Programowanie równoległe
Laboratorium 9

Sprawozdanie

Zadanie 1

Zadanie brzmi: «Na podstawie przykładowego programu zapoznaj się z tworzeniem wątków w Javie a następnie napisz program sumujący dwa losowe wektory o ustalonej długości n. Zaprojektuj właściwe zrównoważenie obciążenia wątków dla przypadku gdy n nie jest podzielne przez liczbę wątków.»

Kod źródłowy rozwiązania:

```
import java.util.Random;
import java.util.Arrays;

class Thr extends Thread {
    private int number;
    private int num_threads;

    private int[] vectorA;
    private int[] vectorB;
    private int[] vectorC;

    public Thr(int number, int num_threads, int[] vectorA, int[]
vectorB, int[] vectorC) {
      this.number = number;
      this.num_threads = num_threads;

      this.vectorA = vectorA;
      this.vectorB = vectorB;
      this.vectorC = vectorC;
    }

    public void run() {
      for (int i = number; i < vectorC.length; i +=
num_threads) {
            vectorC[i] = vectorA[i] + vectorB[i];
      }
    }
}

public class Ad1 {
```

```
    public static void randomFill(int[] vector) {
      Random random = new Random();

      for (int i = 0; i < vector.length; ++i) {
           vector[i] = random.nextInt(10);
      }
    }

    public static void main(String args[]) {
      int n = 15;
      int num_threads = 10;

      int vectorA[] = new int[n];
      int vectorB[] = new int[n];
      int vectorC[] = new int[n];

      randomFill(vectorA);
      randomFill(vectorB);

      System.out.println(Arrays.toString(vectorA));
      System.out.println("+");
      System.out.println(Arrays.toString(vectorB));
      System.out.println("=");

      Thr[] newThr = new Thr[num_threads];

      for (int i = 0; i < num_threads; ++i) {
           (newThr[i] = new Thr(i, num_threads, vectorA,
vectorB, vectorC)).start();
      }

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

      System.out.println(Arrays.toString(vectorC));
    }
}
```

Wydruk przykładowy programu:

```
[4, 7, 8, 5, 1, 6, 2, 8, 8, 0, 1, 0, 0, 5, 0]
+
```

```
[4, 7, 7, 6, 5, 2, 6, 8, 5, 0, 0, 5, 4, 8, 7]
=
[8, 14, 15, 11, 6, 8, 8, 16, 13, 0, 1, 5, 4, 13, 7]
```

