

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Τμήμα Πληροφορικής



Εργασία Μαθήματος **Λειτουργικά Συστήματα**

|   |   |
|---|---|
| <b>Αριθμός εργασίας – Τίτλος εργασίας</b> | 1η Άσκηση – Διαδικεργασιακή Επικοινωνία |
| Όνομα φοιτητή                             | Δημήτρης Ματσαγγάνης                    |
| Αρ. Μητρώου                               | Π17068                                  |
| Ημερομηνία παράδοσης                      | 22/12/2018                              |



# Εκφώνηση ζητούμενης εργασίας

## Λειτουργικά Συστήματα

### Άσκηση – Διαδιεργασιακή Επικοινωνία

#### Α μέρος

Γράψτε ένα πρόγραμμα σε C ή σε Java το οποίο θα υλοποιεί τα παρακάτω:

1. Το πρόγραμμα θα ξεκινάει και θα δημιουργεί ένα πίνακα ακεραίων μεγέθους N θέσεων, όπου το N θα δίνεται ως είσοδος από τον χρήστη, έστω `array[N]`.
2. Στη συνέχεια το πρόγραμμα θα γεμίζει τον πίνακα με N ακεραίους αριθμούς, τυχαία επιλεγμένους από το διάστημα `[-100, 100]`.
3. Στη συνέχεια το πρόγραμμα θα δημιουργεί 4 νήματα (ή διεργασίες, ανάλογα με την υλοποίηση που επιλέξετε), καθένα από τα οποία θα κάνει τα εξής:
  - i. Το 1ο νήμα (ή 1η θυγατρική διεργασία) θα υπολογίζει το άθροισμα T1 των N στοιχείων:
  - ii. Το 2ο νήμα (ή 2η θυγατρική διεργασία) θα υπολογίζει το γινόμενο T2 των N στοιχείων:
  - iii. Το 3ο νήμα (ή 3η θυγατρική διεργασία) θα επιλέγει έναν τυχαίο αριθμό στο διάστημα `[-1000,1000]`.  
$$T3 = \text{random}(X) : -1000 \leq X \leq 1000$$
  - iv. Το 4ο νήμα (ή 4η θυγατρική διεργασία) θα “κατατάσσει” τα 3 προηγούμενα νήματα σε μία σειρά από το 1 μέχρι το 3, ανάλογα με τη σχέση των αποτελεσμάτων T1, T2, T3.

Π.χ. εάν  $T1=1200$ ,  $T2=-53521$  και  $T3 = 850$ , τότε η σειρά είναι: T1, T3, T2.



## **B μέρος**

Με βάση το πρόγραμμα που υλοποιήσατε παραπάνω, γράψτε μία τροποποιημένη εκδοχή του η οποία θα κάνει επιπλέον τα εξής:

1. Όταν κάθε ένα από τα 2 πρώτα νήματα ξεκινάει την εκτέλεσή του, θα επιλέγει με τυχαίο τρόπο τα μισά στοιχεία του πίνακα, και θα τα αλλάζει με νέους τυχαίους αριθμούς στο διάστημα  $[-100,100]$ . Στη συνέχεια θα υπολογίζει, ότι και πριν, το άθροισμα (το 1ο νήμα) ή το γινόμενο (το 2ο νήμα) του τροποποιημένου πίνακα.
2. Σε αυτήν την περίπτωση θα πρέπει να χρησιμοποιήσετε κάποια μέθοδο αμοιβαίου αποκλεισμού (π.χ. mutex/semaphores, κλήσεις sleep/weakup κτλ) ώστε να μην υπάρχει ποτέ περίπτωση να τροποποιούν και τα δύο νήματα ταυτόχρονα τα στοιχεία του κοινόχρηστου πίνακα.

Σε κάθε πρόγραμμα θα πρέπει να εμφανίζονται ενδεικτικά μηνύματα κάθε φορά που εκτελείται ένα νήμα, περιμένει κάποιο άλλο, ολοκληρώσει την εργασία του κτλ.

## **Οδηγίες:**

1. Η εργασία είναι **προαιρετική, ατομική** και ανήκει στην κατηγορία των bonus ασκήσεων.
2. Το πρόγραμμα μπορεί να υλοποιηθεί είτε σε γλώσσα C σε περιβάλλον linux, είτε σε Java σε οποιοδήποτε περιβάλλον.
3. Στην περίπτωση της υλοποίησης σε C μπορείτε να επιλέξετε στη λύση σας είτε τη χρήση θυγατρικών διεργασιών με τη χρήση της fork() είτε την χρήση νημάτων. Στην περίπτωση της υλοποίησης σε Java θα γίνει με τη χρήση νημάτων.
4. Η λύση μπορεί να υλοποιηθεί είτε σε ένα πρόγραμμα και για τα δύο μέρη, είτε σε δύο ξεχωριστά προγράμματα.
5. Το παραδοτέο θα περιλαμβάνει: (α) το έγγραφο με αναλυτική περιγραφή του κώδικα και ενδεικτικά screenshots από την εκτέλεση του προγράμματος. (β) τον πηγαίο κώδικα και (γ) το εκτελέσιμο αρχείο. Η παράδοση θα γίνει **μόνο μέσω του eclass σε .zip ή .rar αρχείο**.
6. Για τη συγγραφή της άσκησης θα χρησιμοποιήσετε το template που θα βρείτε στα έγγραφα του μαθήματος.



## ΠΕΡΙΕΧΟΜΕΝΑ

|     |  |    |
|-----|--|----|
| 1   | Εισαγωγή Εργασίας. ....  | 5  |
| 2   | Α΄ Μέρος. ....   | 5  |
| 2.1 | Βιβλιοθήκες, αρχικοποίηση μεταβλητών, εκτύπωση πίνακα ακεραίων μεγέθους n θέσεων. .... | 5  |
| 2.2 | Υλοποίηση των Threads μέσα στο κυρίως πρόγραμμα. ....                                  | 6  |
| 2.3 | Οι κλάσεις των Διεργασιών-Νημάτων (Threads).....                                       | 7  |
| 2.4 | Εικόνες από την εκτέλεση του προγράμματος.....   | 9  |
| 3   | Β΄ Μέρος.....  | 10 |
| 3.1 | Βιβλιοθήκες και Αρχικοποιήσεις μεταβλητών. ....  | 10 |
| 3.2 | Το κυρίως του προγράμματος.....  | 11 |
| 3.3 | Η κλάση Thread 1 (T1). ....  | 12 |
| 3.4 | Η κλάση Thread 2 (T2). ....  | 14 |
| 3.5 | Οι κλάσεις Thread 3 (T3) και Thread 4 (T4).....  | 15 |
| 3.6 | Εικόνες από την εκτέλεση του προγράμματος.....   | 16 |
| 4   | Βιβλιογραφικές Πηγές .....   | 17 |

## 1 ΕΙΣΑΓΩΓΗ ΕΡΓΑΣΙΑΣ.

Η παρούσα εργασία χωρίζεται σε δύο μέρη (το A\_Meros.java και το B\_Meros.java) και υλοποιεί όλα τα ζητούμενα που αναφέρονται παραπάνω, η επίλυση των οποίων θα περιγράφει αναλυτικά στις παρακάτω ενότητες.

