

Szymon, Michoń

Programowanie równoległe. Przetwarzanie równoległe i rozproszone.

Sprawozdanie z laboratorium 6.

Celem laboratorium było:

- Opanowanie podstaw tworzenia wątków w Javie.
- Opanowanie podstawowych metod synchronizacji w Javie.

W ramach zajęć zrealizowałem następujące kroki:

1. Utworzyłem nowy projekt w IntelliJIDE i dodałem do projektu pliki Histogram_test.java oraz Obraz.java, kod uruchomiłem i zobaczyłem tworzoną na podstawie zdefiniowanej przezemnie wielkości tablicę losowych znaków
2. Utworzyłem wykonywanie równoległe w tym celu:
 - a. W klasie histogram_test odkomentowałem kod odpowiedzialny za pobieranie liczby wątków od użytkownika i uruchamianie wątków. Dodatkowo zakomentowuję linie odpowiedzialne za sekwencyjne wywołanie.

```
System.out.println("Set number of threads");
int num_threads = scanner.nextInt();

Watek[] NewThr = new Watek[num_threads];

for (int i = 0; i < num_threads; i++) {
    (NewThr[i] = new Watek(i, obraz_1)).start();
}

for (int i = 0; i < num_threads; i++) {
    try {
        NewThr[i].join();
    } catch (InterruptedException e) {}
}
}
```

- b. Utworzyłem nową klasę Wątek, dziedziczy ona po klasie Thread, w konstruktorze posiada dwie zmienne, img typu Obraz oraz i typu int. Dodatkowo posiada metodę run uruchamiającą obliczanie i wyświetlanie tablicy.

```
package com.company;

public class Watek extends Thread {
    private Obraz img;
    private int i;

    @Override
    public void run() {
        img.calculate_histogram(i);
        img.print_histogram(i);
    }

    public Watek(int i, Obraz img) {
        this.img = img;
        this.i = i;
    }
}
```

- c. W klasie obraz podmieniłem linię odpowiedzialną za losowe znaki zgodnie z wcześniej przygotowanym komentarzem w kodzie

```
for(int i=0;i<n;i++) {
    for(int j=0;j<m;j++) {
        //tab[i][j] = tab_symb[random.nextInt(94)]; // ascii 33-127
        tab[i][j] = (char)(random.nextInt( bound: 94)+33); // ascii 33-127
        //System.out.print(tab[i][j]+" ");
    }
}
```

- d. W calculate_histogram również dokonałem zmiany zgodnie z wcześniej przygotowanym komentarzem, dodatkowo jako argument metody przekazuję zmienną index typu int przy okazji pozwalającą na pozbycie się wewnętrznej pętli

```
public void calculate_histogram(int index){
    for(int i=0;i<size_n;i++) {
        for(int j=0;j<size_m;j++) {
            // optymalna wersja obliczania histogramu, wykorzystująca fakt, że symbole
            // można przekształcić w indeksy tablicy histogramu
            // histogram[(int)tab[i][j]-33]++;

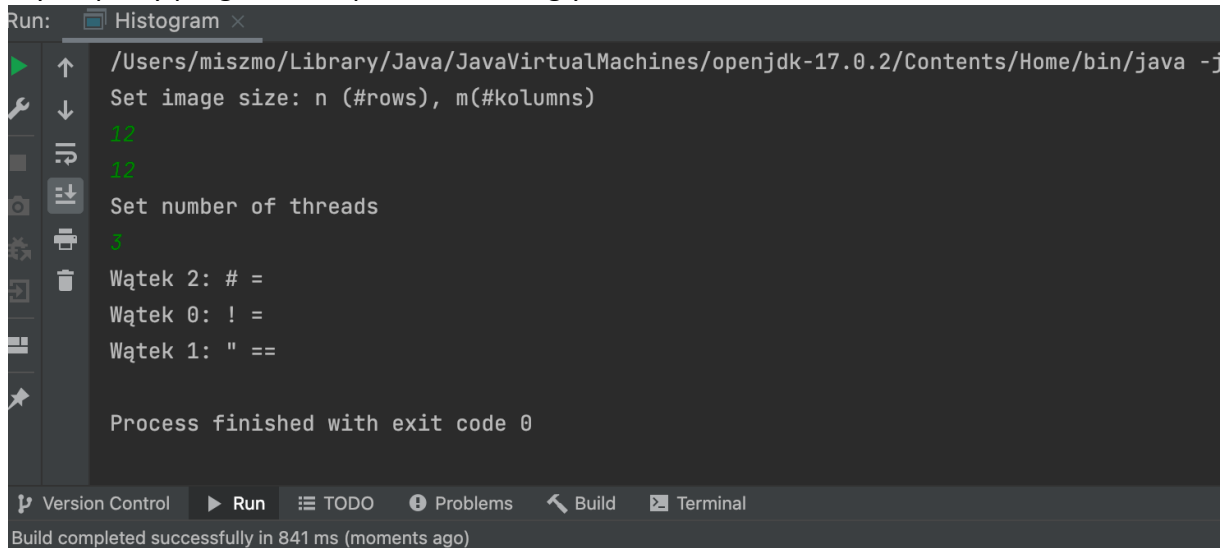
            // wersja bardziej ogólna dla tablicy symboli nie utożsamianych z indeksami
            // tylko dla tej wersji sensowne jest zrównoleglenie w dziedzinie zbioru z
            //for(int k=0;k<94;k++) {
            //if(tab[i][j] == tab_symb[k]) histogram[k]++;
            if(tab[i][j] == (char)(index+33)) histogram[index]++;
            //}
        }
    }
}
```

