תכנות מונחה עצמים – מטלה 3- גרפיקה, threads, lambda-expressions

חלק 1 – בחלק זה יש לבנות מחשבון גרפי המחשב ביטויים אריתמטיים. יש לממש את חישובים בעזרת פונקציה רקורסיבית להערכת ביטויים. יש לממש את המטלה לפי תבנית MVC.

הגדרות:

פעולה: אחת מפעולות החשבון `+,-,*,%`.

מספר: סיפרה שלמה בין `0` ל- `9`.

ביטוי הינו מספר, או הרכבה של שני ביטויים עם פעולה ביניהם וסוגריים מסביב.

דוגמה לביטויים תקינים:

```
1, (5+4), (4*((4%3) + 2)),   ((3+1)*(9-2))
```

דוגמה לביטויים **לא** תקינים:

```
-1,  4+3,  (21+3),  78,  ((2+1))
```

תוכלו להשתמש בפונקציה `find()` אשר מקבלת מחרוזת המייצגת ביטוי תקין, ומחזירה את האינדקס של הפעולה הראשית (המקום במחרוזת). הפעולה הראשית בביטוי אריתמטי היא פעולה המתבצעת אחרונה.

האינדקס של הפעולה הראשית בביטוי `(5+4)` הוא 2,

האינדקס של הפעולה הראשית בביטוי `(4*((4%3)+2))` הוא 2,

האינדקס של הפעולה הראשית בביטוי `((3+1)*(9-2))` הוא 6.

```
public static int find(String expr){
   int num = 0;
   int ans = -1;
   for (int i=0; i<expr.length(); i++){
      char c = expr.charAt(i);
      if (c=='(') num++;
      if (c==')') num--;
      if ((c=='+' || c=='-' || c=='*' || c=='%')&& num==1)  ans=i;
   }
   return ans;
}
```

כמו כן שימו לב לפעולות הבאות על תווים:

```
int valc = -1;
char c = '6';
if(c>='0' && c<='9') valc = c-'0';
System.out.println("valc = "+valc);
Output:  valc = 6
```

שאלות:

כתבו את המחלקה Ex13 המכילה את אוסף הפונקציות הסטטיות הבאות:

.1

```
public static int evalExpr1(String expr){. . .}
```

הפונקציה מקבלת מחרוזת המייצגת ביטוי חוקי (יש להניח קלט תקין) ומחזירה את ערכו.

דוגמה: **`evalExpr1("(4+5)")`** תחזיר 9.

.2

```
public static boolean isLegalExp(String expr){. . .}
```

הפונקציה מקבלת מחרוזת ומחזירה true אם ורק אם המחרוזת מייצגת ביטוי חוקי, אחרת היא מחזירה false.

.3

```
public static String evalExpr2(String expr){. . .}
```

הפונקציה מקבלת מחרוזת כלשהי. אם המחרוזת היא ביטוי חוקי - מחזירה את ערכו (כמחרוזת), אחרת מחזירה את המחרוזת "error".

<u>הערות:</u>

- כתבו תוכנית **`main`** (לא תיבדק) לבדיקה עצמית של כל הפונקציות שכתבתם.
- אחרי שסיימתם את העבודה בדקו את עצמכם שוב בעזרת תוכנית הבדיקה האוטומטית. על ידי כך תוודאו שהתוכנית שלכם כוללת את חתימות הפונקציות הנכונות – שימו לב התוכנית שלכם תיבדק גם על-ידי תוכנית אחרת.