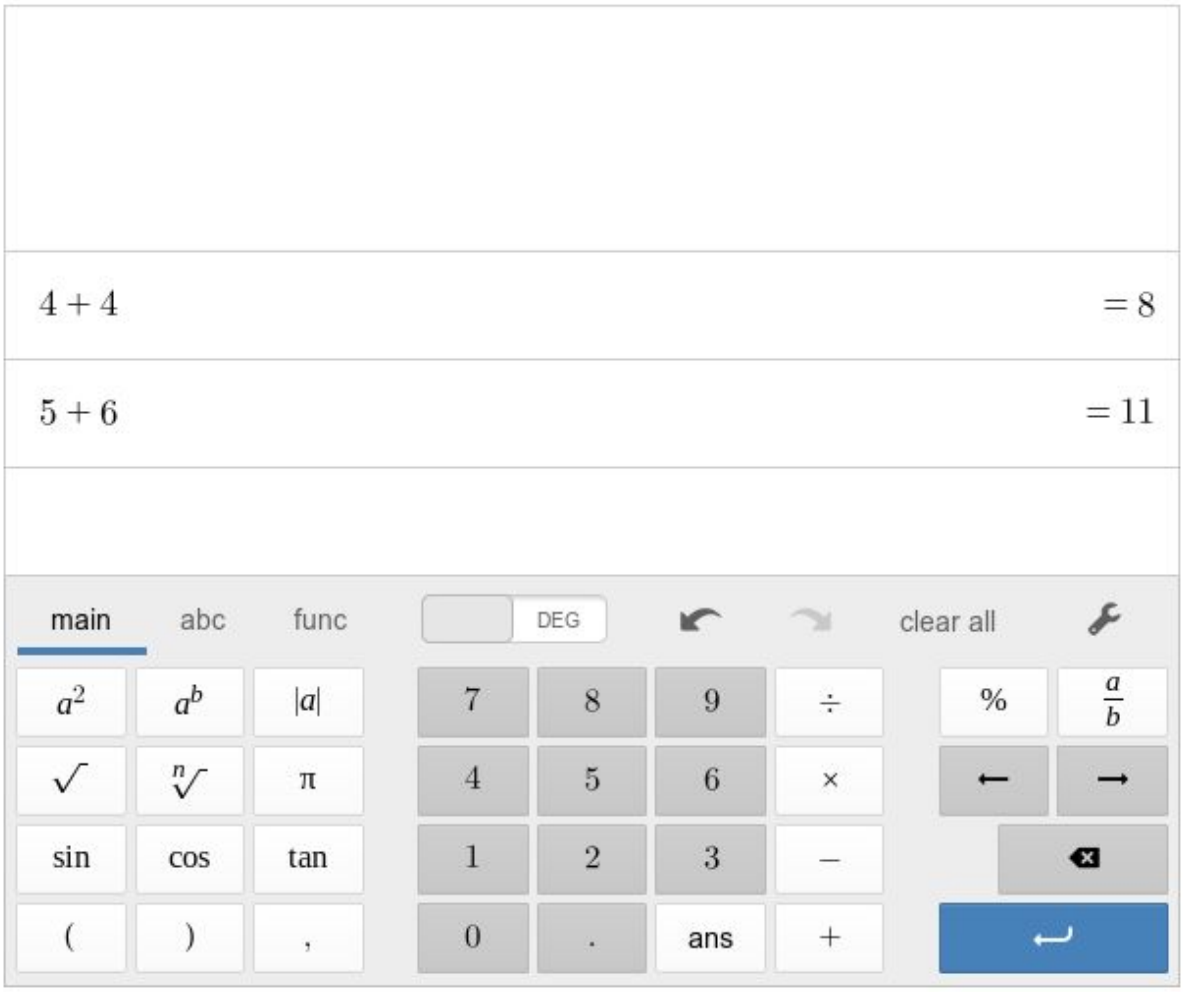
Algorytm przydziału zadań każdemu wątkowi jest cykliczny. Na przykład, dla 10 wątków i wektorów o długości 15 działa on tak:

	w1	w2	w3	w4	w5	w6	w7	w8	w9	w10
[0]	x									
[1]		x								
[2]			x							
[3]				x						
[4]					x					
[5]						x				
[6]							x			
[7]								x		
[8]									x	
[9]										x
[10]	x									
[11]		x								
[12]			x							
[13]				x						
[14]					x					

Dzięki temu algorytmowi, w przypadku gdy n nie jest podzielne przez liczbę wątków, obciążenie wątków zostaje zrównoważone.

Klasa Ad1{} jest klasą podstawową programu. Jej metoda randomFill() odpowiada za wypełnienie wektorów danymi losowymi. Jej metoda main() odpowiada testowanie wątków.
Klasa Thr{} i jej metoda run() odpowiadają za wykonanie kodu równoległego i przydział odpowiedzialności wątków.

Wyniki są poprawne ponieważ z przykładowego wydruku wyżej wynika że 4 + 4 = 8, a 5 + 6 = 11. Jest to zgodne z wynikami kalkulatora.



Zadanie 2

Zadanie brzmi: «Napisać program współbieżny wyliczający histogram dla obrazu o wymiarze n na m, przy użyciu wątków Javy. Dla uproszczenia niech obraz będzie dwuwymiarową tablicą zmiennych typu char.Każdy wątek powinien zliczać wystąpienie danego znaku w tablicy. Każdy wątek może obsługiwać kilka znaków.»