Το πρόγραμμα υλοποιήθηκε σε γλώσσα προγραμματισμού Java με τη χρήση του εργαλείου Eclipse για την συγγραφή και την εκτέλεση του κώδικα και του κειμενογράφου Atom για την λήψη των εικόνων (screenshot) του κώδικα.

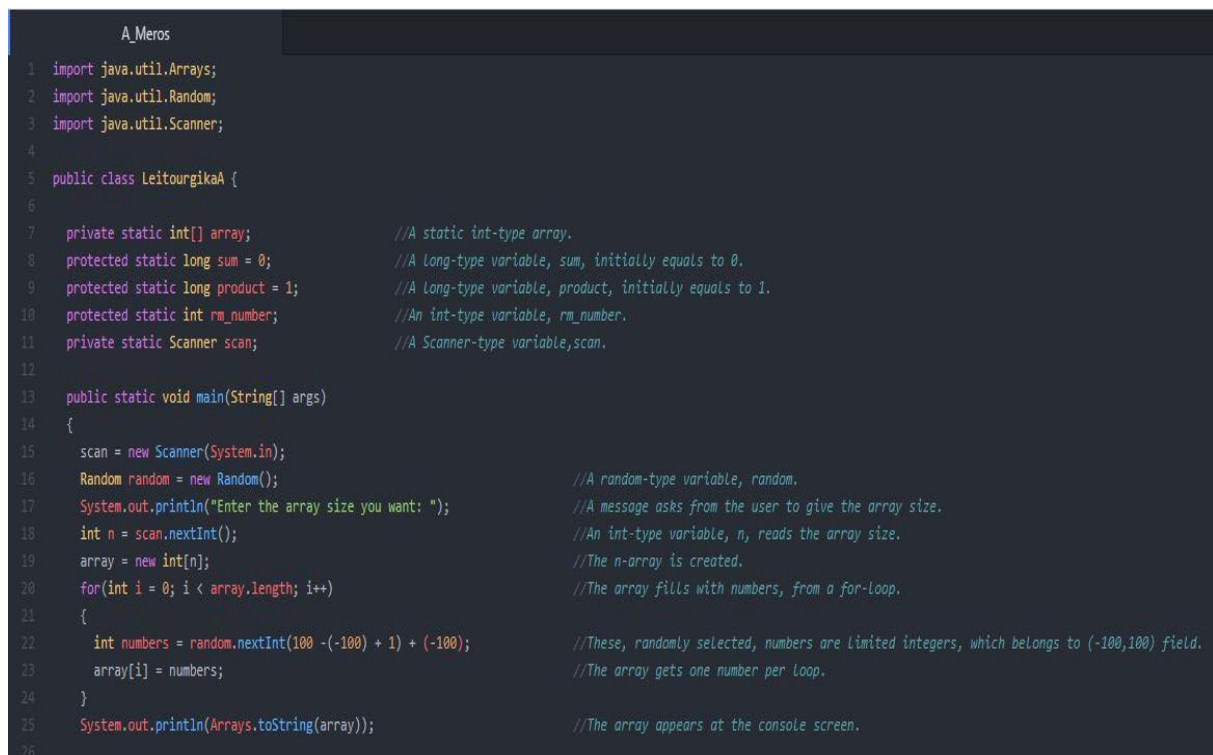
Στους αλγορίθμους επίλυσης της εργασίας έγινε χρήση γνώσεων και τεχνικών που παραδοθήκαν κατά τη διάρκεια του μαθήματος Λειτουργικά Συστήματα του 3<sup>ου</sup> Εξαμήνου.

## 2 Α΄ ΜΕΡΟΣ.

Σε αυτήν την ενότητα θα περιγράφει η αλγοριθμική λύση και η δομή του προγράμματός, και θα αναλυθεί ο κώδικας του Α΄ μέρους της παραπάνω εργασίας.

### 2.1 Βιβλιοθήκες, αρχικοποίηση μεταβλητών, εκτύπωση πίνακα ακεραιών μεγέθους η θέσεων.

Στο πρόγραμμα του Α΄ μέρους γίνεται χρήση των απαραίτητων βιβλιοθηκών, όπως φαίνεται και στην παρακάτω εικόνα.



```
1  import java.util.Arrays;
2  import java.util.Random;
3  import java.util.Scanner;
4
5  public class LeitourgikaA {
6
7      private static int[] array;           //A static int-type array.
8      protected static long sum = 0;        //A long-type variable, sum, initially equals to 0.
9      protected static long product = 1;    //A long-type variable, product, initially equals to 1.
10     protected static int rm_number;       //An int-type variable, rm_number.
11     private static Scanner scan;          //A Scanner-type variable, scan.
12
13     public static void main(String[] args)
14     {
15         scan = new Scanner(System.in);
16         Random random = new Random();     //A random-type variable, random.
17         System.out.println("Enter the array size you want: "); //A message asks from the user to give the array size.
18         int n = scan.nextInt();           //An int-type variable, n, reads the array size.
19         array = new int[n];               //The n-array is created.
20         for(int i = 0; i < array.length; i++) //The array fills with numbers, from a for-loop.
21         {
22             int numbers = random.nextInt(100 - (-100) + 1) + (-100); //These, randomly selected, numbers are limited integers, which belongs to (-100,100) field.
23             array[i] = numbers;           //The array gets one number per loop.
24         }
25         System.out.println(Arrays.toString(array)); //The array appears at the console screen.
26     }
```

Εικονά Κώδικα Α΄ Μέρους 1.

Ακόμη, για τις ανάγκες του προγράμματος αρχικοποιούνται : ένας πίνακας ακέραιων μεταβλητών(int-type variables), που ονομάζεται array, δύο μεταβλητές τύπου long, που



ονομάζονται `sum` και `product` και αρχικά είναι ίσες αντίστοιχα με το 0 και το 1, μιας ακέραιας (`int-type variable`) μεταβλητής, που ονομάζεται `gm_number` και μιας μεταβλητής τύπου `Scanner`, που ονομάζεται `scan` και μας βοηθάει στον ορισμό των θέσεων του πίνακα `array`, η οποία θα γίνεται από τον χρήστη.

Έπειτα, και μέσα στο κεντρικό πρόγραμμα ορίζεται μια τυχαία μεταβλητή (`Random-type variable`), που ονομάζεται `random`. Στη συνέχεια, θα εμφανίζεται ένα μήνυμα στην κονσόλα που θα ζητάει από τον χρήστη να ορίσει τις διαστάσεις του πίνακα (αριθμός θέσεων), αυτός ο ακέραιος αριθμός θα διαβάζεται ως «`n`» και έπειτα θα ορίζεται ο πίνακας `array`. Τα στοιχεία αυτού του πίνακα (δηλαδή του `array`) θα επιλέγονται τυχαία ένα-ένα μέσω ενός `for-loop` (ένα ανά επανάληψη) και η τιμή τους θα περιορίζεται από το -100 έως το 100. Τέλος, μετά τη λήξη του `for-loop` και όταν θα επιλεγθούν τόσα στοιχεία, όσες και οι θέσεις που επέλεξε νωρίτερα ο χρήστης τότε, αυτά τα στοιχεία θα εμφανίζονται στην οθόνη του χρήστη.