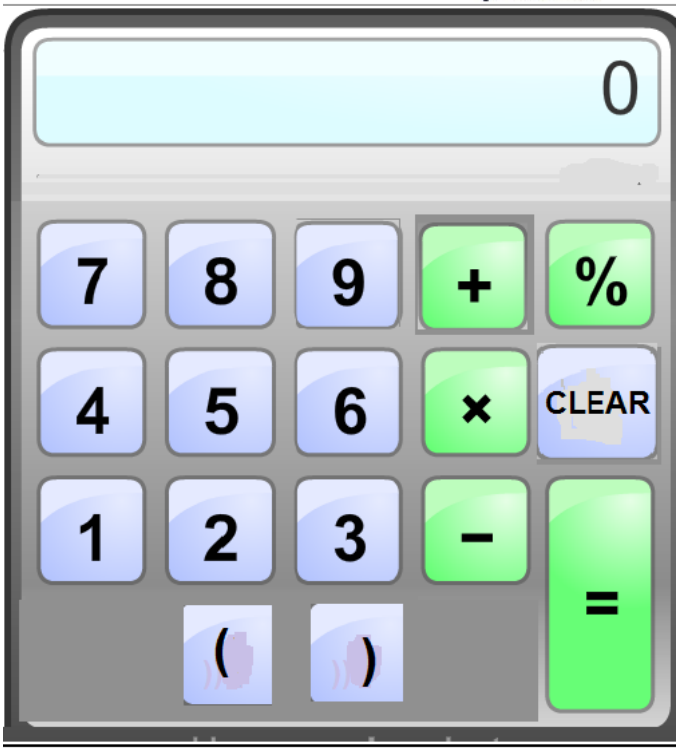
<u>הרצת תוכנית בדיקה</u>

הכנו עבורכם קובץ בדיקה **`Ex1Test.java`**, לאחר שתסיימו את הכנת התרגיל השתמשו בו כדי לבדוק את עצמכם.

<u>ממשק גרפי:</u>

יש לבנות ממשק גרפי ולהכין קובץ JAR לביצוע את החישובים. הממשק צריך להיות בסגנון הבא:

(לא בדיוק של אותו מראה)



חלק 2 – Threads

א)

<u>כללי:</u>

בתרגיל זה נבצע משימה המורכבת ממספר תת משימות. כל תת משימה תופעל בתהליכון נפרד, גם אם לא קיימת מניעה להפעלתה בתהליכון main או בתהליכון אחר.

אנו עומדים להכין פיצה. הכנתה כוללת 5 שלבים. לכל שלב משך זמן מוגדר:

תאור המשימה	משך המשימה (שניות)	שם המשימה
הכנת בצק	15	MakeDough
רידוד	8	RollDough
הכנת רוטב ותוספות	10	MakeSauce
הוספת הרוטב והתוספות	3	AddSauce
אפיה בתנור	20	BakePizza

כבסיס לתשובות נשתמש בקוד להלן. כתשובה, בקוד ביצוע משימה (פונקצית run) הציגו כפלט את המשימה המבוצעת ("מכין בצק" לדוגמא) ומשך המשימה יתורגם לקריאה ל- sleep. הצמידו לכל אובייקט משימה שם אחר.

```
class Worker implements Runnable
{
  public String TaskName;
  public long Duration;

  public Worker(String Name,long Duration)
  {
   this.TaskName=Name;
   this.Duration= Duration;
  }
  public void run()
  {
    System.out.println(this.TaskName +" Is Running.");
    Thread.sleep(this.Duration);
    System.out.println(this.TaskName +" Is Done.");
  }
}

…

  public static void main(String args[])
   {
     Worker worker=new Worker("TaskName",1000);
     Thread task =new Thread(worker,worker.TaskName);
     task.start();

   }
```

<u>שאלות:</u>

א. ממשו קוד המכניס את שמות המשימות ל - Collection , בפונקציה main קלטו שם משימה ממשתמש, ואם היא מוגדרת באוסף השמות, צרו אובייקט משימה מתאים והריצו אותו בתהליכון חדש.

ב. הגבילו את האפשרות ליצור אובייקט משימה אם היא כבר נוצרה בעבר.

ג. הגבילו את האפשרות להרצת משימה עד שלא הושלמו אלו המוגדרים לפניה ב- Collection, השתמשו בפונקציות join() או isAlive() כדי לוודא סטטוס השלמת משימות קודמות.

ד. מסתבר כי את משימת "הכנת רוטב ותוספות" ניתן לבצע בלי קשר למשימות האחרות, צרו מצב מלאכותי של Deadlock שבו משימת "הוספת הרוטב והתוספות" ממתינה להשלמת "הכנת רוטב ותוספות" וההפך . הערה : השתמשו באובייקטים סטטיים לביצוע סינכרונים.

ה. תקנו את הנעילה מסעיף ד'.

ב) בחלק זה יש לממש בסביבה של multithreading את האלגוריתם שלמדנו אותו בקורס אלגוריתמים 1.