Kod źródłowy rozwiązania:

```
import java.util.Random;
import java.util.Arrays;

class Thr extends Thread {
    private int number;
    private int num_threads;
```

```
    private char[][] matrix;
    private int[] stats;

    public Thr(int number, int num_threads, char[][] matrix,
int[] stats) {
      this.number = number;
      this.num_threads = num_threads;

      this.matrix = matrix;
      this.stats = stats;
    }

    public void run() {
      for (char symbol = (char)(number + 33); symbol <
(char)(94 + 33); symbol += (char)num_threads) {
            for (int i = 0; i < matrix.length; ++i) {
                 for (int j = 0; j < matrix[i].length; ++j) {
                      if (matrix[i][j] == symbol) {
                           ++stats[(int)symbol - 33];
                      }
                 }
            }

            System.out.println("Wątek " + number + ":\t" +
symbol + " " + stats[(int)symbol - 33] + "x");
      }
    }
}

public class Ad2 {
    public static void randomFill(char[][] matrix) {
      Random random = new Random();

      for (int i = 0; i < matrix.length; ++i) {
            for (int j = 0; j < matrix[i].length; ++j) {
                 matrix[i][j] = (char)(random.nextInt(94) + 33);
            }
      }
    }

    public static void main(String args[]) {
      int n = 15;
      int m = 20;
      int stats[] = new int[94];
      int num_threads = 10;
```

```
      char matrix[][] = new char[n][m];

      randomFill(matrix);

      System.out.println("Calculating histogram for image:");

      for (int i = 0; i < matrix.length; ++i) {
           System.out.println(Arrays.toString(matrix[i]));
      }

      Thr[] newThr = new Thr[num_threads];

      for (int i = 0; i < num_threads; ++i) {
           (newThr[i] = new Thr(i, num_threads, matrix,
stats)).start();
      }

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

Wydruk przykładowy programu:

```
Calculating histogram for image:
[*, f, <, c, 0, g, ', ], U, g, E, L, q, U, 0, f, a, y, l, (]
[(, ,, L, ], x, W, #, X, ~, L, K, h, k, !, k, *, n, d, N, D]
[Q, H, 7, y, ', w, c, U, %, A, +, Q, o, ", \, p, Y, U, P, j]
[w, 7, J, =, ', |, =, F, l, %, Y, T, r, #, ~, H, $, [, 4, N]
[p, K, h, g, j, Q, i, 6, M, t, 5, c, ^, z, q, =, ~, T, y, 5]
[K, Q, /, 9, {, 6, 7, L, s, y, o, q, 4, :, 8, q, %, [, z, /]
[f, A, U, %, }, ), 0, f, [, ], V, $, o, ., <, W, e, ., t, r]
[:, #, ^, `, ~, r, (, f, {, @, G, [, B, B, 6, /, e, ), o, h]
[}, w, t, A, #, z, &, *, S, ^, r, s, ., i, /, s, h, H, D, Y]
[T, H, B, v, }, :, &, m, U, s, j, m, S, A, &, E, 6, h, w, `]
[(, 6, j, T, z, ), *, Z, &, w, U, A, 4, S, s, `, ^, z, &, *]
[A, _, C, 3, j, ", =, &, f, ', ', E, D, s, \, >, |, a, `, ;]
[I, e, ~, 3, Z, {, *, n, T, L, W, \, }, 7, z, l, &, W, %, s]
[&, ', U, >, `, G, L, d, `, 8, p, \, &, E, d, H, ,, Y, j, j]
```

