Міністерство освіти і науки України

Національний технічний університет

«Харківський політехнічний інститут»

Кафедра «Обчислювальна техніка та програмування»

ЗВІТ

Про виконання лабораторної роботи 13

“Параллельне виконання. Multithreading”

Виконавець:

студент гр. КН-920В

Олексієнко Микита

Харків 2022

Лабораторна робота №13

**Параллельне виконання. Multithreading**

**Мета**

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

**1.Вимоги**

**1.1 Розробник:**

Олексієнко Микита Віталійович

студент групи КН-920В

1. варіант

**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 (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.

**1.3 Прикладні задачі**

Автомобіль: марка, рік випуску, технічні характеристики у вигляді "параметр - значення" (серед яких витрата палива в міському і заміському циклі), технічний стан, ціна.

**2. Опис програми**

**2.1 Засоби ООП:**

У лабороторній роботі 13 , я використовую клас Program, який має у собі лише запуск меню, розробленого у лабороторної роботі 8. Ця работа передбачає удосконалення класу Menu, використанням багатопоточного програмування, завдяки якому обробка контейнеру працювати у рази швидше.

Попередньо оголошено клас Auto, який має поля з індивідуального завдання та методи обробки ціх полей.

**2.2 Ієрархія та структура класів:**

* Auto.java
* Container.java
* Iterator.java ( interface )
* LinkedList.java
* Menu.java
* Program.java

**2.3 Важливі фрагменти програми:**

Menu.java ( додаткові методи та класи ) :

private static void CarContainerInit() throws IOException {  
  
 container.loadFromFile("save\_lab\_13.txt");  
  
 System.out.println("Input max thread`s working time:");  
 Scanner scanner1 = new Scanner(System.in);  
 int time = scanner1.nextInt();  
  
 System.out.println("MAIN thread -> " + Thread.currentThread().getName() + " / " + Thread.currentThread().getId());  
  
 Thread thread1 = new Thread(new Runner("sort", time));  
 thread1.start();  
  
 Thread thread2 = new Thread(new Runner("find", time));  
 thread2.start();  
  
 Thread thread3 = new Thread(new Runner("print", time));  
 thread3.start();  
  
 Scanner scanner2 = new Scanner(System.in);  
 scanner2.nextLine();  
   
 System.exit(0);  
}  
  
private static class Runner implements Runnable {  
  
 private String method = "";  
 private int time = 0;  
  
 Runner ( String method, int time ) {  
 this.method = method;  
 this.time = time;  
 }  
  
 @Override  
 public void run() {  
 switch (method) {  
 case "sort" -> {  
 System.out.println("sort thread -> " + Thread.currentThread().getName() + " / "  
 + Thread.currentThread().getId() );  
 container.sortByPrice();  
 }  
 case "find" -> {  
 System.out.println("find thread -> " + Thread.currentThread().getName() + " / "  
 + Thread.currentThread().getId());  
 container.findCar();  
 }  
 case "print" -> {  
 System.out.println("print thread -> " + Thread.currentThread().getName() + " / "  
 + Thread.currentThread().getId());  
 container.carList();  
 }  
 }  
  
 try {  
 Thread.sleep(time);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
}

**3. Висновок:**

* Ознайомлено з моделлю потоків *Java*.
* Організовано паралельне виконання декількох частин програми.