נזכור את הבעיה: נתונים שני מערכים שווי אורך של מספרים שלמים:
`a[0]<a[1]<…<a[n-1], b[0]<b[1]<…<b[n-1]`, כאשר n הוא מספר איברי המערך.
יש למצוא את כל האיברים הגדולים מחציון במערך שהתקבל בעזרת מיזוג של שני המערכים הנתונים. נזכור את האלגוריתם:

(*) `ans[i] = max(a[i], b[n-i-1]), i=0,…,n-1`.

המערך ans מכיל את כל האיברים הגדולים מחציון במערך שהתקבל בעזרת מיזוג של מערכים a,b.

<u>הדרכה</u>: יש להגדיר n thread–ים, החישוב (*) לכל i צריך להתבצע ב-thread נפרד.
לבדיקת נכונות התכנה יש להשתמש בפונקציה רגילה של מיזוג שני מערכים (merge).

<u>תכנות</u>: יש להכין מחלקה בשם BigThanMedian, בתוך המחלקה יש לכתוב פונקציה סטטית

```
public static int[] bigThanMedianAlgo(int []a, int[] b){…}
```

הפונקציה מקבלת שני מערכים שווי אורך ממוינים בסדר עולה ומחזירה מערך של מספרים שלמים המכיל את כל האיברים הגדולים מחציון של מערך ממוזג. הפונקציה מממשת את האלגוריתם (*).

<u>דוגמא</u>: קלט: a[] = {3,5,7,9}, b[] = {1,4,7,12}, פלט: ans[] = {12,7,7,9}

יש למדוד את זמן ביצוע האלגוריתם למערכים גדולים.

לצורך בדיקת התכנה יש לכתוב פונקציה סטטית

```
public static int[] bigThanMedianMerge(int[]a, int[] b){…}
```

המקבלת את אותם שני מערכים ממוינים ומחזירה מערך של מספרים שלמים המכיל את כל האיברים הגדולים מחציון של מערך ממוזג. הפונקציה משתמשת במיזוג רגיל של שני מערכים.

קוד של ספריית MyLibrary מצורף למטלה.
תכנית הבדיקה:

<u>הערה</u>: כמובן זמן של שימוש במיזוג רגיל הוא קטן מזמן של האלגוריתם (*), כוון שבדיקה נעשתה

במחשב בעל שני CPUs בלבד .

```java
public static void test(int size){

      // Get 2 sorted arrays
      int[]a = MyLibrary.randomIntegerArray(size);
      int[]b = MyLibrary.randomIntegerArray(size);
      Arrays.sort(a);
      Arrays.sort(b);

      //The algorithm which is studied on the course Algorithms 1
      long start = System.currentTimeMillis();
      int[] ans1 = BigThanMedian.bigThanMedianAlgo(a, b);
      long end = System.currentTimeMillis();
      System.out.println("algo time = " + (end - start) + " ms");

      // The algorithm that uses the conventional merge algorithm
      start = System.currentTimeMillis();
      int[] ans2 = BigThanMedian.bigThanMedianMerge(a,b);
      end = System.currentTimeMillis();
      System.out.println("merge time = "+(end - start)+" ms");

      // Check of correctness
      Arrays.sort(ans1);
      Arrays.sort(ans2);
      if (Arrays.equals(ans1, ans2)) System.out.println("OK!!!");
      else System.out.println("ERROR...");
}

public static void main(String[] args) {
      int processors=Runtime.getRuntime().availableProcessors();
      int size = 10000;
      System.out.println("number of processors in this
                          evironment: " + processors);
      System.out.println("size = "+ size);
      test(size);
}
```

OUTPUT:

```
number of processors in this environment: 2
size = 10000
algo time = 535 ms
merge time = 1 ms
OK!!!
```

חלק 3 – Lambda-Expressions

בחלק זה יש לבנות מחשבון לביצוע פעולות מתמטיות לשלושה סוגי מספרים :

Integer
Double
Complex

כל הפעולות צריך לממש בעזרת **Lambda Expression**
ו-**Predefined Functional Interfaces**

Integer : n + m
n - m
n * m
n / m
n % m
n^2
n!
max(m,n)
min(m,n)
fibonacci(n)

Double : n + m
n - m
n * m
n / m
square root(n)
n^2
cos(n)
sin(n)
log(n)