- e. Również zmieniam metodę print_histogram w taki sposób by każdy wątek mógł tworzyć swoją ilustrację

```
1 related problem
public void print_histogram(int index){
    //for(int i=0;i<94;i++) {
    //System.out.print(tab_symb[i]+" "+histogram[i]+"\\n");
    //System.out.print((char)(i+33)+" "+histogram[i]+"\\n");
    //}

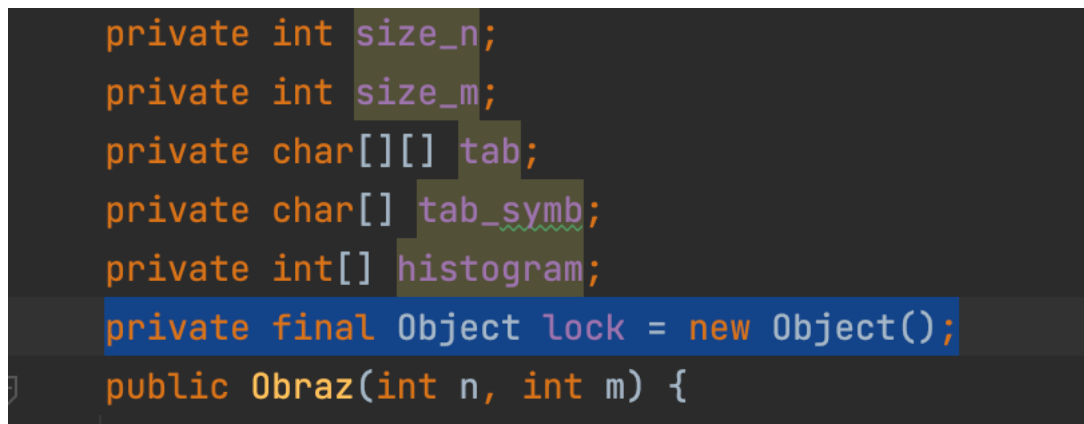
    System.out.print("Wątek " + index + ": "+ (char)(index+33)+" ");
    for(int i=0;i<histogram[index];i++){
        System.out.print("=");
    }
    System.out.print("\\n");
}
}
```

3. Wynik pracy programu w sposób równoległy



```
Run: Histogram x
/Users/miszmo/Library/Java/JavaVirtualMachines/openjdk-17.0.2/Contents/Home/bin/java -j
Set image size: n (#rows), m(#kolumns)
12
12
Set number of threads
3
Wątek 2: # =
Wątek 0: ! =
Wątek 1: " ==
Process finished with exit code 0
Version Control Run TODO Problems Build Terminal
Build completed successfully in 841 ms (moments ago)
```

4. Następnie rozszerzyłem program o interfejs Runnable, w tym celu stworzyłem nową klasę WątekRunnable która implementuje interfejs Runnable i jest zmodyfikowaną wersją klasy Wątek która dziedziczyła po Thread, dodana została zmienna dPorcja typu int oznaczająca porcję jaką bierze wątek przy dekompozycji
5. Również w klasie Obraz zaszły kolejne zmiany, dodałem zmienną typu object do blokowania wątku



