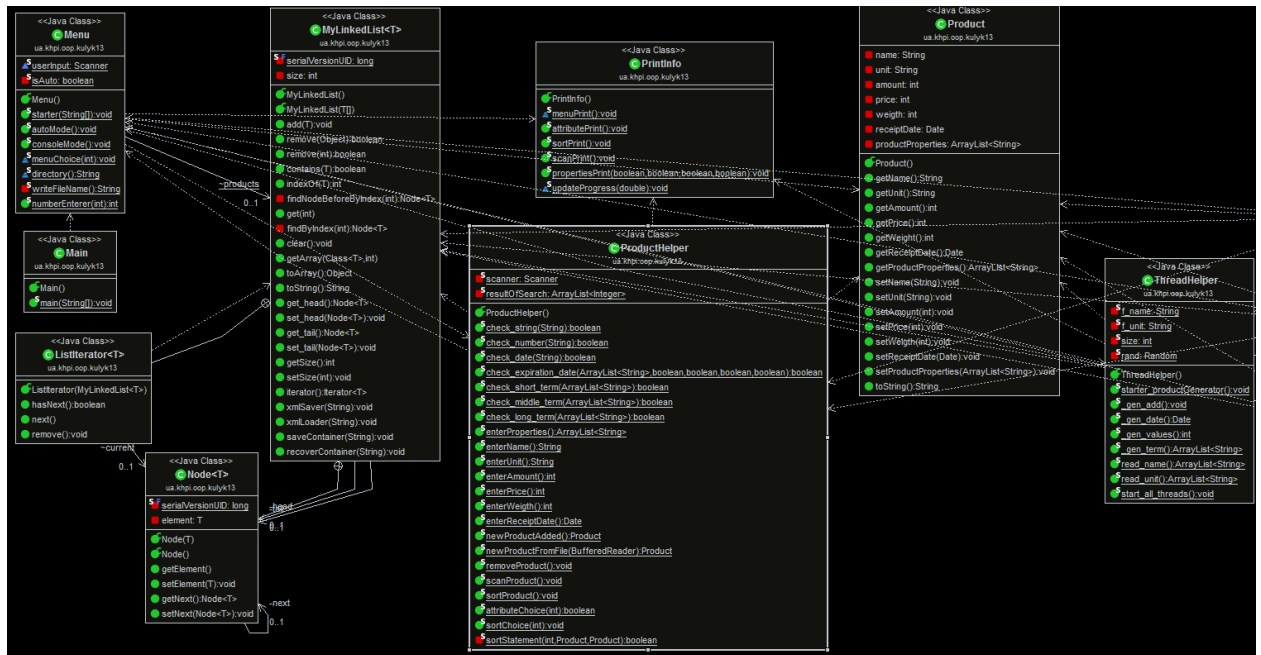


Рисунок 1 – Діаграма класів

2.3 Важливі фрагменти програми

```
public static void start_all_threads() {
    System.out.println("Set the timer [0 - 100 000 ms]: ");
    int timer_num = Menu.numberEnterer(100000);
    System.out.println("Starting all threads...");

    FirstThread first = new FirstThread();
    Thread t1 = new Thread(first, "FirstThread");

    SecondThread second = new SecondThread();
    Thread t2 = new Thread(second, "SecondThread");

    ThirdThread third = new ThirdThread();
    Thread t3 = new Thread(third, "ThirdThread");

    t1.start();
    t2.start();
    t3.start();
}
```

Рисунок 2 – Запуск трьох потоків

```

    Timer timer = new Timer(timer_num, new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent event) {
            System.out.println("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();
    }
    System.out.println("Finishing all threads...");
}

```

Рисунок 3 – Створення таймеру

```

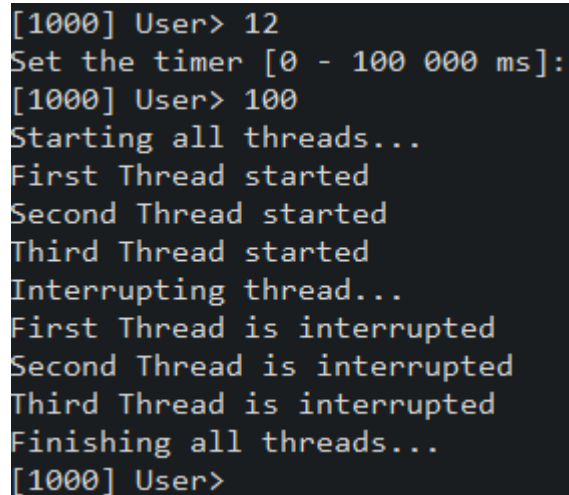
class ThirdThread implements Runnable {
    public void run() {
        int count = 0;
        System.out.println("Third Thread started");
        try {
            for (Product product : Menu.products) {
                Thread.sleep(100);
                if (!Thread.currentThread().isInterrupted()) {
                    if (ProductHelper.check_middle_term(product.getProductProperties())) {
                        count++;
                    }
                } else {
                    throw new InterruptedException();
                }
            }
            System.out.println("Third Thread finished. Products with long term : " + count);
        } catch (InterruptedException e) {
            System.out.println("Third Thread is interrupted");
        }
    }
}

```

Рисунок 4 – Один із потоків

3 ВАРІАНТИ ВИКОРИСТАННЯ

Програма дозволяє створювати об'єкти – товари – , що заносяться у запис каталогу, тобто створюється масив об'єктів. Користувач може додавати об'єкти до масиву, видаляти елементи вибірково, а також очистити увесь масив одним викликом відповідної кнопки меню. Також присутня можливість серіалізувати/десеріалізувати об'єкти з файлу.



```
[1000] User> 12
Set the timer [0 - 100 000 ms]:
[1000] User> 100
Starting all threads...
First Thread started
Second Thread started
Third Thread started
Interrupting thread...
First Thread is interrupted
Second Thread is interrupted
Third Thread is interrupted
Finishing all threads...
[1000] User>
```

Рисунок 5 – Спрацьовування потоків у випадку не встигання таймеру

ВИСНОВКИ

В даній лабораторній роботі організував паралельне виконання декількох частин програми.