Object Oriented Programming Course (CEID_NNY106)

Εργαστηριακή Άσκηση

RPN Calculator σε Java

1. Στόχος ΕΑ

Εξοικείωση με:

- συγγραφή και εκτέλεση java εφαρμογής
- κατασκευές της Java που η γλώσσα υιοθετεί από την C
- βασικές έννοιες του αντικειμενοστρεφούς παραδείγματος προγραμματισμού,
- την βασική βιβλιοθήκη της Java (Java API) https://docs.oracle.com/javase/8/docs/api/

Reverse Polish Notation Calculator (Java version)

Η Εργαστηριακή Άσκηση (ΕΑ)έχει ως στόχο την μεταφορά της υλοποίησης με C της RPN Calculator της ΕΑ 1, στο περιβάλλον της Java με την ελάχιστη δυνατή αξιοποίηση των κατασκευών της γλώσσας που υποστηρίζουν το αντικειμενοστρεφές παράδειγμα προγραμματισμού.

Η Δραστηριότητα βασίζεται στις Δραστηριότητες 6.2, 6.3 και 6.4 της ενότητας 6.3 του βιβλίου "Από τις Υπολογιστικές στις Κυβερνο-Φυσικές Διεργασίες και το IoT: Αντικείμενα και Υπηρεσίες", Κ. Θραμπουλίδης, ISBN 978-960-418-961-8, 2022, ΕΚΔΟΣΕΙΣ Α. ΤΖΙΟΛΑ & ΥΙΟΙ Α.Ε. https://sites.google.com/view/fromcomputationaltocyber-physi/home

2. Το Πρόγραμμα

Αναπτύξτε ένα πρόγραμμα σε Java που θα έχει την λειτουργικότητα της RPN αριθμομηχανής που αναπτύχθηκε στην ΕΑ Νο 1. Για την υλοποίηση θα χρησιμοποιήσετε στοίβα (stack).

3. Οδηγίες εκτέλεσης

Για να ικανοποιήσουμε τους σκοπούς της ΕΑ αυτής υποθέτουμε πως:

Α) ο χρήστης δίνει την έκφραση παρεμβάλλοντας ανάμεσα σε τελεστές και τελεστέους ένα κενό χαρακτήρα, π.χ., "12 24 + = ", "26 14 + 310 210 - + = ", και

Β) υποστηρίζονται μόνο ακέραιοι τελεστέοι και οι τελεστές + - * / =.

Θα δουλέψετε σε τρεις εκδόσεις του προγράμματος με την σειρά που παρουσιάζονται παρακάτω.

Μην διστάσετε να αξιοποιήσετε την επικοινωνία με τους συμφοιτητές σας και τους διδάσκοντες για να ξεπεράσετε τα δύσκολα στην κατανόηση ή εκτέλεση σημεία και πιθανόν να συμβάλετε στην βελτίωση της ΕΑ.

4. Έκδοση No 1 (V1.0) - Χρήση της κλάσης java.util.Stack

Στην έκδοση αυτή το πρόγραμμα σας θα αποτελείται από μία μόνο κλάση, την ***RpnCalc***. Θα χρησιμοποιήσετε την κλάση **Scanner** για να διαβάσετε την RPN έκφραση από την βασική είσοδο η οποία είναι η **`System.in`**.

Για την αποτελεσματικότερη ανάπτυξη της έκδοσης αυτής σας συνιστούμε να εκτελέσετε τις παρακάτω Δράσεις.

4.1 **Δράση 1** – Παίζοντας με την κλάση **Scanner**

Εξοικειωθείτε με την κλάση Scanner δημιουργώντας στο **BlueJ διαδραστικά** (Tools->Use library class) ένα στιγμιότυπο της με όρισμα την βασική είσοδο (System.in) και παίρνοντας από αυτό τα συνθετικά μιας έκφρασης που θα δώσετε εσείς από την βασική είσοδο.