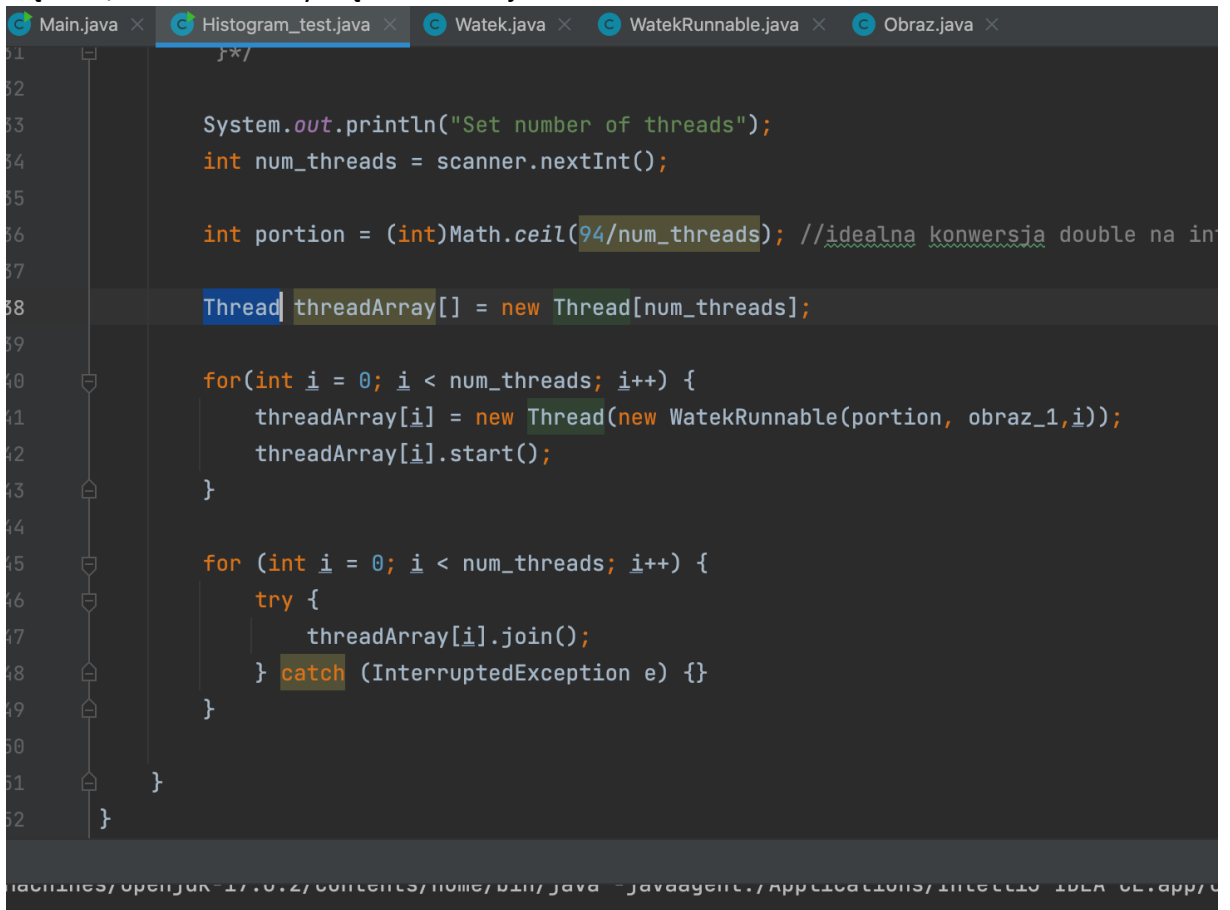
```
private int size_n;
private int size_m;
private char[][] tab;
private char[] tab_symb;
private int[] histogram;
private final Object lock = new Object();
public Obraz(int n, int m) {
```