## 2.2 Υλοποίηση των Threads μέσα στο κυρίως πρόγραμμα.

Σε αυτήν την ενότητα θα περιγράψουν οι λειτουργίες των διεργασιών-νημάτων (`Threads`) μέσα στο κυρίως πρόγραμμα. Αρχικά, όλα τα `Threads` στο Α' Μέρος είναι αντικείμενα μιας ξεχωριστής κλάσης (τα οποία θα περιγράψουν παρακάτω). Τα τρία πρώτα `Thread`, `T1`, `T2` και `T3` εκτελούνται μέσω της κλήσης των αντικειμένων των κλάσεων τους που έχουν δημιουργηθεί εκτός της συνάρτησης `main`.

```
27 Thread T1 = new T1(); //A T1 class object, T1.
28 System.out.println("Starting Thread T1."); //Thread 1 (T1) start message appears at the console screen.
29 T1.start(); //Thread 1 (T1) executes.
30
31 Thread T2 = new T2(); //A T2 class object, T2.
32 System.out.println("Starting Thread T2."); //Thread 2 (T2) start message appears at the console screen.
33 T2.start(); //Thread 2 (T2) executes.
34
35 Thread T3 = new T3(); //A T3 class object, T3.
36 System.out.println("Starting Thread T3."); //Thread 3 (T3) start message appears at the console screen.
37 T3.start(); //Thread 3 (T3) executes.
38
39 Thread T4 = new T4(); //A T4 class object, T4.
40 System.out.println("Thread T4 will start when Threads T1, T2, T3 are completed."); //A Message appears at the control screen.
41 try
42 {
43     T1.join(); //Thread 4 waits for the completion of Thread T1.
44     T2.join(); //Thread 4 waits for the completion of Thread T2.
45     T3.join(); //Thread 4 waits for the completion of Thread T3.
46 }
47 catch (InterruptedException e)
48 {
49     throw new IllegalStateException(e); //Exception.
50 }
51 System.out.println("Starting Thread T4."); //Thread 4 (T4) start message appears at the console screen.
52 T4.start(); //Thread 4 (T4) executes.
53 }
54
```

Εικονά Κώδικα Α' Μέρους 2.



Ομοίως, καλείται και η τέταρτη και τελευταία διεργασία (Thread) T4, η οποία όμως σε αντίθεση με τις προηγούμενες, εκτελείται μόνο εάν οι τρεις υπόλοιπες διεργασίες έχουν ολοκληρωθεί, σε κάθε άλλη περίπτωση δεν θα ενεργοποιείται η τέταρτη διεργασία T4.

## 2.3 Οι κλάσεις των Διεργασιών-Νημάτων (Threads).

Σε αυτή την ενότητα θα περιγράφουν οι κλάσεις των Threads, των οποίων τα αντικείμενα (objects) έχουν χρησιμοποιηθεί προηγουμένως στη main του προγράμματος.

```
55 public static class T1 extends Thread{
56
57     public void run()                                //Run void.
58     {
59         System.out.println("Creating Thread T1.");    //Thread 1 (T1) is created and this appears at the console screen.
60         for (int i : array)                          //A for loop for all the numbers of the array.
61         {
62             sum += i ;                               //The last number is added to the others during every loop.
63         }
64         System.out.println("T1 = " + sum);           //Thread 1 (T1) appears at the console screen.
65         System.out.println("Exiting Thread T1.");    //Thread 1 (T1) exit message appears at the console screen.
66     }
67 }
68
69 public static class T2 extends Thread{
70
71     public void run()                                //Run void.
72     {
73         System.out.println("Creating Thread T2.");    //Thread 2 (T2) is created and this appears at the console screen.
74         for (int i : array)                          //A for loop for all the numbers of the array.
75         {
76             product *= i ;                          //The last number is multiplied to the others during every loop.
77         }
78         System.out.println("T2 = " + product);       //Thread 2 (T2) appears at the console screen.
79         System.out.println("Exiting Thread T2.");    //Thread 2 (T2) exit message appears at the console screen.
80     }
81 }
82
```

Εικονά Κώδικα Α' Μέρους 3.

Η πρώτη κλάση T1 που χρησιμοποιεί Threads έχει ως αντικείμενο της στη main το T1, θα υπολογίζει το άθροισμα των «n» στοιχείων μέσω ενός for-loop, το οποίο θα προσθέτει ένα-ένα τον κάθε αριθμό με το προηγούμενο άθροισμα, δηλαδή ο τελευταίος ανά επανάληψη αριθμός θα προσθέτεται στο άθροισμα όλων των προηγούμενων αριθμών, το οποίο αποθηκεύεται στη μεταβλητή τύπου long, sum και μετά το τέλος του for-loop θα εκτυπώνεται στην οθόνη του χρήστη όταν αυτό ζητηθεί από την main του προγράμματος.

Στη συνέχεια, η δεύτερη κλάση T2 που χρησιμοποιεί Threads και έχει ως αντικείμενο της στη main το T2, θα υπολογίζει το γινόμενο των «n» στοιχείων μέσω ενός for-loop, το οποίο θα πολλαπλασιάζει ένα-ένα τον κάθε αριθμό με το προηγούμενο γινόμενο με μια αντίστοιχη διαδικασία όπως η προηγούμενη του T1, δηλαδή ο τελευταίος ανά επανάληψη αριθμός θα πολλαπλασιάζεται στο γινόμενο όλων των προηγούμενων αριθμών, το οποίο αποθηκεύεται





στη μεταβλητή τύπου long, product και μετά το τέλος του for-loop θα είναι σε θέση να τα εκτυπώσει στην οθόνη του χρήστη όταν αυτό ζητηθεί.