Complex :

$$(a + bi) + (c + di) = (a + c) + (b + d)i$$

$$(a + bi)-(c + d,i) = (a-c) + (b-d)i$$

$$(a + bi)\cdot(c + di) = (ac-bd) + (ad + bc)i$$

$$\frac{a+bi}{c+di} = \frac{a+bi}{c+di} \cdot \frac{c-di}{c-di} = \frac{ac+bd}{c^2+d^2} + \frac{bc-ad}{c^2+d^2} i$$

ל-Integer ו-Double עדיף להשתמש ב Lambda Expression
ו Predefined Functional Interfaces
(**BinaryOperator<T>** ו- **UnaryOperator<T>**)

לביצוע פעולות עם מספרים מרוכבים לשני הממשקים האלה צריך להוסיף ממשק פונקציונלי

```
interface ComplexCalc {
	double calc(double re1, double im1, double re2, double im2);
}
```

למימוש של פעולות **כפל וחילוק**.
<u>דוגמה לתכנית בדיקה</u>

```
public class MyCalculator {

	Scanner sc = new Scanner(System.in);
	String str = null;
	boolean cond = true;
	.........................................
	private void menu(){

		System.out.print("Enter type of variable : I (int), D (double, C (complex) or
				E for exit\n");
```

```
            str = sc.next();
            switch ( str ) {
            case "I":
                    intCalc();
                    break;
            case "D":
                    doubleCalc();
                    break;
            case "C":
                    complexCalc();
                    break;
            case "E":
                    System.out.print("\nEND\n") ;
                    cond = false;
                    break;

            default:
                    System.out.print("Enter  I, D, C or E\n") ;
                    break;
            }
    }
    public static void main(String[] args) {

            MyCalculator myCal = new MyCalculator();

            while(myCal.cond)
                    myCal.menu();
    }
……................................................}
```