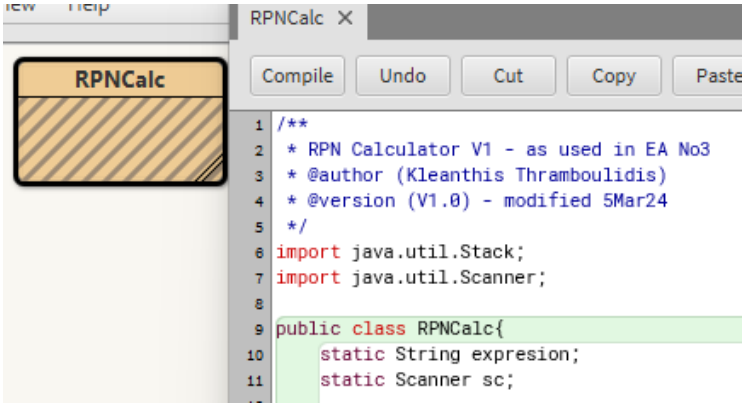
Δοκιμάστε πάνω σε αυτό τις μεθόδους nextLine(), nextInt(), nextDouble(), hasNext(), hasNextInt(), κ.λ.π.

Βάλτε σε μία στοίβα τους τελεστέους της έκφρασης.

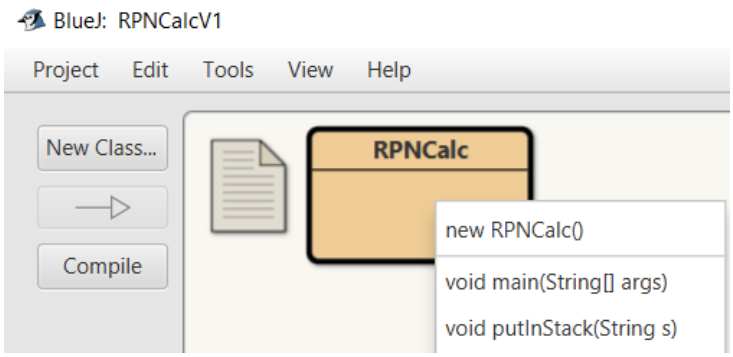
4.2 **Δράση 2** - Δημιουργία Project και εκτέλεση προγράμματος

Δημιουργήστε ένα project στο BlueJ με όνομα RPNCalcV1 και στην συνέχεια την κλάση RPNCalc. Δομήστε την main() έτσι ώστε να ζητά από τον χρήστη μια RPN έκφραση και να διαβάζει την έκφραση αυτή αξιοποιώντας την κλάση Scanner. Τυπώστε την έκφραση.

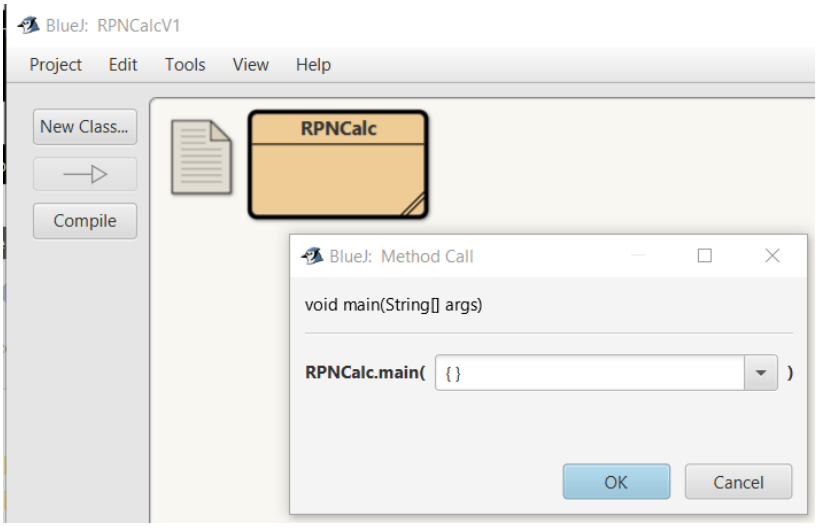
Μεταγλωττίστε τον κώδικα σας επιλέγοντας Compile στον editor όπως φαίνεται παρακάτω



