**Ministerul Educaţiei al Republicii Moldova**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică şi Microelectronică**

**Raport**

Lucrare de laborator **№2**

Disciplina: Programarea in Retea

A îndeplinit: st.gr.FI-161 Panfil Nicolae

A controlat: Gavrilita Mihai

Chișinău 2019

**Lucrare de Laborator Nr.2**

**Scop:**

* Realizarea firelor de execuţie în Java/C#.
* Proprietăţile firelor. Stările unui fir de execuţie.
* Lansarea, suspendarea şi oprirea unui fir de execuţie.
* Grupuri de Thread-uri.
* Elemente pentru realizarea comunicării şi sincronizării.

**Sarcina:**

* Fiind data diagrama dependentelor cauzale de modelat activitatile reprezentate de acestea prin fire de executie.
* Pentru a initiaza un thread în Java,avem extensia Thread **public class Threads exteds Thread** si implementarea **Runnable**. Am efectuat **ArrayList** de threaduri, variabila booleana si o variabila pentru numele firului. În continuare am creat metodele de obținere și setare pentru **threadWaiting** (lista de fire). Stiind ca fiecare fir are proprietatea de **monitor lock** sau poate implementa metoda de sincronizare, ceea ce înseamnă că fiecare fir are o cheie care îi permite să execute sarcina. Funcția **runThreads()** face exact acest lucru, oprește execuția firului, dacă aceasta ar trebui să aștepte execuția anterioară a firului. De exemplu, atunci cand firul 5,6,4 sau 7 va termina execuția va face **notifyAll()** despre aceasta și următorul thread care depinde de unul din aceste threaduri, va începe execuția.
* În clasa Main, am creat, conform grafului meu, threadurile. Apoi am numit metoda **setThreadWaiting()** care va stabili dependența. Apoi am folosit metoda **start()** predefinită pentru a porni threadurile și **join()** pentru a comunica despre executarea firelor.

**Code Source:**

* **class Main**

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

public class Main {

public static void main(String[] args) {

Threads threads1 = new Threads("Node1");

Threads threads2 = new Threads("Node2");

Threads threads3 = new Threads("Node3");

Threads threads4 = new Threads("Node4");

Threads threads5 = new Threads("Node5");

Threads threads6 = new Threads("Node6");

Threads threads7 = new Threads("Node7");

threads1.setThreadWaiting(new ArrayList<>(Arrays.asList(threads2,threads3)));

threads2.setThreadWaiting(new ArrayList<>(Arrays.asList(threads5,threads6)));

threads3.setThreadWaiting(new ArrayList<>(Arrays.asList(threads4,threads7)));

threads1.start();

threads2.start();

threads3.start();

threads4.start();

threads5.start();

threads6.start();

threads7.start();

try {

threads1.join();

threads2.join();

threads3.join();

threads4.join();

threads5.join();

threads6.join();

threads7.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println("Finish");

};

}

* **class Threads**

import java.util.ArrayList;

public class Threads extends Thread {

private ArrayList<Threads> threadWaiting;

private boolean finished = false;

private String threadName;

public Threads(String threadName) { this.threadName = threadName;

}

@Override

public void run() {this.runThreads();

}

public ArrayList<Threads> getthreadWaiting() { return threadWaiting;

}

public void setThreadWaiting(ArrayList<Threads> threadWaiting) {

this.threadWaiting = threadWaiting;

}

public boolean isFinished() { return finished;

}

public void setFinished(boolean finished) { this.finished = finished;

}

public String getThreadName() { return threadName;

}

public void setThreadName(String threadName) throws IllegalMonitorStateException {

this.threadName = threadName;

}

public void runThreads() {

if (threadWaiting != null) {

for (Threads threads : threadWaiting) {

if (!threads.isFinished()) {

synchronized (threads) {

try {

threads.wait();

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}

}

System.out.println(threadName);

this.finished = true;

synchronized (this) {

this.notifyAll();

}

}

}

##Rezultat:

* Rezultatul se poate modifica de fiecare dată, deoarece există mai multe modalități posibile de a executa toate taskurile din threaduri.