МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М.Є. Жуковського “ХАІ”

Кафедра 603

Лабораторна робота № 6

# ПРИМЕНЕНИЕ МНОГОПОТОЧНОЙ ОБРАБОТКИ

# В JAVA-ПРИЛОЖЕНИЯХ

з дисципліни програмування Інтернет

ХАІ.603.631п.15З.050103.126322.ПЗ

Виконав студент гр. 631п Кошиль В. А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (№ групи) (П.І.Б)

(підпис, дата)

Перевірив:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(наукова ступінь, вчене звання)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис, дата) (П.І.Б)

Харків – 2015

**Цель работы:** получение практических навыков работы по созданию многопоточных приложений на языке Java.

**Задание**

Разработать и реализовать на языке Java приложение для демонстрации многопоточной обработки информации. В разрабатываемом приложении показать применение классов, интерфейсов и методов для создания и управления потоками вычислений, а также обеспечить их синхронизацию при работе с объектами приложения.

Примечание. В качестве основы можно использовать программные приложения, разработанные в предыдущих лабораторных работах.

**Ход работы**

В качестве основы было взято приложение из лабораторной работы №5. В данном приложении создается автопарк – коллекция объектов, и предоставляется возможность сохранять ее в файл и считывать из файла. Для выполнения данной лабораторной работы исходный код был дополнен: класс для работы с файлом теперь наследуется от Runnable, добавлены поля класса, и аргументы из функций этого класса перенесены в конструктор. Функция записи теперь вызывается в переопределенном методе run(). В MainClass создается объект потока и инициализируется объектом класса ReadWrite. Сохранение происходит в отдельном потоке.

ReadWrite.java

**private** String path;

**private** Vector<Transport> collection;

**private** Boolean isWrite = **false**;

**public** ReadWrite(String path, Vector<Transport> collection, Boolean isWrite){

**this**.path = path;

**this**.collection= collection;

**this**.isWrite = isWrite;

}

@Override

**public** **void** run(){

System.***out***.println("Override method run() calls. Saving works in separate thread.");

**if**(**this**.isWrite){

**this**.SaveCollection();

}

}

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

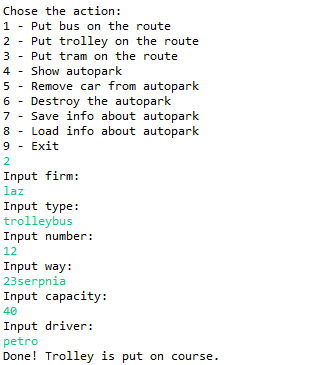
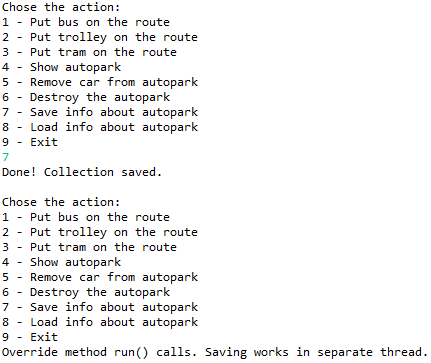
MainClass.java

readerWriter = **new** ReadWrite(path, transports, **true**);

fileThread = **new** Thread(readerWriter);

fileThread.start();

**Выполнение программы**

**Выводы:** в результате выполнения лабораторной работы было разработано и реализовано на языке Java приложение для демонстрации многопоточной обработки информации.

**Листинг**

Main.java

**import** java.io.IOException;

**import** java.util.Scanner;

**import** java.util.Vector;

**import** java.lang.Thread;

**public** **class** MainClass {

**private** **static** Scanner *scan*;

**public** **static** **void** main(String[] args) {

Vector<Transport> transports = **new** Vector<Transport>();

Transport transport;

**int** choose=0;

String path = "D:/SkyDrive/3 курс/!2 cemester/!COMPLETED/Java/Laba5/collection.txt";

ReadWrite readerWriter = **null**;

*scan* = **new** Scanner(System.***in***);

Thread fileThread = **null**;

**while**(**true**){

System.***out***.println("\nChose the action:");

System.***out***.println("1 - Put bus on the route");

System.***out***.println("2 - Put trolley on the route");

System.***out***.println("3 - Put tram on the route");

System.***out***.println("4 - Show autopark");

System.***out***.println("5 - Remove car from autopark");

System.***out***.println("6 - Destroy the autopark");

System.***out***.println("7 - Save info about autopark");

System.***out***.println("8 - Load info about autopark");

System.***out***.println("9 - Exit");

choose = *scan*.nextInt();

**switch**(choose){

**case** 1:

transport = **new** Bus();

System.***out***.println("Input firm: ");

transport.SetFirm(*scan*.next());

System.***out***.println("Input type: ");

transport.SetType(*scan*.next());

System.***out***.println("Input number: ");

transport.SetNumber(*scan*.nextInt());

System.***out***.println("Input way: ");

transport.SetWay(*scan*.next());

System.***out***.println("Input capacity: ");

transport.SetCapacity(*scan*.nextInt());

System.***out***.println("Input driver: ");

transport.SetDriver(*scan*.next());

transports.add(transport);