```
[y, D, U, (, ~, T, y, ", ,, X, ;, G, f, z, 9, ], Z, c, >, ']
Wątek 1:    " 3x
Wątek 8:    ) 3x
Wątek 5:    & 9x
Wątek 2:    # 4x
Wątek 2:    - 0x
Wątek 2:    7 4x
Wątek 2:    A 6x
Wątek 2:    K 3x
Wątek 9:    * 6x
Wątek 0:    ! 1x
Wątek 9:    4 3x
Wątek 2:    U 9x
Wątek 5:    0 3x
Wątek 3:    $ 2x
Wątek 4:    % 5x
Wątek 8:    3 2x
Wątek 8:    = 4x
Wątek 6:    ' 7x
Wątek 7:    ( 5x
Wątek 1:    , 3x
Wątek 7:    2 0x
Wątek 6:    1 0x
Wątek 8:    G 3x
Wątek 4:    / 4x
Wątek 3:    . 3x
Wątek 5:    : 3x
Wątek 2:    _ 1x
Wątek 9:    > 3x
Wątek 0:    + 1x
Wątek 9:    H 5x
Wątek 2:    i 2x
Wątek 5:    D 4x
Wątek 3:    8 2x
Wątek 4:    9 2x
Wątek 8:    Q 4x
Wątek 6:    ; 2x
Wątek 6:    E 4x
Wątek 7:    < 2x
```

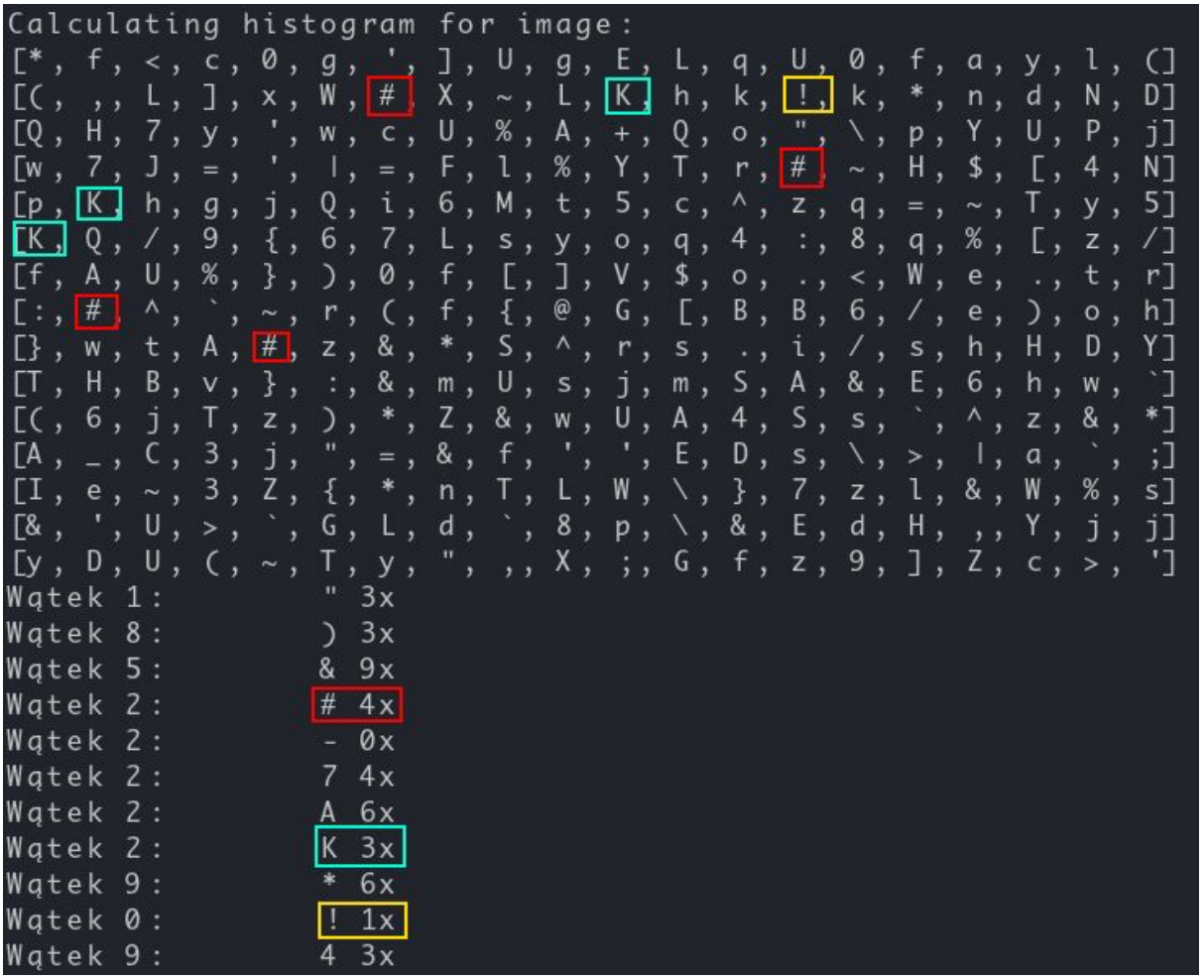
```
Wątek 1:    6 5x
Wątek 1:    @ 1x
Wątek 1:    J 1x
Wątek 7:    F 1x
Wątek 6:    O 0x
Wątek 8:    [ 4x
Wątek 4:    C 1x
Wątek 3:    B 3x
Wątek 5:    N 2x
Wątek 2:    s 7x
Wątek 9:    R 0x
Wątek 0:    5 2x
Wątek 9:    \ 4x
Wątek 2:    } 4x
Wątek 5:    X 2x
Wątek 3:    L 6x
Wątek 4:    M 1x
Wątek 8:    e 3x
Wątek 6:    Y 4x
Wątek 7:    P 1x
Wątek 1:    T 6x
Wątek 7:    Z 3x
Wątek 6:    c 4x
Wątek 8:    o 4x
Wątek 4:    W 4x
Wątek 3:    V 1x
Wątek 3:    ` 6x
Wątek 5:    b 0x
Wątek 5:    l 3x
Wątek 5:    v 1x
Wątek 9:    f 7x
Wątek 0:    ? 0x
Wątek 9:    p 3x
Wątek 3:    j 7x
Wątek 4:    a 2x
Wątek 8:    y 6x
Wątek 6:    m 2x
Wątek 7:    d 3x
Wątek 1:    ^ 4x
```

