Ministerul Educației, Culturii si Cercetarii Republicii Moldova
Universitatea Tehnică a Moldovei
Facultatea Calculatoare, Informatică și Microelectronică
Departamentul Ingineria Software și Automatică

Programarea aplicațiilor distribuite

LUCRARE DE LABORATOR nr. 1

Tema: Multi Threading

A efectuat:
st. gr. TI-151 F/R Constantinecu Nadejda

A verificat:
Lector universitar Antohi Ionel

Chișinău 2019

Tema: Multi threading.

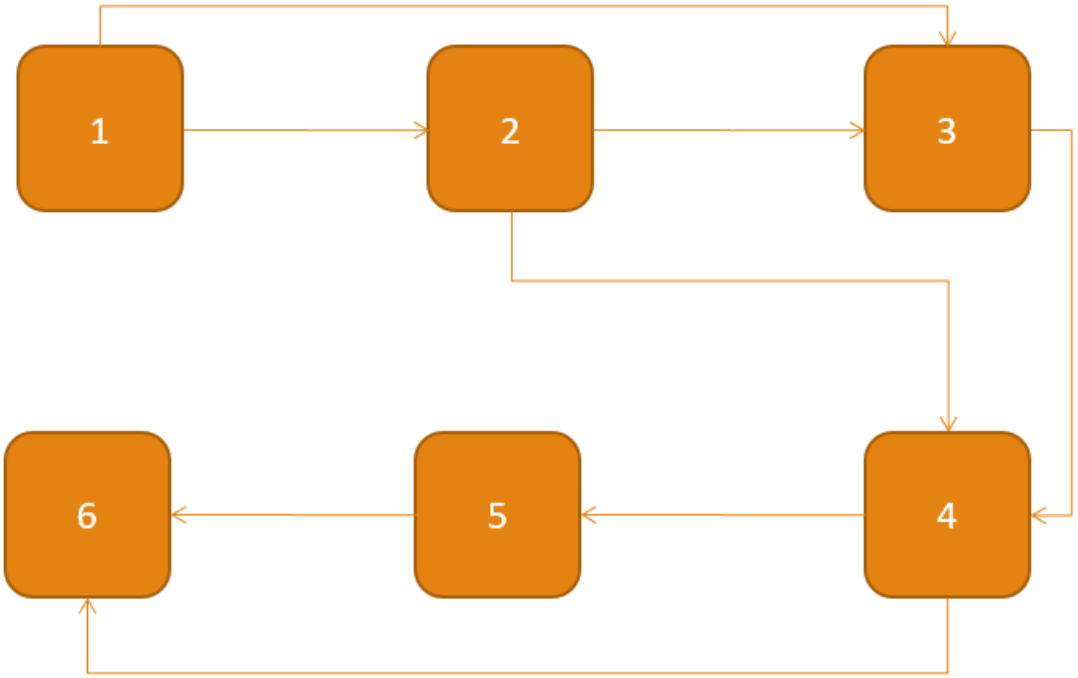
Scopul lucrării: Fiind dată diagrama dependențelor cauzale de modelat activitățile reprezentate de acestea prin fire de execuție.

Noțiuni teoretice:

Diagrama dependențelor cauzale determină o mulțime ordonată de evenimente/activități ordonate de relația de cauzalitate. Evenimentele/activitățile sunt reprezentate printr-un dreptunghi rotunjit, iar dependențele prin săgeți, primind astfel un graf orientat aciclic.

Diagrama

Varianta 6



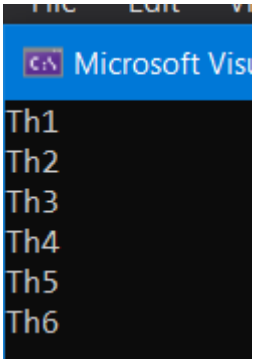
Codul sursă:

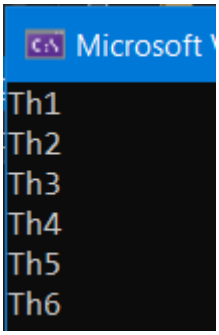
```
using System;
using System.Collections.Generic;
using System.Threading;

namespace thread
{
    public class MultiThread
    {
        private readonly CountdownEvent _countdown;
        private readonly List<CountdownEvent> _waiters;
        private int _threadsToWait;
        private int _workTime;
        private readonly Thread _thread;
        public string Name { get; }
        public MultiThread(string name, int workTime)
        {
```

```
        Name = name;
        _workTime = workTime;
        _countdown = new CountdownEvent(0);
        _waiters = new List<CountdownEvent>();
        _thread = new Thread(DoWork);
    }

    public void WaitFor(params MultiThread[] workersToWait)
    {
        CheckIsRunning(); foreach (var MultiThread in workersToWait)
        {
            if (MultiThread == this || MultiThread._waiters.Contains(_countdown)) continue;
            MultiThread._waiters.Add(_countdown);
            _threadsToWait++;
        }
    }

    private void CheckIsRunning()
    {
        if (_thread.ThreadState == ThreadState.Running)
        {
            throw new ThreadStateException("Thread is running");
        }
    }

    private void DoWork()
    {
        try
        {
            _countdown.Reset(_threadsToWait);
            _countdown.Wait();
            Console.WriteLine(Name);
            Thread.Sleep(_workTime);
        }
        catch (Exception e)
        {
            Console.WriteLine(e); throw;
        }
        finally
        {
            foreach (var waiter in _waiters)
            {
                waiter.Signal();
            }
        }
    }

    public void Start()
    {
        _thread.Start();
    }
}

public class Program
{
    public static void Main()
    {
        var rand = new Random();
        var w1 = new MultiThread("Th1", rand.Next(2000));
        var w2 = new MultiThread("Th2", rand.Next(2000));
        var w3 = new MultiThread("Th3", rand.Next(2000));
        var w4 = new MultiThread("Th4", rand.Next(2000));
        var w5 = new MultiThread("Th5", rand.Next(2000));
        var w6 = new MultiThread("Th6", rand.Next(2000));
```

```
            w2.WaitFor(w1);
            w3.WaitFor(w1, w2);
            w4.WaitFor(w2, w3);
            w5.WaitFor(w4);
            w6.WaitFor(w4, w5);

            MultiThread[] workers = { w1, w2, w3, w4, w5, w6 };
            foreach (var worker in workers) { worker.Start(); }
        }
    }
}
```

Rezultatele obținute:





Concluzie: La elaborarea acestei lucrări de laborator am studiat firele de execuție în programare precum și metodele lor de sinhronizare. Am lansat oprit și suspendat fire de execuție. Pentru laboratorul dat a fost folosită metoda de sinchronizare CountDown.