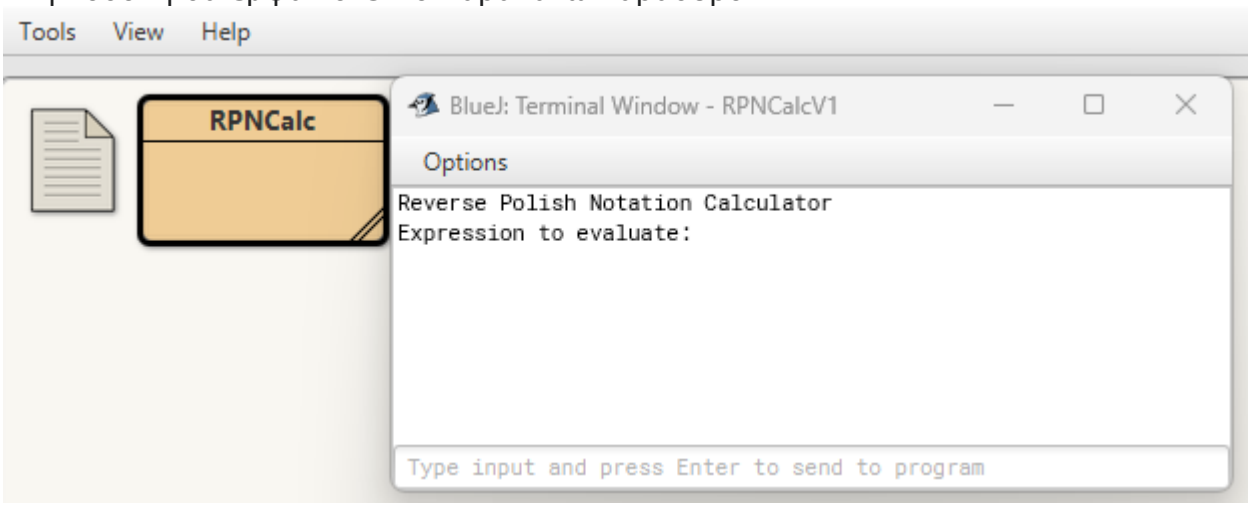
Αν δεν έχετε λάθη μεταγλώττισης επιλέξτε την κλάση RPNCalc στην κύρια οθόνη του BlueJ και με δεξί κλικ επιλέξτε την main όπως φαίνεται παρακάτω.



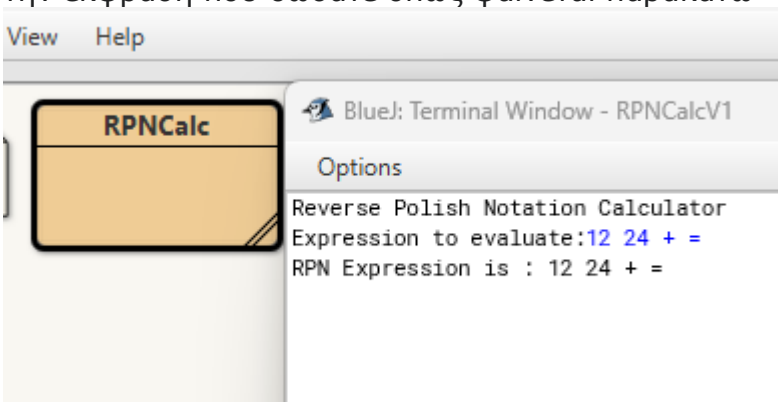
Αυτό έχει ως αποτέλεσμα να εκτελεστεί η main. Στην οθόνη θα εμφανιστεί το παρακάτω παράθυρο



Επιλέξτε OK.
Στην οθόνη θα εμφανιστεί το παρακάτω παράθυρο.



Πληκτρολογήστε την έκφραση στο κάτω μέρος του παραθύρου και θα δείτε στο παράθυρο την έκφραση που δώσατε όπως φαίνεται παρακάτω



Ορίστε μια μέθοδο getExpression(), αν δεν το έχετε ήδη κάνει, για να κάνετε τον κώδικα σας αρθρωτό (modular).

