Sprawozdanie nr 1

Data ćwiczeń: 19.03.2015

Data oddania sprawozdania: 23.04.2015

* Tematyka

Tematem poruszanym na pierwszych laboratoriach z teorii współbieżności było krótkie wprowadzenie w temat współbieżności oraz tego, jak została ona zaimplementowana w języku Java.

Współbieżność jest realizowana dzięki wykorzystaniu wątków, które niezależnie od ilości procesorów, w które wyposażony jest dany komputer, pozwalają na jednoczesne wykonywanie pewnych operacji. Jako przykład można podać sytuację, w której dany proces jest *oczekujący*, na przykład na wykonanie jakiejś operacji wejścia/wyjścia. Podczas tego oczekiwania *w tle* mogą się wykonać konkretne obliczenia. Dzięki takiemu zabiegowi zyskujemy na szybkości wykonywania się danego oprogramowania.

Istnieje wiele sytuacji, w których można, a nawet należy skorzystać z wątków. Należą do nich m.in.: złożone obliczenia, które mogą być podzielone na mniejsze kawałki (ang. *divide and conquer*), GUI, operacje które nie wymagają stałego nadzoru, tzn mogą wykonać się asynchronicznie.

W języku programowania Java istnieje kilka sposobów, na które można utworzyć wątek. Najczęściej dana klasa rozszerza klasę *Thread*. Można również stworzyć klasę implementującą interfejs *Runnable*, bądź *Callable*.

* Zadanie

Na zajęciach należało zaimplementować sytuację, w której pole *int* klasy *singleton* jest odpowiednio inkrementowane, bądź dekrementowane przez dwa wątki. Jest to sytuacja tzw. wyścigu, w wyniku której nie możemy przewidzieć jaka będzie wartość zmiennej na końcu programu.

* Kod

/\*\*

\* Created by Patryk on 2015-03-19.

\*/

public class Main {

public static void main(String[] args)

throws InterruptedException {

final long startTime = System.currentTimeMillis();

final Num myNum = new Num();

Thread numeroUno = new Thread(){

public void run() {

for(int i=0; i<1000000; i++) {

myNum.inc();

}

}

};

Thread numeroDuo = new Thread(){

public void run() {

for(int i=0; i<1000000 ; i++) {

myNum.dec();

}

}

};

numeroUno.start();

numeroDuo.start();

numeroDuo.join();

numeroUno.join();

final long endTime = System.currentTimeMillis();

System.out.println(myNum.getVal());

System.out.println(endTime - startTime);

}

}

public class Num extends Thread {

private int val;

public Num() {

this.val = 0;

this.numOp = 0;

}

public void inc() {

this.val++;

}

public void dec() {

this.val--;

}

public int getVal() {

return val;

}

}