Έπειτα, η τρίτη κλάση T3 που χρησιμοποιεί Thread, και έχει ως αντικείμενο στη main το T3, είναι μια τυχαία επιλογή ενός ακέραιου αριθμού από το -1000 έως το 1000, ο οποίος θα εκτυπώνεται στην οθόνη του χρήστη, το οποίο υλοποιείται με την βοήθεια μιας μεταβλητής τύπου Random, random.

```
83 v public static class T3 extends Thread{
84
85     public void run()                                //Run void.
86     {
87         System.out.println("Creating Thread T3.");    //Thread 3 (T3) is created and this appears at the console screen.
88         Random random = new Random();                //A random-type variable, random.
89         rm_number = random.nextInt(1000-(-1000)+1)+(-1000); //rm_number is a randomly selected limited integer, which belongs to (-1000,1000) field.
90         System.out.println("T3 = "+ rm_number);      //Thread 3 (T3) appears at the console screen.
91         System.out.println("Exiting Thread T3.");    //Thread 3 (T3) exit message appears at the console screen.
92     }
93 }
94
95 v public static class T4 extends Thread{
96
97     public void run()                                //Run void.
98     {
99         System.out.println("Creating Thread T4.");    //Thread 4 (T4) is created and this appears at the console screen.
100        long list[] = {sum,product,rm_number};        //A Long-type List array.
101
102        System.out.println("Numbers in Descending Order:"); //A message prints on the console screen.
103        for (int i = list.length-1; i >= 0; i--)      //A for Loop.
104        {
105            Arrays.sort(list);                        //The long-type array List is sorted in descending order.
106        }
107        System.out.println("1. " + list[2]);          //Thread 4 (T4) appears at the console screen with the highest number of the array, list.
108        System.out.println("2. " + list[1]);          //Thread 4 (T4) appears at the console screen with the second higher number of the array, list.
109        System.out.println("3. " + list[0]);          //Thread 4 (T4) appears at the console screen with the lowest number of the array, list.
110        System.out.println("Exiting Thread T4.");    //Thread 4 (T4) exit message appears at the console screen.
111    }
112 }
113 }
114
```

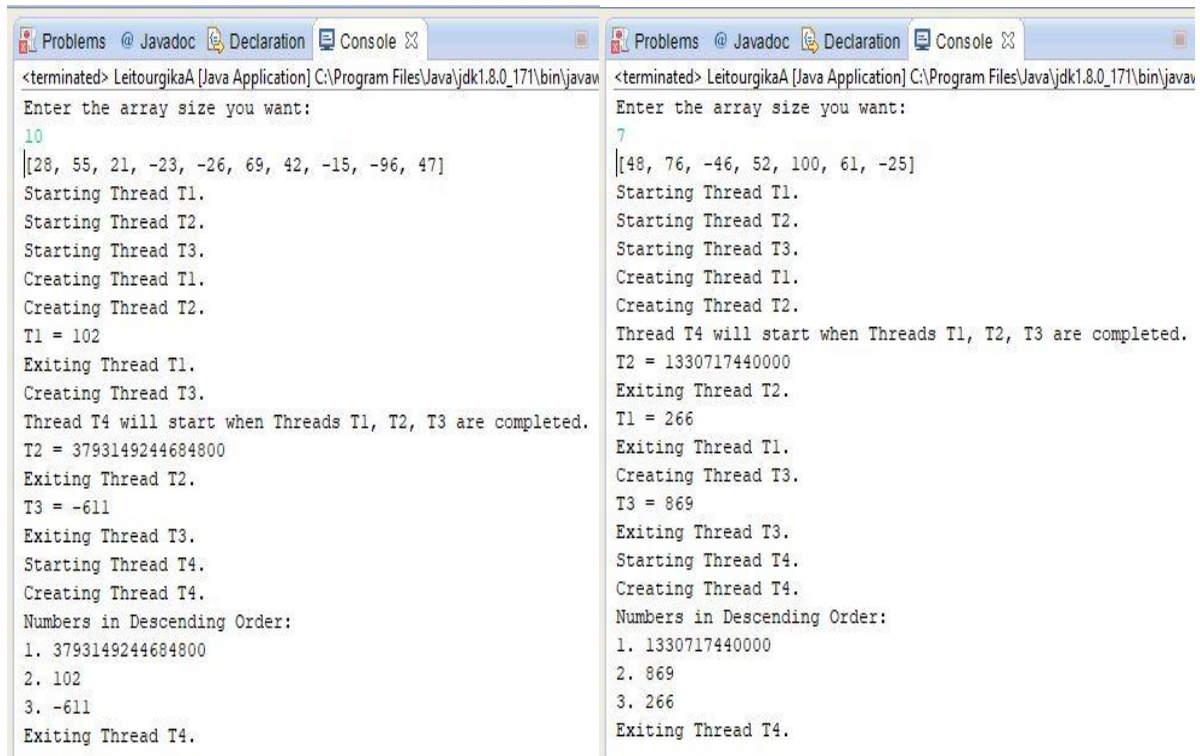
Εικονά Κώδικα Α΄ Μέρους 4.

Τέλος, η τέταρτη και τελευταία κλάση T4 που χρησιμοποιεί Thread, και έχει ως αντικείμενο στη main το T4, είναι μια λίστα μεταβλητών τύπου long που θα κατατάσσει τα προηγούμενα Threads (τα T1, T2 και T3) σε φθίνουσα σειρά και θα τα εμφανίζει στην οθόνη του χρήστη όταν αυτό ζητηθεί από τον τελευταίο.



## 2.4 Εικόνες από την εκτέλεση του προγράμματος.

Παρακάτω υπάρχουν εικόνες από τα αποτελέσματα του προγράμματος A\_Meros.java όταν αυτό εκτελέστηκε από το εργαλείο Eclipse.



```
<terminated> LeitourgikaA [Java Application] C:\Program Files\Java\jdk1.8.0_171\bin\javaw
Enter the array size you want:
10
[28, 55, 21, -23, -26, 69, 42, -15, -96, 47]
Starting Thread T1.
Starting Thread T2.
Starting Thread T3.
Creating Thread T1.
Creating Thread T2.
T1 = 102
Exiting Thread T1.
Creating Thread T3.
Thread T4 will start when Threads T1, T2, T3 are completed.
T2 = 3793149244684800
Exiting Thread T2.
T3 = -611
Exiting Thread T3.
Starting Thread T4.
Creating Thread T4.
Numbers in Descending Order:
1. 3793149244684800
2. 102
3. -611
Exiting Thread T4.
```

```
<terminated> LeitourgikaA [Java Application] C:\Program Files\Java\jdk1.8.0_171\bin\javaw
Enter the array size you want:
7
[48, 76, -46, 52, 100, 61, -25]
Starting Thread T1.
Starting Thread T2.
Starting Thread T3.
Creating Thread T1.
Creating Thread T2.
Thread T4 will start when Threads T1, T2, T3 are completed.
T2 = 1330717440000
Exiting Thread T2.
T1 = 266
Exiting Thread T1.
Creating Thread T3.
T3 = 869
Exiting Thread T3.
Starting Thread T4.
Creating Thread T4.
Numbers in Descending Order:
1. 1330717440000
2. 869
3. 266
Exiting Thread T4.
```

Εικόν Αποτελεσμάτων Α' Μέρους 1-2.

## 3 Β' ΜΕΡΟΣ.

Σε αυτήν την ενότητα θα περιγράψει η αλγοριθμική λύση και η δομή του προγράμματός, και θα αναλυθεί ο κώδικας του Β' μέρους της παραπάνω εργασίας.