```
Wątek 7:    n 2x
Wątek 6:    w 5x
Wątek 4:    k 2x
Wątek 3:    t 3x
Wątek 9:    z 7x
Wątek 0:    I 1x
Wątek 3:    ~ 6x
Wątek 4:    u 0x
Wątek 7:    x 1x
Wątek 1:    h 5x
Wątek 0:    S 3x
Wątek 1:    r 4x
Wątek 0:    ] 4x
Wątek 1:    | 2x
Wątek 0:    g 3x
Wątek 0:    q 4x
Wątek 0:    { 3x
```

Algorytm przydziału zadań wątkom jest taki sam jak w poprzednim zadaniu, lecz przydziela on nie indeksy elementów w wektorze, a wyszukiwane przez wątek znaki.

Klasa Ad2{} jest klasą podstawową programu. Jej metoda randomFill() odpowiada za wypełnienie macierzy danymi losowymi. Jej metoda main() odpowiada testowanie wątków.
Klasa Thr{} i jej metoda run() odpowiadają za wykonanie kodu równoległego i przydział odpowiedzialności wątków.

Wyniki są poprawne ponieważ wyszukiwanie ręczne znaków w macierzy dało takie same wyniki:



Zadanie 3

Zadanie brzmi: «Napisz program znajdujący liczby pierwsze używając tzw. sita Erastotenesa.»

Kod źródłowy rozwiązania:

