מעבדה 5. נושא: מחסנית

תאריך הגשה לקבוצה של יום ב': 30.11.2020

תאריך הגשה לקבוצה של יום ד': 02.02.2020

(הגשה בזוגות)

יש לקרוא היטב לפני תחילת העבודה!

עבודה נעימה!

מבוא:

מחסנית (Stack) הוא מבנה נתונים התומך בפעולות הכנסת והוצאת איברים על פי העיקרון של LIFO (נכנס אחרון-יוצא ראשון). ניתן לממש את המחסנית בעזרת מערך או בעזרת רשימה מקושרת.
במעבדה זו נעסוק במימוש המחסנית בשני האופנים.

שלב 1.

ראשית ניצור מנשק של המחלקה

```
public interface Stack<T> {
    // Default maximum stack size
    public static final int DEF_MAX_STACK_SIZE = 10;

    public void push(T newElement);        // Push newElement onto stack
    public T pop();                        // Pop Element from top of stack
    public void clear();                   // Remove all elements from stack
    public boolean isEmpty();              // Return true if stack is empty
    public boolean isFull();               // Return true if stack is full
    } // interface Stack
```

שלב 2.

נכתוב מחלקה בשם StackArray המממשת את המנשק הנ"ל בעזרת **מערך** ומכילה:

1). **בנאי**

StackArray (int size);

הבונה מחסנית בגודל size.

2). מממשת **toString** משלה.

שים לב- יש להחזיק מצביע לתחילת המחסנית top שהוא קיצור של TopOfStack. ניתן להניח כי המערך המוגדר בהתחלה יהיה מספיק גדול כדי להחזיק את המחסנית (אין צורך ב realloc).

```
public class StackArray  <T>  implements Stack<T>
{
        private int top;                    // Top Of Stack:  Index for the top element
        private T [ ] element;   // Array containing stack elements
        //element = (T [ ]) new Object[size] …..
……................
}
```

שלב 3. כתוב מחלקה בשם StackList המממשת את המנשק הנ"ל בעזרת רשימה מקושרת חד-כיוונית שמימשת במעבדה 3 ומכילה:

1). **בנאי**

```
StackList (int size);
```

הבונה מחסנית בגודל size (בפעל, יש להתעלם מהגודל הנתון).

2). ממשת **toString** משלה.

```
public class StackList <T>  implements Stack<T>
{
        private SNode<T> top;                // Top Of Stack:  Reference to top of stack
        SLinkedList  <T> listOfStackElements;
……................
}
```

שלב 4. בדיקות

צור שני קלאסים חדשים בשם TestAStack.java ובשם TestLStack.java שבעזרת tokenizer תומכים בפעולות הבאות ובודקים את עבודתם של StackArray ו StackList בהתאמה **עבור איברים שהם מספרים שלמים Integer**:

פקודה	פעולה
Push x	מוסיף איבר x
Pop	מוחק איבר המסומן ע"י הסמן
P	מדפיס את כל אברי המחסנית
F	שאילתא: האם המחסנית מלאה (full)
E	שאילתא: האם המחסנית ריקה (empty)
C	מוחק את כל המחסנית (clear)
Q	סיים את התוכנית

שלב 5. בדיקת תקינות הסוגריים בביטוי אריתמטי בעזרת מחסנית

ביטוי חשבוני מכיל סוגריים מסוגים שונים. נגדיר מהו ביטוי חשבוני תקין מבחינת סוגריים: ביטוי המכיל סוגריים מסוגים שונים במספר לא מוגבל, ובלבד שיהיו **מאוזנים**. איזון הסוגריים מחייב שמספר

הסוגריים הפותחים והסוגריים הסוגרים יהיה שווה, וכן לכל פותח יימצא סוגר מאותו סוג במקום המתאים. לדוגמה, הביטויים האלה תקינים:

((a))

(b + a - 2 * 7)

{ + 32 * (37 *) / [5 + 1]} - 4

שים לב: הביטוי האחרון תקין מבחינת הסוגריים, אף שכביטוי חשבוני הוא אינו תקין.

ואילו הביטויים האלה אינם תקינים:

a + ((c

([3 + a) + 4]

[ ) (5 - 3] * [2 - 3]

כתוב תוכנית **checkBalancedBrackets.java** המקבלת ביטוי אריתמטי וממשת את המשימה בעזרת מחסנית.

המלצה: הגדר מחרוזת של סוגריים פותחים וסוגריים סוגרים והשתמש בפעולת charAt.

הגשה:

יש להגיש את הקבצים הבאים:

Stack.java, StackArray.java, StackList.java, TestAStack.java, TestLStack.java, checkBalancedBrackets.java.