### 3.1 Βιβλιοθήκες και Αρχικοποιήσεις μεταβλητών.

Στο πρόγραμμα του Β' μέρους γίνεται χρήση των απαραίτητων βιβλιοθηκών, όπως φαίνεται και στην παρακάτω εικόνα.

```

B_Meros
A_Meros

1 import java.util.ArrayList;
2 import java.util.Arrays;
3 import java.util.Scanner;
4 import java.util.concurrent.Semaphore;
5 import java.util.Random;
6
7 public class LeitourgikaB {
8     private static int[] array;           //A static int-type array.
9     protected static int rm_number;       //An int-type variable, rm_number.
10    protected static long sum = 0;        //A long-type variable, sum, initially equals to 0.
11    protected static long product = 1;    //A long-type variable, product, initially equals to 1.
12    static ArrayList<Integer> used1 = new ArrayList<Integer>(); //An static array list for used numbers.
13    static ArrayList<Integer> used2 = new ArrayList<Integer>(); //An static array list for used numbers.
14    static Semaphore mutex = new Semaphore(1); //Semaphore methon for 1 Thread.
15    private static Scanner scan;          //A Scanner-type variable, scan.
16
17
```

Εικονά Κώδικα Β' Μέρους 1.

Επιπλέον, για τις ανάγκες του προγράμματος B\_Meros.java αρχικοποιούνται : ένας πίνακας ακέραιων μεταβλητών (int-type variables), που ονομάζεται array, δύο μεταβλητές τύπου long, που ονομάζονται sum και product και αρχικά είναι ίσες αντίστοιχα με το 0 και το 1, μιας ακέραιας (int-type variable) μεταβλητής, που ονομάζεται rm\_number μιας μεταβλητής τύπου Scanner, που ονομάζεται scan και μας βοηθάει στον ορισμό των θέσεων του πίνακα array, η οποία θα γίνεται από τον χρήστη, ακριβώς όπως στο προηγούμενο μέρος της εργασίας. Επιπρόσθετα σε αυτό το μέρος, χρησιμοποιούνται άλλες 2 λίστες με ονόματα used1 και used2, οι οποίες χρησιμοποιούνται για την αναγνώριση των ήδη χρησιμοποιημένων αριθμών των Threads 1 και 2 αντίστοιχα, και η μέθοδος Semaphore Mutex η οποία θα ξεκινάει μόνο το ένα από τα δύο Threads, με τυχαία επιλογή και το άλλο μόνο όταν η διεργασία του πρώτου (κατά επιλογή πρώτου) έχει ολοκληρωθεί.

### 3.2 Το κυρίως του προγράμματος.

Στη συνέχεια του προγράμματος έρχεται η main, όπου καλούνται τα Threads που θα αναλυθούν σε παρακάτω ενότητες.

```
18 public static void main(String[] args)
19 {
20     scan = new Scanner(System.in);
21     Random random = new Random(); //A random-type variable, random.
22     System.out.println("Enter the array size you want: "); //A message asks from the user to give the array size.
23     int n = scan.nextInt(); //An int-type variable, n, reads the array size.
24     array = new int[n]; //The n-array is created.
25
26     for(int i = 0; i < array.length; i++) //The array fills with numbers, from a for-loop.
27     {
28         int numbers = random.nextInt(100 - (-100) + 1) + (-100); //These numbers are limited integers, which belongs to (-100,100) field.
29         array[i] = numbers; //The array gets one number per loop.
30     }
31     System.out.println(Arrays.toString(array)); //The array appears at the console screen.
32
33     Thread T1 = new T1(); //A T1 class object, T1.
34     T1.start(); //Thread 1 (T1) executes.
35
36     Thread T2 = new T2(); //A T2 class object, T2.
37     T2.start(); //Thread 2 (T2) executes.
38
39     Thread T3 = new T3(); //A T3 class object, T3.
40     System.out.println("Starting Thread T3."); //Thread 3 (T3) start message appears at the console screen.
41     T3.start(); //Thread 3 (T3) executes.
42
43     Thread T4 = new T4(); //A T4 class object, T4.
44     System.out.println("Thread T4 will start when Threads T1, T2, T3 are completed."); //A Message appears at the control screen.
45     try
46     {
47         T1.join(); //Thread 4 waits for the completion of Thread T1.
48         T2.join(); //Thread 4 waits for the completion of Thread T2.
49         T3.join(); //Thread 4 waits for the completion of Thread T3.
50     }
51     catch (InterruptedException e)
52     {
53         throw new IllegalStateException(e); //Exception.
54     }
55     System.out.println("Starting Thread T4."); //Thread 4 (T4) start message appears at the console screen.
56     T4.start(); //Thread 4 (T4) executes.
57 }
```

Εικόνα Κώδικα Β' Μέρους 2.

Αρχικά, μέσα στο κεντρικό πρόγραμμα ορίζεται μια τυχαία μεταβλητή (Random-type variable), που ονομάζεται random. Έπειτα, θα εμφανίζεται ένα μήνυμα στην κονσόλα που θα ζητάει από τον χρήστη να ορίσει τις διαστάσεις του πίνακα (αριθμός θέσεων), αυτός ο ακέραιος αριθμός θα διαβάζεται ως «n» και έπειτα θα ορίζεται ο πίνακας array. Τα στοιχεία αυτού του πίνακα (δηλαδή του array) θα επιλέγονται τυχαία ένα-ένα μέσω ενός for-loop (ένα ανά επανάληψη) και η τιμή τους θα περιορίζεται από το -100 έως το 100. Τέλος, μετά τη λήξη του for-loop και όταν θα επιλεγθούν τόσα στοιχεία, όσες και οι θέσεις που επέλεξε νωρίτερα ο χρήστης τότε, αυτά τα στοιχεία θα εμφανίζονται στην οθόνη του χρήστη και θα αποτελούν τον πίνακα array.

Ακόμη, τα Threads στο Β' Μέρος, όπως αναφέρθηκε και προηγουμένως είναι αντικείμενα μιας ξεχωριστής κλάσης (τα οποία θα περιγράφουν παρακάτω). Τα τρία πρώτα Thread, T1, T2 και T3 εκτελούνται μέσω της κλήσης των αντικειμένων των κλάσεων τους που έχουν δημιουργηθεί εκτός της συνάρτησης main και ομοίως, καλείται και η τέταρτη και τελευταία



διεργασία (Thread) T4, η οποία όμως σε αντίθεση με τις προηγούμενες, εκτελείται μόνο εάν οι τρεις υπόλοιπες διεργασίες έχουν ολοκληρωθεί, ενώ σε κάθε άλλη περίπτωση δεν θα ενεργοποιείται η τέταρτη διεργασία T4.

