**Міністерство Освіти І НАУКИ України**

**Національний університет "Львівська політехніка"**

Інститут **КНІТ**

Кафедра **ПЗ**

**ЗВІТ**

До лабораторної роботи № 4

**На тему:** *“* *Робота з потоками та виключеннями”*

**З дисципліни:** *“Кросплатформне програмування”*

**Лектор:**

викладач каф. ПЗ

Подольський І. В.

**Виконав:**

ст. гр. ПЗ-31

Підгорецький В. І.

**Прийняв:**

асистент каф. ПЗ

Шкраб Р. Р.

« \_\_\_\_ » \_\_\_\_\_\_\_\_ 2018 р.

∑= \_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Львів – 2018

**Тема роботи:** Робота з потоками та виключеннями.

**Мета роботи**: Вивчити основнову роботи з потоками Java, освоїти роботу виключень у JAVA.

**Теоретичні відомості**

Потік (англ. thread) або повніше потік виконання (англ. thread of execution), часто застосовуються назви нитка, нитка виконання та англіцизм тред — в інформатиці так називається спосіб програми розділити себе на дві чи більше паралельні задачі. Реалізація потоків та процесів відрізняються в різних операційних системах, але загалом потік міститься всередині процесу і різні потоки одного процесу спільно розподіляють деякі ресурси, у той час як різні процеси ресурси не розподіляють.

**Завдання на лабораторну**

В залежності від варіанту N-D будуть різні

1. Згенеруйте N файлів розміром N2 Мб з текстовим контентом.

Кожен файл напвнений однією літерою алфавіту.

Наприклад перший – аааа, другий – bbbb і т.д.

\При роботі з файлами використовуйте блок try-catch для відстедення виключних ситуацій

1. Створіть N потоків які працюють кожен зі своїм файлом.

Потоки створюйте за допомогою імплементації інтерфейсу Runnable

1. Задача кожного потоку вичитати зі свого файлу N3 байт даних і помістити послідовно у власний буфер. Тобто у буфері отримуємо мікс: aaa-bbb-ccc

При роботі з буфером не забувайте про синхронізацію.

1. Після міксування записуємо вміст буфера у вихідний файл out.txt
2. Повторюємо доти поки увесь вміст вхідних файлів буде зміксований в один.

Варіант 2.

1. N=2

2. N2 = 1

3. N3=64

**Хід роботи**

Код програми:

import java.io.File;

import java.io.FileOutputStream;

import java.io.IOException;

import java.util.Random;

import java.util.Scanner;

public class Main {

public static void main(String[]args){

Scanner in = new Scanner(System.in);

Out out = new Out();

int NumOfFiles=2;

int FileSize=1024;

// System.out.println("Enter num of files:");

while(true) {

System.out.println("Enter num of files:");

if (in.hasNextInt() == true) {

NumOfFiles=in.nextInt();

//break;

} else {

continue;

}

System.out.println("Enter size of files(in kbytes):");

if (in.hasNextInt() == true) {

FileSize=in.nextInt()\*1024;

// break;

} else {

continue;

}

break;

}

String[] FilesPath;

Thread []threads = new Thread[NumOfFiles];

FilesPath=new String[NumOfFiles];

createFile(NumOfFiles, FileSize, FilesPath);

for(int i=0; i<NumOfFiles; i++) {

threads[i]=new Thread(new Read(FilesPath[i], out));

threads[i].run();

}

System.out.println(out.out);

//rand();

out.WriteToFile();

}

static public void createFile(int NumOfFiles, int FileSize, String []FilesPath){

//Створюємо файли

//єCounter починється з acii коду букви а і йде до кінця алфавіту або до необхідного числа файлів

for(int i=97; i<122 && i<97+NumOfFiles; i++){

FilesPath[i-97]=(char)i+".txt";

try(FileOutputStream fos=new FileOutputStream(FilesPath[i-97])){

byte letter = (byte)i;

for(int j=0; j<FileSize; j++){//Записуємо у файл FileSize байт символів

fos.write(letter);