דוגמה הרצת התכנה:

```
Enter type of variable : I (int), D (double, C (complex) or E for exit
I

Enter operation :  + (addition)
                   - (subtraction)
                   * (multiplication)
                   / (division)
                   % (remainder)
                   MAX (maximum)
                   MIN (minimum)
                   S square (n²)
                   F (factorial)
```

```
                    FI (fibonacci)
+
Enter first parameter  :4
Enter second parameter :5
addition = 9
Enter type of variable : I (int), D (double, C (complex) or E for exit
I

Enter operation : + (addition)
                  - (subtraction)
                  * (multiplication)
                  / (division)
                  % (remainder)
                  MAX (maximum)
                  MIN (minimum)
                  S square (n²)
                  F (factorial)
                  FI (fibonacci)
-
Enter first parameter  :8
Enter second parameter :4
subtraction = 4
Enter type of variable : I (int), D (double, C (complex) or E for exit
I

Enter operation : + (addition)
                  - (subtraction)
                  * (multiplication)
                  / (division)
                  % (remainder)
                  MAX (maximum)
                  MIN (minimum)
                  S square (n²)
                  F (factorial)
                  FI (fibonacci)
*
Enter first parameter  :6
Enter second parameter :8
multiplication = 48
Enter type of variable : I (int), D (double, C (complex) or E for exit
I

Enter operation : + (addition)
```

```
                    - (subtraction)
                    * (multiplication)
                    / (division)
                    % (remainder)
                    MAX (maximum)
                    MIN (minimum)
                    S square (n²)
                    F (factorial)
                    FI (fibonacci)
/
Enter first parameter  :14
Enter second parameter :2
division = 7
Enter type of variable : I (int), D (double, C (complex) or E for exit
I

Enter operation : + (addition)
                    - (subtraction)
                    * (multiplication)
                    / (division)
                    % (remainder)
                    MAX (maximum)
                    MIN (minimum)
                    S square (n²)
                    F (factorial)
                    FI (fibonacci)
%
Enter first parameter  :24
Enter second parameter :5
remainder = 4
Enter type of variable : I (int), D (double, C (complex) or E for exit
I

Enter operation : + (addition)
                    - (subtraction)
                    * (multiplication)
                    / (division)
                    % (remainder)
                    MAX (maximum)
                    MIN (minimum)
                    S square (n²)
                    F (factorial)
                    FI (fibonacci)
```

```
MAX
Enter first parameter  :56
Enter second parameter :34
maximum = 56
Enter type of variable : I (int), D (double, C (complex) or E for exit
I

Enter operation : + (addition)
                  - (subtraction)
                  * (multiplication)
                  / (division)
                  % (remainder)
                  MAX (maximum)
                  MIN (minimum)
                  S square (n²)
                  F (factorial)
                  FI (fibonacci)
MIN
Enter first parameter  :34
Enter second parameter :12
minimum = 12
Enter type of variable : I (int), D (double, C (complex) or E for exit
I

Enter operation : + (addition)
                  - (subtraction)
                  * (multiplication)
                  / (division)
                  % (remainder)
                  MAX (maximum)
                  MIN (minimum)
                  S square (n²)
                  F (factorial)
                  FI (fibonacci)
S
Enter first parameter  :7
square = 49
Enter type of variable : I (int), D (double, C (complex) or E for exit
I

Enter operation : + (addition)
                  - (subtraction)
                  * (multiplication)
```

```
                    / (division)
                    % (remainder)
                    MAX (maximum)
                    MIN (minimum)
                    S square (n²)
                    F (factorial)
                    FI (fibonacci)
F
Enter first parameter  :6
factorial = 720
Enter type of variable : I (int), D (double, C (complex) or E for exit
I

Enter operation : + (addition)
                    - (subtraction)
                    * (multiplication)
                    / (division)
                    % (remainder)
                    MAX (maximum)
                    MIN (minimum)
                    S square (n²)
                    F (factorial)
                    FI (fibonacci)
FI
Enter first parameter  :12
fibonacci = 144
Enter type of variable : I (int), D (double, C (complex) or E for exit
I

Enter operation : + (addition)
                    - (subtraction)
                    * (multiplication)
                    / (division)
                    % (remainder)
                    MAX (maximum)
                    MIN (minimum)
                    S square (n²)
                    F (factorial)
                    FI (fibonacci)
ZZ
incorrect operation
Enter type of variable : I (int), D (double, C (complex) or E for exit
E
```

```
END

*****************************************************

Enter type of variable : I (int), D (double, C (complex) or E for exit
D

Enter operation : + (addition)
                  - (subtraction)
                  * (multiplication)
                  / (division)
                  SR (square root(n))
                  S square (n²)
                  COS  (cos(n))
                  SIN  (sin(n))
                  LOG ( log(n))
+
Enter first parameter  :4.5
Enter second parameter :3.2
addition = 7.7
Enter type of variable : I (int), D (double, C (complex) or E for exit
D

Enter operation : + (addition)
                  - (subtraction)
                  * (multiplication)
                  / (division)
                  SR (square root(n))
                  S square (n²)
                  COS  (cos(n))
                  SIN  (sin(n))
                  LOG ( log(n))
-
Enter first parameter  :4.5