### 3.3 Η κλάση Thread 1 (T1).

Σε αυτήν την ενότητα θα γίνει η περιγραφή της κλάσης T1 που χρησιμοποιεί διεργασίες (Threads) και έχει ως αντικείμενο της το T1, το οποίο αναφέρθηκε προηγουμένως.

```
69 public static class T1 extends Thread{
70
71     public void run()
72     {
73         try
74         {
75             mutex.acquire();                                //Mutex acquire Thread 1 (T1).
76             System.out.println("Thread 1 got the permit.");    //Permit Message for Thread 1 (T1) appears.
77
78             try
79             {
80                 System.out.println("Starting Thread T1.");    //Thread 1 (T1) start message appears at the console screen.
81                 Random random = new Random();                //A random-type variable, random.
82                 boolean new_numbers = true;                  //A boolean-type variable, new_numbers
83                 int count = 0;                                //An int-type variable, count.
84
85                 while (new_numbers)                          //While new_numbers is true.
86                 {
87                     int randomNumber = random.nextInt(array.length);    //A random number from the array is selected.
88                     int numbers = random.nextInt(100 - (-100) + 1) + (-100);    //These, randomly selected, numbers are limited integers, which belongs to (-100,100) field.
89
90                     if (!used1.contains(randomNumber))        //The new number adds to the array when is not duplicate.
91                     {
92                         used1.add(randomNumber);                //Adds to an array list the array's spot where number will change.
93                         count++;                                //count = count + 1.
94                         array[randomNumber] = numbers;          //Random spot from the array changes with the randomly selected number.
95                     }
96
97                     if (count >= array.length/2)              //If count is over the half of the elements of the array.
98                     {
99                         new_numbers = false;                  //Then new_numbers is false.
100                     }
101                 }
102                 System.out.println("Thread 1: " + Arrays.toString(array));    //The new array is printed at the screen.
103                 System.out.println("Thread 1 changed spots: " + used1);    //The array's spots which changed, appears at the console screen.
104
105                 for (int i : array)                            //A for Loop for all the numbers of the array.
106                 {
107                     sum += i ;                                //The last number is added to the others during every loop.
108                 }
109             }
110             finally
111             {
112                 System.out.println("T1 = " + sum);            //Thread 1 (T1) appears at the console screen.
113                 System.out.println("Exiting Thread T1.");    //Thread 1 (T1) exit message appears at the console screen.
114                 System.out.println("Thread 1 releasing lock.");    //Thread 1 (T1) releases Lock message appears at the console screen.
115                 mutex.release();                                //Mutex release Thread T1 after T1's process is completed and now is empty.
116             }
117         }
118         catch (InterruptedException e)
119         {
120             e.printStackTrace();                                //Exception.
121         }
122     }
123 }
```

Εικόνα Κώδικα Β' Μέρους 3.

Αρχικά, μέσω ενός try όταν επιλέγεται από τη μέθοδο mutex τότε θα εμφανίζεται μήνυμα στο χρήστη που, θα τον ενημερώνει ότι επιλέχθηκε η παραπάνω μέθοδος. Έπειτα, θα εμφανίζεται άλλο ένα μήνυμα στο χρήστη, ότι ξεκινάει η διεργασία ενώ ακόμη, ορίζονται μια μεταβλητή τύπου Random, random, μια μεταβλητή τύπου Boolean, new\_numbers, η οποία



αρχικά είναι αληθής (true) και μια ακέραια μεταβλητή (int-type variable), count, η οποία αρχικά είναι ίση με το μηδέν («0»).

Έπειτα, και ενώ (δηλαδή η μέθοδος while) η μεταβλητή τύπου Boolean, new\_numbers είναι αληθής (true), τότε ορίζονται άλλες 2 μεταβλητές, μια ακέραια μεταβλητή random Numbers, η οποία επιλεγεί κάθε φορά μια θέση μέσα στην λίστα array και μια ακέραια μεταβλητή numbers, η οποία επιλεγεί τυχαία αριθμούς που η τιμή τους θα περιορίζεται από το -100 έως το 100. Εάν ο τυχαίος αριθμός δεν περιέχεται στη λίστα των χρησιμοποιημένων used1, τότε ο αριθμός που βρίσκεται σε εκείνη τη θέση θα αλλάζει, ο αριθμός θα προστίθεται στη λίστα των χρησιμοποιημένων αριθμών used1 και η ακέραια μεταβλητή count θα αυξάνεται κατά ένα. Ακόμη, όταν αλλάξουν οι μισές θέσεις του πίνακα ( $\text{count} \geq \text{array\_length}/2$ ) τότε η διαδικασία θα σταματήσει με την απενεργοποίηση (θα γίνει false) της Boolean μεταβλητής new\_numbers. Υστέρα, θα εμφανίζεται στην οθόνη ο νέος πίνακας καθώς και οι θέσεις του πίνακα, όπου αλλάξαν οι τιμές τους. Στη περίπτωση που η παρούσα διεργασία επιλεγεί να εκκινήσει δεύτερη από τη μέθοδο mutex τότε, ο πίνακας που θα μεταβληθεί θα ναι αυτός που, έχει δημιουργηθεί από την προηγούμενη διεργασία και όχι ο αρχικός.

Τέλος, θα υπολογίζει το άθροισμα των «n» στοιχείων μέσω ενός for-loop, το οποίο θα προσθέτει ένα-ένα τον κάθε αριθμό με το προηγούμενο άθροισμα, δηλαδή ο τελευταίος ανά επανάληψη αριθμός θα προσθέτεται στο άθροισμα όλων των προηγούμενων αριθμών, το οποίο αποθηκεύεται στη μεταβλητή τύπου long, sum και μετά το τέλος του for-loop θα εκτυπώνεται στην οθόνη του χρήστη μαζί με το μήνυμα εξόδου της διεργασίας T1 και το μήνυμα ότι η μέθοδος mutex θα είναι σε θέση να δεκτή μια άλλη διεργασία, αφού η διεργασία T1 ολοκληρώθηκε και μπορεί πλέον να αποδεσμευτεί.



### 3.4 Η κλάση Thread 2 (T2).

Σε αυτήν την ενότητα θα γίνει η περιγραφή της κλάσης T2 που χρησιμοποιεί διεργασίες (Threads) και έχει ως αντικείμενο της το T2, το οποίο αναφέρθηκε προηγουμένως.

```
115 public static class T2 extends Thread{
116
117     public void run()                //Run void.
118     {
119         try
120         {
121             mutex.acquire();          //Mutex acquire Thread 2 (T2).
122             System.out.println("Thread 2 got the permit."); //Permit Message for Thread 2 (T2) appears.
123
124             try
125             {
126                 System.out.println("Starting Thread T2."); //Thread 2 (T2) start message appears at the console screen.
127                 Random random = new Random();              //A random-type variable, random.
128                 boolean new_numbers = true;                //A boolean-type variable, new_numbers
129                 int count = 0;                             //An int-type variable, count.
130
131                 while (new_numbers)                        //While new_numbers is true.
132                 {
133                     int randomNumber = random.nextInt(array.length); //A random number from the array is selected.
134                     int numbers = random.nextInt(100 - (-100) + 1) + (-100); //These, randomly selected, numbers are limited integers, which belongs to (-100,100) field.
135
136                     if (!used2.contains(randomNumber))      //The new number adds to the array when is not duplicate.
137                     {
138                         used2.add(randomNumber);            //Adds to an array list the array's spot where number will change.
139                         count++;                             //count = count + 1.
140                         array[randomNumber] = numbers;      //Random spot from the array changes with the randomly selected number.
141                     }
142
143                     if (count >= array.length/2)           //If count is over the half of the elements of the array.
144                     {
145                         new_numbers = false;               //Then new_numbers is false.
146                     }
147
148                     System.out.println("Thread 2: " + Arrays.toString(array)); //The new array is printed at the screen.
149                     System.out.println("Thread 2 changed spots: " + used2);    //The array's spots which changed, appears at the console screen.
150
151                     for (int i : array)                    //A for Loop for all the numbers of the array.
152                     {
153                         product *= i;                      //The Last number is multiplied to the others during every loop.
154                     }
155                 }
156             } finally
157             {
158                 System.out.println("T2 = " + product);     //Thread 2 (T2) appears at the console screen.
159                 System.out.println("Exiting Thread T2.");  //Thread 2 (T2) exit message appears at the console screen.
160                 System.out.println("Thread 2 releasing lock."); //Thread 2 (T2) release message appears at the console screen.
161                 mutex.release();                            //Mutex release Thread T2 after T2's process is completed and now is empty.
162             }
163         }
164         catch (InterruptedException e)
165         {
166             e.printStackTrace();                          //Exception.
167         }
168     }
169 }
```

Εικόνα Κώδικα Β' Μέρους 4.