}

}catch (IOException ex){

System.out.println(ex.getMessage());

}

}

}

static void rand(){

Random rand = new Random();

System.out.println(rand.nextInt(1000));

}

}

public class Out {

public String out="";

public void WriteToFile() {

try (FileOutputStream fos = new FileOutputStream("out.txt")) {

byte[] buffer=out.getBytes();

fos.write(buffer, 0, buffer.length);

} catch (IOException ex) {

System.out.println((ex.getMessage()));

}

}

}

public class Read implements Runnable{

String path;

String buff;

Out out;

public Read(String path, Out out){

this.path=path;

this.out=out;

buff="";

}

@Override

public void run() {

int steps=1024/64;

try(FileInputStream fin=new FileInputStream(path)){

for(int i=0; i<steps; i++) {

byte letter;

for (int j = 0; j < 64; j++) {

letter = (byte) fin.read();

buff += (char) letter;

}

synchronized (out) {

out.out += buff + "-";

}

buff = "";

}

}catch (IOException ex){

System.out.println(ex.getMessage());

}

}

}

**Результат виконання**

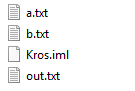


Рис. 1. Результат виконання програми

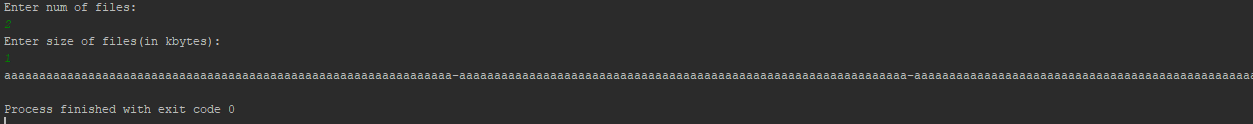


Рис. 2. Результат виконання програми методом з синхронізацією

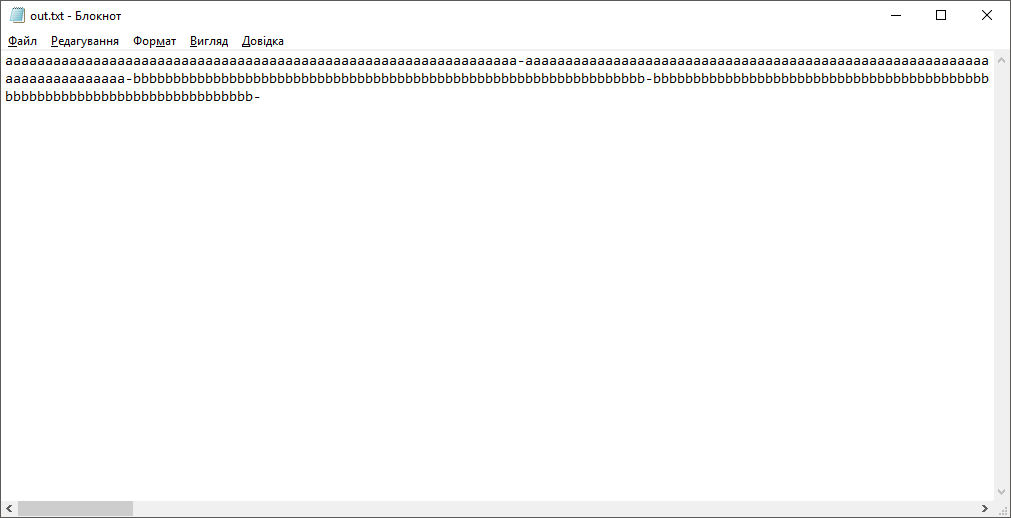


Рис. 3 Вміст файлу out.txt

**Висновки**

На даній лабораторній роботі, я ознайомився з винятковими ситуаціями в Java та багато поточністю. Визначив методи синхронізації потоків, практично випробував ключове слово synchronized в Java. Ознайомився з ієрархією виняткових ситуацій в Java. Навчився працювати з буфером та файлами у Java.