Enter second parameter :3.2
subtraction = 1.2999999999999998
Enter type of variable : I (int), D (double, C (complex) or E for exit
D

Enter operation : + (addition)
                  - (subtraction)
                  * (multiplication)
```

```
                    / (division)
                    SR (square root(n))
                    S square (n²)
                    COS  (cos(n))
                    SIN  (sin(n))
                    LOG ( log(n))
*
Enter first parameter  :3.4
Enter second parameter :6.7
multiplication = 22.78
Enter type of variable : I (int), D (double, C (complex) or E for exit
D

Enter operation : + (addition)
                    - (subtraction)
                    * (multiplication)
                    / (division)
                    SR (square root(n))
                    S square (n²)
                    COS  (cos(n))
                    SIN  (sin(n))
              LOG ( log(n))
/
Enter first parameter  :3.8
Enter second parameter :2.1
division = 1.8095238095238093
Enter type of variable : I (int), D (double, C (complex) or E for exit
D

Enter operation : + (addition)
                    - (subtraction)
                    * (multiplication)
                    / (division)
                    SR (square root(n))
                    S square (n²)
                    COS  (cos(n))
                    SIN  (sin(n))
                    LOG ( log(n))
SR
Enter first parameter  :9.0
squareRoot = 3.0
Enter type of variable : I (int), D (double, C (complex) or E for exit
D
```

```
Enter operation : + (addition)
                  - (subtraction)
                  * (multiplication)
                  / (division)
                  SR (square root(n))
                  S square (n²)
                  COS  (cos(n))
                  SIN  (sin(n))
                  LOG ( log(n))
S
Enter first parameter  :3.6
square = 12.96
Enter type of variable : I (int), D (double, C (complex) or E for exit
D

Enter operation : + (addition)
                  - (subtraction)
                  * (multiplication)
                  / (division)
                  SR (square root(n))
                  S square (n²)
                  COS  (cos(n))
                  SIN  (sin(n))
                  LOG ( log(n))
COS
Enter first parameter  :4.5
cosinus = -0.2107957994307797
Enter type of variable : I (int), D (double, C (complex) or E for exit
D

Enter operation : + (addition)
                  - (subtraction)
                  * (multiplication)
                  / (division)
                  SR (square root(n))
                  S square (n²)
                  COS  (cos(n))
                  SIN  (sin(n))
                  LOG ( log(n))
SIN
Enter first parameter  :5.6
sinus = -0.6312666378723216
```

```
Enter type of variable : I (int), D (double, C (complex) or E for exit
D

Enter operation : + (addition)
                  - (subtraction)
                  * (multiplication)
                  / (division)
                  SR (square root(n))
                  S square (n²)
                  COS  (cos(n))
                  SIN  (sin(n))
                  LOG ( log(n))
LOG
Enter first parameter  :6.7
logarithm = 1.9021075263969205
Enter type of variable : I (int), D (double, C (complex) or E for exit
SSS
incorrect operation
Enter type of variable : I (int), D (double, C (complex) or E for exit
E

END

*******************************************************

Enter type of variable : I (int), D (double, C (complex) or E for exit
C

Enter operation : + (addition)
                  - (subtraction)
                  * (multiplication)
                  / (division)
+
Enter first parameter re :2
Enter first parameter im :3
First complex number : 2.0 + 3.0 * i
Enter second parameter re:4
Enter second parameter im:5
Second complex number : 4.0 + 5.0 * i
addition = 6.0 + 8.0 * i
Enter type of variable : I (int), D (double, C (complex) or E for exit
C
```

```
Enter operation : + (addition)
                  - (subtraction)
                  * (multiplication)
                  / (division)
-
Enter first parameter re :6
Enter first parameter im :3
First complex number : 6.0 + 3.0 * i
Enter second parameter re:8
Enter second parameter im:5
Second complex number : 8.0 + 5.0 * i
subtraction = -2.0 + -2.0 * i
Enter type of variable : I (int), D (double, C (complex) or E for exit
C

Enter operation : + (addition)
                  - (subtraction)
                  * (multiplication)
                  / (division)
*
Enter first parameter re :4
Enter first parameter im :2
First complex number : 4.0 + 2.0 * i
Enter second parameter re:7
Enter second parameter im:4
Second complex number : 7.0 + 4.0 * i
multiplication = 20.0 + 30.0 * i
Enter type of variable : I (int), D (double, C (complex) or E for exit
Enter type of variable : I (int), D (double, C (complex) or E for exit
C

Enter operation : + (addition)
                  - (subtraction)
                  * (multiplication)
                  / (division)
/
Enter first parameter re :5
Enter first parameter im :3
First complex number : 5.0 + 3.0 * i
Enter second parameter re:7
Enter second parameter im:2
Second complex number : 7.0 + 2.0 * i
division = 0.7735849056603774 + 0.20754716981132076 * i
```

```
Enter type of variable : I (int), D (double, C (complex) or E for exit
C

Enter operation : + (addition)
                  - (subtraction)
                  * (multiplication)
                  / (division)
AAA
incorrect operation
Enter type of variable : I (int), D (double, C (complex) or E for exit
E

END
```

```
class Complex {
   private double re;   // the real part
   private double im;   // the imaginary part

   // create a new object with the given real and imaginary parts
   public Complex(double real, double imag) {
      re = real;
      im = imag;
   }

   // return a string representation of the invoking Complex object
   public String toString() {
      if (im == 0) return re + "";
      if (re == 0) return im + "i";
      if (im <  0) return re + " - " + (-im) + "i";
      return re + " + " + im + "i";
   }

   // return the real or imaginary part
   public double re() { return re; }
   public double im() { return im; }

}
```

בהצלחה רבה!