Καταρχάς, μέσω ενός try όταν επιλέγεται από τη μέθοδο mutex τότε θα εμφανίζεται μήνυμα στο χρήστη που, θα τον ενημερώνει ότι επιλέχθηκε η παραπάνω μέθοδος. Έπειτα, θα εμφανίζεται άλλο ένα μήνυμα στο χρήστη, ότι ξεκινάει η διεργασία ενώ ακόμη, ορίζονται μια μεταβλητή τύπου Random, random, μια μεταβλητή τύπου Boolean, new\_numbers, η οποία αρχικά είναι αληθής (true) και μια ακέραια μεταβλητή (int-type variable), count, η οποία αρχικά είναι ίση με το μηδέν («0»).

Έπειτα, και ενώ (δηλαδή η μέθοδος while) η μεταβλητή τύπου Boolean, new\_numbers είναι αληθής (true), τότε ορίζονται άλλες 2 μεταβλητές, μια ακέραια μεταβλητή random Numbers, η οποία επιλεγεί κάθε φορά μια θέση μέσα στην λίστα array και μια ακέραια μεταβλητή





numbers, η οποία επιλεγεί τυχαία αριθμούς που η τιμή τους θα περιορίζεται από το -100 έως το 100. Εάν ο τυχαίος αριθμός δεν περιέχεται στη λίστα των χρησιμοποιημένων used2, τότε ο αριθμός που βρίσκεται σε εκείνη τη θέση θα αλλάζει, ο αριθμός θα προστίθεται στη λίστα των χρησιμοποιημένων αριθμών used2 και η ακέραια μεταβλητή count θα αυξάνεται κατά ένα. Ακόμη, όταν αλλάξουν οι μισές θέσεις του πίνακα ( $count \geq array.length/2$ ) τότε η διαδικασία θα σταματήσει με την απενεργοποίηση (θα γίνει false) της Boolean μεταβλητής new\_numbers. Υστέρα, θα εμφανίζεται στην οθόνη ο νέος πίνακας καθώς και οι θέσεις του πίνακα, όπου αλλάξαν οι τιμές τους. Στη περίπτωση που η παρούσα διεργασία επιλεγεί να εκκινήσει δεύτερη από τη μέθοδο mutex τότε, ο πίνακας που θα μεταβληθεί θα ναι αυτός που, έχει δημιουργηθεί από την προηγούμενη διεργασία και όχι ο αρχικός.

Τέλος, θα υπολογίζει το γινόμενο των «n» στοιχείων μέσω ενός for-loop, το οποίο θα πολλαπλασιάζει ένα-ένα τον κάθε αριθμό με το προηγούμενο γινόμενο με μια αντίστοιχη διαδικασία όπως η προηγούμενη του T1, δηλαδή ο τελευταίος ανά επανάληψη αριθμός θα πολλαπλασιάζεται στο γινόμενο όλων των προηγούμενων αριθμών, το οποίο αποθηκεύεται στη μεταβλητή τύπου long, product και μετά το τέλος του for-loop θα εκτυπώνεται στην οθόνη του χρήστη μαζί με το μήνυμα εξόδου της διεργασίας T2 και το μήνυμα ότι η μέθοδος mutex θα είναι σε θέση να δεκτή μια άλλη διεργασία, αφού η διεργασία T2 ολοκληρώθηκε και μπορεί πλέον να αποδεσμευτεί.

### 3.5 Οι κλάσεις Thread 3 (T3) και Thread 4 (T4).

Σε αυτήν την ενότητα θα περιγράφουν οι κλάσεις T3 και T4.

```
171 ~ public static class T3 extends Thread{
172
173     public void run()                                //Run void.
174 ~ {
175         System.out.println("Creating Thread T3.");    //Thread 3 (T3) is created and this appears at the console screen.
176         Random random = new Random();                //A random-type variable, random.
177 ~         rm_number = random.nextInt(1000-(-1000)+1)+(-1000); //rm_number is a randomly selected limited integer, which belongs to (-1000,1000) field.
178         System.out.println("T3 = "+ rm_number);      //Thread 3 (T3) appears at the console screen.
179         System.out.println("Exiting Thread T3.");    //Thread 3 (T3) exit message appears at the console screen.
180     }
181 }
182
183 ~ public static class T4 extends Thread{
184
185     public void run()                                //Run void.
186 ~ {
187         System.out.println("Creating Thread T4.");    //Thread 4 (T4) is created and this appears at the console screen.
188 ~         long list[] = {sum,product,rm_number};      //A long-type List array.
189
190         System.out.println("Numbers in Descending Order:"); //A message prints on the console screen.
191 ~         for (int i = list.length-1; i >= 0; i--)    //A for loop.
192 ~         {
193             Arrays.sort(list);                      //The long-type array List is sorted in descending order.
194         }
195 ~         System.out.println("1. " + list[2]);        //Thread 4 (T4) appears at the console screen with the highest number of the array, List.
196         System.out.println("2. " + list[1]);        //Thread 4 (T4) appears at the console screen with the second higher number of the array, List.
197         System.out.println("3. " + list[0]);        //Thread 4 (T4) appears at the console screen with the lowest number of the array, List.
198         System.out.println("Exiting Thread T4.");    //Thread 4 (T4) exit message appears at the console screen.
199     }
200 }
201 }
```

Εικόνα Κώδικα Β' Μέρους 5.