6. Oraz dodałem nową metodę do wyświetlania histogramu korzystając z tej zmiennej



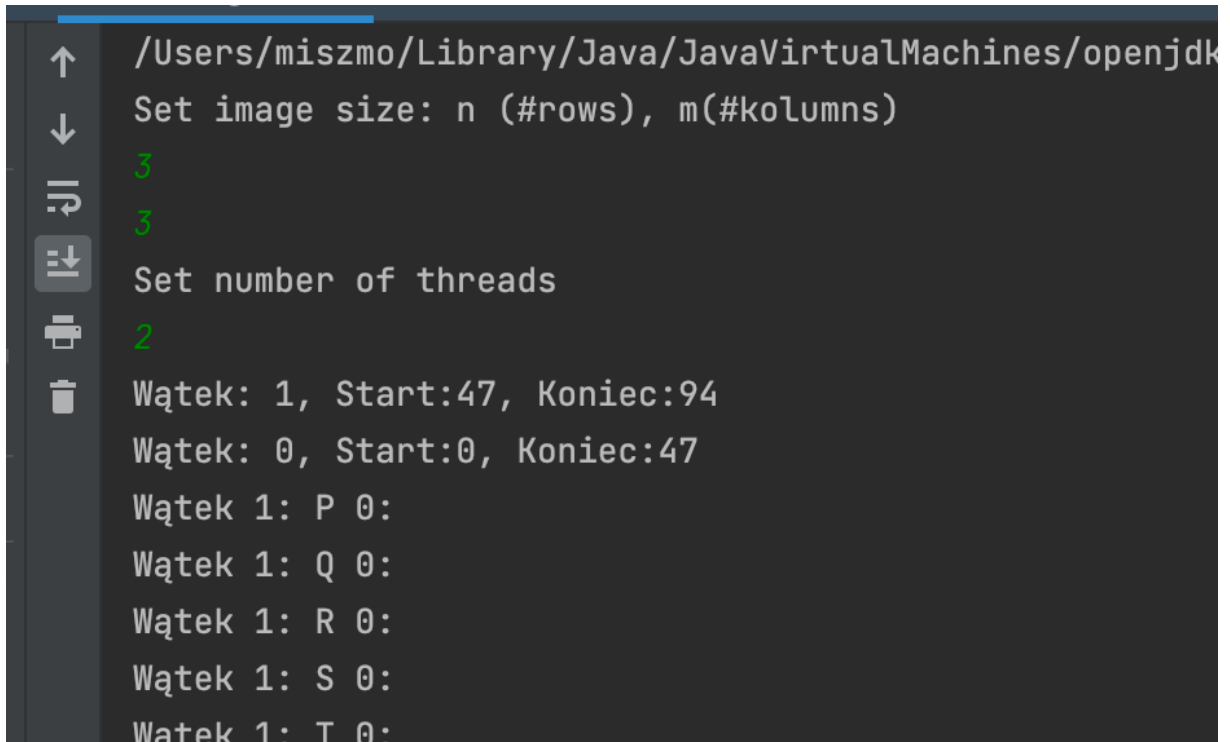
```
public void print_histogram2(int index, int num) {
    synchronized (lock) {
        System.out.print("Wątek " + index + ": " + (char)(num+33) + " " + histogram[num] + ": ");
        for(int i = 0; i < histogram[num]; i++) {
            System.out.print((char)(num+33));
        }
        System.out.print("\n");
    }
}
```

7. Pozostały jedynie zmiany w głównej klasie, dzielimy na porcje, tworzymy tablice wątków, i uruchamiamy wątki z interfejsem Runnable



```
31  }
32
33  System.out.println("Set number of threads");
34  int num_threads = scanner.nextInt();
35
36  int portion = (int)Math.ceil(94/num_threads); //idealna konwersja double na int
37
38  Thread threadArray[] = new Thread[num_threads];
39
40  for(int i = 0; i < num_threads; i++) {
41      threadArray[i] = new Thread(new WatekRunnable(portion, obraz_1,i));
42      threadArray[i].start();
43  }
44
45  for (int i = 0; i < num_threads; i++) {
46      try {
47          threadArray[i].join();
48      } catch (InterruptedException e) {}
49  }
50
51  }
52  }
```

8. Wynik tego programu to:



```
/Users/miszmo/Library/Java/JavaVirtualMachines/openjdk-17.0.2/Contents/Home/bin/java -javaagent:/Applications/IntelliJ\ IDEA\ CE.app/Contents/bin/jrebel.jar -jar ./Histogram_test.jar
Set image size: n (#rows), m(#kolumns)
3
3
Set number of threads
2
Wątek: 1, Start:47, Koniec:94
Wątek: 0, Start:0, Koniec:47
Wątek 1: P 0:
Wątek 1: Q 0:
Wątek 1: R 0:
Wątek 1: S 0:
Wątek 1: T 0:
```