System.***out***.println("Done! Bus is put on course.");

**break**;

**case** 2:

transport = **new** Trolley();

System.***out***.println("Input firm: ");

transport.SetFirm(*scan*.next());

System.***out***.println("Input type: ");

transport.SetType(*scan*.next());

System.***out***.println("Input number: ");

transport.SetNumber(*scan*.nextInt());

System.***out***.println("Input way: ");

transport.SetWay(*scan*.next());

System.***out***.println("Input capacity: ");

transport.SetCapacity(*scan*.nextInt());

System.***out***.println("Input driver: ");

transport.SetDriver(*scan*.next());

transports.add(transport);

System.***out***.println("Done! Trolley is put on course.");

**break**;

**case** 3:

transport = **new** Tram();

System.***out***.println("Input firm: ");

transport.SetFirm(*scan*.next());

System.***out***.println("Input type: ");

transport.SetType(*scan*.next());

System.***out***.println("Input number: ");

transport.SetNumber(*scan*.nextInt());

System.***out***.println("Input way: ");

transport.SetWay(*scan*.next());

System.***out***.println("Input capacity: ");

transport.SetCapacity(*scan*.nextInt());

System.***out***.println("Input driver: ");

transport.SetDriver(*scan*.next());

transports.add(transport);

System.***out***.println("Done! Tram is put on course.");

**break**;

**case** 4:

**for**(**int** i=0; i<transports.size(); i++){

System.***out***.println("The "+(i+1)+" car in autopark: ");

System.***out***.println("Car made by: " + transports.elementAt(i).GetFirm());

System.***out***.println("Car type is: " + transports.elementAt(i).GetType());

System.***out***.println("Car number is: " + transports.elementAt(i).GetNumber());

System.***out***.println("Car course is: " + transports.elementAt(i).GetWay());

System.***out***.println("Car capacity is: " + transports.elementAt(i).GetCapacity());

System.***out***.println("Driver name is: " + transports.elementAt(i).GetDriver());

}

**break**;

**case** 5:

System.***out***.println("Input position will be removed!");

transports.remove(*scan*.nextInt() - 1);

**break**;

**case** 6:

System.***out***.println("Autopark will be destroyed!");

transports.clear();

**break**;

**case** 7:

readerWriter = **new** ReadWrite(path, transports, **true**);

fileThread = **new** Thread(readerWriter);

fileThread.start();

//readerWriter.SaveCollection();

System.***out***.println("Done! Collection saved.");

**break**;

**case** 8:

readerWriter = **new** ReadWrite(path, transports, **false**);

**try** {

Vector<String> info = readerWriter.LoadCollection();

System.***out***.println("Done! Info about collection loaded.");

**for**(String str : info){

System.***out***.println(str);

}

}

**catch** (IOException e) {

System.***err***.println(e);

}

**break**;

**case** 9:

System.*exit*(0);

}

}

}

}

ReadWrite.java

**import java.io.\*;**

**import java.util.Vector;**

**public class ReadWrite implements Runnable{**

**private String path;**

**private Vector<Transport> collection;**

**private Boolean isWrite = false;**

**public ReadWrite(String path, Vector<Transport> collection, Boolean isWrite){**

**this.path = path;**

**this.collection= collection;**

**this.isWrite = isWrite;**

**}**

**public void SaveCollection(){**

**Writer writer = null;**

**try {**

**writer = new FileWriter(this.path);**

**for (int i = 0; i<this.collection.size(); i++) {**

**writer.write("The "+(i+1)+" car in autopark: ");**

**writer.write("\n");**

**writer.write("Car made by: " + this.collection.elementAt(i).GetFirm());**

**writer.write("\n");**

**writer.write("Car type is: " + this.collection.elementAt(i).GetType());**

**writer.write("\n");**

**writer.write("Car number is: " + this.collection.elementAt(i).GetNumber());**

**writer.write("\n");**

**writer.write("Car course is: " + this.collection.elementAt(i).GetWay());**

**writer.write("\n");**

**writer.write("Car capacity is: " + this.collection.elementAt(i).GetCapacity());**

**writer.write("\n");**

**writer.write("Driver name is: " + this.collection.elementAt(i).GetDriver());**

**writer.write("\n");**

**}**

**writer.flush();**

**}**

**catch (IOException e) {**

**System.err.println(e);**

**}**

**finally {**

**if (writer != null) {**

**try {**

**writer.close();**

**}**

**catch (IOException ex) {**

**}**

**}**

**}**

**}**

**public Vector<String> LoadCollection() throws FileNotFoundException, IOException{**

**String line = null;**

**Vector<String> lines = new Vector<String>();**

**File file = new File(this.path);**

**BufferedReader br = new BufferedReader(new InputStreamReader(new FileInputStream(file)));**

**while ((line = br.readLine()) != null) {**

**lines.add(line);**

**}**

**br.close();**

**return lines;**

**}**

**@Override**

**public void run(){**

**System.out.println("Override method run() calls. Saving works in separate thread.");**

**if(this.isWrite){**

**this.SaveCollection();**

**}**

**}**

**}**