```
import java.util.Arrays;

class Thr extends Thread {
    private int start;
    private int finish;

    private int[] vector;

    public Thr(int start, int finish, int[] vector) {
      this.start = start;
      this.finish = finish;

      this.vector = vector;
```

```
    }

    public void run() {
      for (int i = 1; i < vector.length; ++i) {
           if (vector[i] == -1) {
                continue;
           }

           for (int j = start; j < finish; ++j) {
                if (vector[j] != vector[i] && vector[j] %
vector[i] == 0) {
                     vector[j] = -1;
                }
           }
      }
    }
}

public class Ad3 {
    public static void consecutiveFill(int[] vector) {
      for (int i = 0; i < vector.length; ++i) {
           vector[i] = i + 1;
      }
    }

    public static void main(String args[]) {
      int n = 100;
      int num_threads = 8;

      int vector[] = new int[n];

      consecutiveFill(vector);

      System.out.println("Vector before sieving:");
      System.out.println(Arrays.toString(vector) + "\n");

      Thr[] newThr = new Thr[num_threads];

      int start = 0;
      int extra = n % num_threads;

      for (int i = 0; i < num_threads; ++i) {
           int portion = n / num_threads;
           if (extra > 0) {
                ++portion;
```

```
                --extra;
            }

            System.out.println("wątek " + i + " szuka liczb w
przedziale " + start + "-" + (start + portion - 1));

            (newThr[i] = new Thr(start, start + portion,
vector)).start();

            start = start + portion;
      }

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

      System.out.println("\nVector after sieving:");

System.out.println(Arrays.toString(Arrays.stream(vector).filter(
x -> x > 0).toArray()));
    }
}
```

Wydruk programu:

```
Vector before sieving:
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66,
67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82,
83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98,
99, 100]

wątek 0 szuka liczb w przedziale 0-12
wątek 1 szuka liczb w przedziale 13-25
wątek 2 szuka liczb w przedziale 26-38
wątek 3 szuka liczb w przedziale 39-51
wątek 4 szuka liczb w przedziale 52-63
wątek 5 szuka liczb w przedziale 64-75
wątek 6 szuka liczb w przedziale 76-87
```

```
wątek 7 szuka liczb w przedziale 88-99

Vector after sieving:
[1, 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53,
59, 61, 67, 71, 73, 79, 83, 89, 97]
```

Algorytm przydziału zadań wątkom jest inny niż w poprzednich zadaniach, dlatego że w przykładzie do tego zadania jest pokazane że każdy wątek musi szukać liczb w sąsiednich częściach wektora. Na wynik to nie ma wpływu. Dla 4 wątków i wektora o długości 6 działa on tak:

	w1	w2	w3	w4
[0]	x			
[1]	x			
[2]		x		
[3]		x		
[4]			x	
[5]				x

Dla 4 wątków i wektora o długości 8 działa on tak:

	w1	w2	w3	w4
[0]	x			
[1]	x			
[2]		x		
[3]		x		
[4]			x	
[5]			x	
[6]				x
[7]				x

Dzięki temu algorytmowi, w przypadku gdy n nie jest podzielne przez liczbę wątków, obciążenie wątków zostaje zrównoważone.

Klasa Ad3{} jest klasą podstawową programu. Jej metoda consecutiveFill() odpowiada za wypełnienie wektora kolejnymi liczbami. Jej metoda main() odpowiada testowanie wątków i przydział odpowiedzialności wątków.
Klasa Thr{} i jej metoda run() odpowiadają za wykonanie kodu równoległego.

Wydruk jest prawie zgodny z ciągiem liczb pierwszych z Wikipedii. To że pierwszą liczbą w ciągu jest 1, nie jest zgodne z definicją liczb pierwszych, ale jest zgodne z wydrukiem przykładowym w postawionym zadaniu. Więc wynik jest poprawny.

Opisać najważniejsze elementy dotyczące wątków w javie i jak one działają

Dla stworzenia wątku w Javie najpierw trzeba stworzyć klasę dziedziczącą po klasie wbudowanej Thread{}, na przykład Thr{}, z kodem wykonywanym przez wątek w jej metodzie run().

Dalej, można stworzyć wątek.
Thr newThr = new Thr(...);

Aby uruchomić wątek:
newThr.start();

Żeby poczekać póki wątek skończy wykonywanie:
newThr.join();