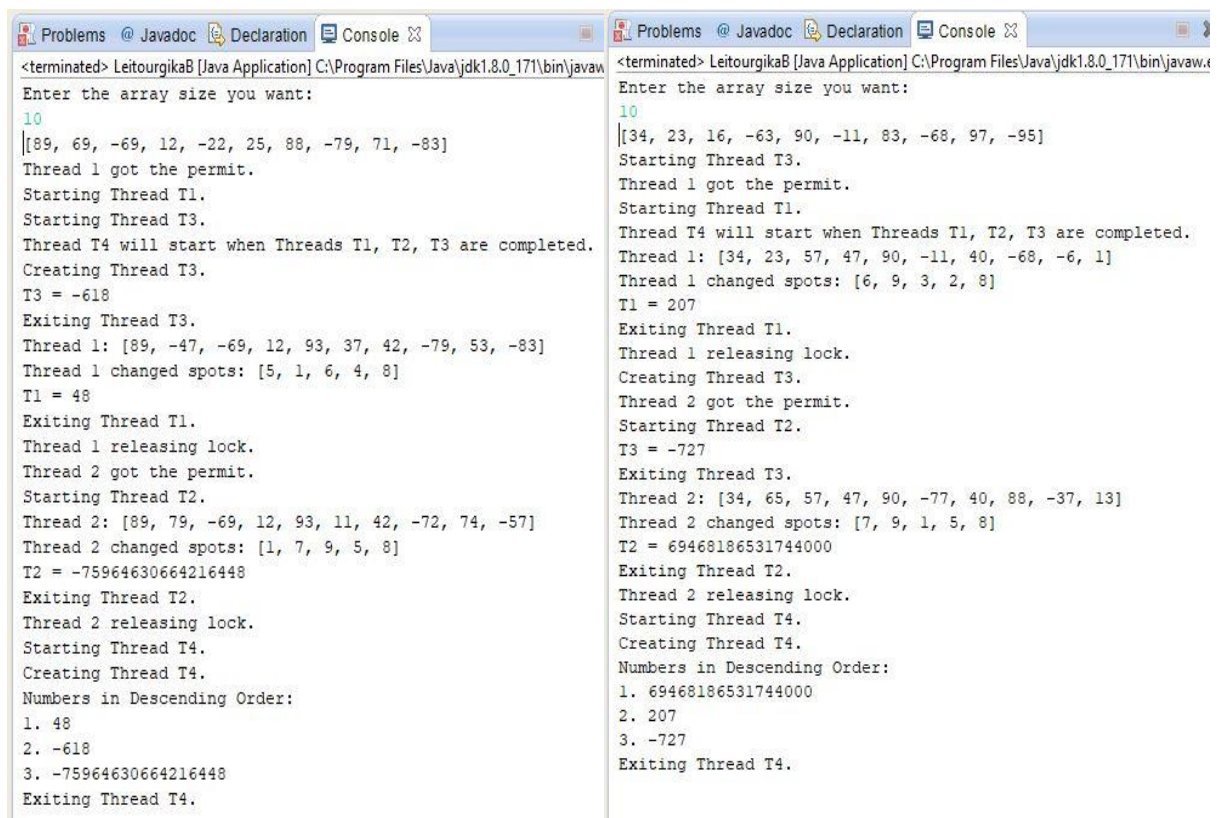
Αρχικά, η τρίτη κλάση T3 που χρησιμοποιεί Thread, και έχει ως αντικείμενο στη main το T3, είναι μια τυχαία επιλογή ενός ακέραιου αριθμού από το -1000 έως το 1000, ο οποίος

θα εκτυπώνεται στην οθόνη του χρήστη, το οποίο υλοποιείται με την βοήθεια μιας μεταβλητής τύπου Random, random.

Τέλος, η τέταρτη και τελευταία κλάση T4 που χρησιμοποιεί Thread, και έχει ως αντικείμενο στη main το T4, είναι μια λίστα μεταβλητών τύπου long που θα κατατάσσει τα προηγούμενα Threads (τα T1, T2 και T3) σε φθίνουσα σειρά και θα τα εμφανίζει στην οθόνη του χρήστη όταν αυτό ζητηθεί από τον τελευταίο.

### 3.6 Εικόνες από την εκτέλεση του προγράμματος.

Παρακάτω υπάρχουν εικόνες από τα αποτελέσματα του προγράμματος B\_Meros.java όταν αυτό εκτελέστηκε από το εργαλείο Eclipse.



```
<terminated> LeitourgikaB [Java Application] C:\Program Files\Java\jdk1.8.0_171\bin\javaw.exe
Enter the array size you want:
10
[89, 69, -69, 12, -22, 25, 88, -79, 71, -83]
Thread 1 got the permit.
Starting Thread T1.
Starting Thread T3.
Thread T4 will start when Threads T1, T2, T3 are completed.
Creating Thread T3.
T3 = -618
Exiting Thread T3.
Thread 1: [89, -47, -69, 12, 93, 37, 42, -79, 53, -83]
Thread 1 changed spots: [5, 1, 6, 4, 8]
T1 = 48
Exiting Thread T1.
Thread 1 releasing lock.
Thread 2 got the permit.
Starting Thread T2.
Thread 2: [89, 79, -69, 12, 93, 11, 42, -72, 74, -57]
Thread 2 changed spots: [1, 7, 9, 5, 8]
T2 = -75964630664216448
Exiting Thread T2.
Thread 2 releasing lock.
Starting Thread T4.
Creating Thread T4.
Numbers in Descending Order:
1. 48
2. -618
3. -75964630664216448
Exiting Thread T4.

<terminated> LeitourgikaB [Java Application] C:\Program Files\Java\jdk1.8.0_171\bin\javaw.exe
Enter the array size you want:
10
[34, 23, 16, -63, 90, -11, 83, -68, 97, -95]
Starting Thread T3.
Thread 1 got the permit.
Starting Thread T1.
Thread T4 will start when Threads T1, T2, T3 are completed.
Thread 1: [34, 23, 57, 47, 90, -11, 40, -68, -6, 1]
Thread 1 changed spots: [6, 9, 3, 2, 8]
T1 = 207
Exiting Thread T1.
Thread 1 releasing lock.
Creating Thread T3.
Thread 2 got the permit.
Starting Thread T2.
T3 = -727
Exiting Thread T3.
Thread 2: [34, 65, 57, 47, 90, -77, 40, 88, -37, 13]
Thread 2 changed spots: [7, 9, 1, 5, 8]
T2 = 69468186531744000
Exiting Thread T2.
Thread 2 releasing lock.
Starting Thread T4.
Creating Thread T4.
Numbers in Descending Order:
1. 69468186531744000
2. 207
3. -727
Exiting Thread T4.
```

Εικόνες Αποτελεσμάτων 3-4.



## 4 ΒΙΒΛΙΟΓΡΑΦΙΚΕΣ ΠΗΓΕΣ

---

Για την επίλυση του προβλήματος και για τη συγγραφή της παρούσας εργασίας χρησιμοποιήθηκαν οι παρακάτω πηγές:

1. Σύγχρονα Λειτουργικά Συστήματα (Modern Operating Systems) A.S.Tanenbaum, Prentice Hall 2008, 4th Edition Ελληνική Μετάφραση, 2009, Εκδόσεις Κλειδάριθμος.
2. "Το πέρασμα από τη JAVA στη C#" Σύγγραμμα, Ευθύμιος Αλέπης, Ιωάννης Χρήστος Παναγιωτόπουλος, 2018, Εκδόσεις Βαρβαρήγου.
3. "Αντικειμενοστραφείς Γλώσσες Προγραμματισμού: JAVA" Σύγγραμμα, Ιωάννης Χρήστος Παναγιωτόπουλος, 2011, Εκδόσεις Βαρβαρήγου.