Łukasz Jaworski 155 NCI

Lab 01

Programowanie współbieżne i równoległe. Wątki i procesy.

W tym zadaniu mieliśmy wykorzystać metodę synchroniczną oraz asynchroniczną, do pobrania danych z internetu oraz wykorzystać działanie funkcji matematycznej, aby porównać wyniki czasów działania wątków.

```
Sync Thread 0 Running
Sync Thread 1 Running
Sync Thread 2 Running
Sync Thread 3 Running
Sync Thread 4 Running
Sync Thread 5 Running
Sync Thread 6 Running
Sync Thread 7 Running
Sync Thread 8 Running
Sync Thread 9 Running
Summary time of synchronic threads: 1.536s
Async Thread 0 Running
Async Thread 1 Running
Async Thread 2 Running
Async Thread 3 Running
Async Thread 4 Running
Async Thread 5 Running
Async Thread 6 Running
Async Thread 7 Running
Async Thread 8 Running
Async Thread 9 Running
Summary time of asynchronic threads: 0.266s
```

W pierwszej części pobrałem 10 razy obrazek z internetu przy pomocy funkcji **ImageIO.read(**url).

W przypadku, kiedy wykorzystujemy równoczesny dostęp do sieci operacje pobierania są dużo szybsze od dostępu synchronicznego.

Do drugiej części zadania wykorzystałem ciąg Fibonacci'ego, wykonując 40 cykli pętli w obu przypadkach.

```
4181
Fibonacci Sync Thread 2 Running
6765
10946
17711
28657
46368
75025
121393
196418
317811
514229
Fibonacci Sync Thread 3 Running
832040
1346269
2178309
3524578
5702887
9227465
14930352
24157817
39088169
63245986
Summary time of synchronic threads: 0.875s
```

```
Fibonacci Async Thread 0 Running
Fibonacci Async Thread 1 Running
55
89
144
233
Fibonacci Async Thread 2 Running
0
1
1
2
3
5
8
13
21
34
377
610
Fibonacci Async Thread 3 Running
14930352
24157817
39088169
63245986
Summary time of asynchronic threads: 0.866s
```

W tym przypadku lepiej widać zachowanie systemu asynchronicznego, który dostaje dostęp w losowej kolejności. W tym przypadku jednak chodziło o działanie matematyczne, które dało metodzie asynchronicznej niewiele lepszy rezultat czasowy.