Στην τελευταία ενότητα θα βρείτε μια ενδεικτική απάντηση. Μελετήστε την μόνο μετά από την συγγραφή του δικού σας κώδικα.

4.3 **Δράση 3** – Ορίστε την main χρησιμοποιώντας την μέθοδο getOp()

Ορίστε την κύρια δομή της main() όπως κάνατε στην έκδοση για C αξιοποιώντας μια μέθοδο getOp(). Μπορείτε να δημιουργήσετε ένα άλλο στιγμιότυπο της Scanner το οποίο θα κάνει parse την έκφραση που έχετε αποθηκεύσει στην μεταβλητή expression. Η getOp σας θα παίρνει το επόμενο συνθετικό της έκφρασης και θα ενημερώνει κατάλληλα τις operand και inputType.

Στην τελευταία ενότητα θα βρείτε μια ενδεικτική απάντηση. Μελετήστε την μόνο μετά από την συγγραφή του δικού σας κώδικα για την έκδοση.

4.4 **Δράση 4** - Επέκταση λειτουργικότητας

Συμπληρώστε τον κώδικα για να έχετε πλήρη λειτουργικότητα της αριθμομηχανής σας.

5. Δράση 5 - Έκδοση No 2 (V2.0)

Στην έκδοση αυτή θα υλοποιήσετε μια στοίβα ακεραίων πρωτογενούς τύπου int όπως είχατε κάνει στην C με τη δήλωση ενός πίνακα ακεραίων και της ακέραιας μεταβλητής sp. Την στοίβα αυτή θα χρησιμοποιήσετε αντί για την κλάση Stack. Δημιουργήστε ένα αντίγραφο του project της πρώτης έκδοσης και δουλέψτε πάνω σε αυτό. Προσέξτε στην έκδοση αυτή δεν ορίζουμε κλάση στοίβα. Αυτό θα γίνει στην επόμενη έκδοση.

Προσέξτε τη διαφορά (από την C) στον ορισμό του πίνακα stack. Στην τελευταία ενότητα θα βρείτε μια ενδεικτική απάντηση. Μελετήστε την μόνο μετά από την συγγραφή του δικού σας κώδικα για την έκδοση.

6. Δράση 6 - Έκδοση No 3 (V3.0) – Ορισμός κλάσης MyStack

Στην έκδοση αυτή θα υλοποιήσετε μια στοίβα ακεραίων ορίζοντας μια δική σας κλάση, την MyStack. Δημιουργήστε ένα αντίγραφο του project της πρώτης έκδοσης και δουλέψτε πάνω σε αυτό. Αφαιρέστε την κλάση Stack της βασικής βιβλιοθήκης και αντικαταστήστε την στο πρόγραμμα σας από την δική σας MyStack.

Στην έκδοση αυτή η στοίβα θα είναι στιγμιότυπο της δικής σας κλάσης MyStack την οποία θα πρέπει να ορίσετε και στη συνέχεια να τροποποιήσετε την RpnCal ώστε να την χρησιμοποιεί.

Ορίστε κατάλληλα την δομή της κλάσης σας (array of primitive int and sp index) και την συμπεριφορά της (push και pop methods)

7. Δράση 7 - Έκδοση No 4 (V4.0) – Πρόσθετη λειτουργικότητα

Προσθέστε επιπλέον λειτουργικότητα ώστε :

a) το πρόγραμμα σας να δέχεται και να υπολογίζει τιμές εκφράσεων μέχρις ότου ο χρήστης δώσει 'q' ως ένδειξη τερματισμού προγράμματος, και

b) να εκτελεί πράξεις με δεκαεξαδικούς αριθμούς.

8. Ενδεικτικές Απαντήσεις

Στην ενότητα αυτή θα βρείτε ενδεικτικές απαντήσεις για τις Δράσεις της άσκησης. Χρησιμοποιήστε τις μόνο μετά από την δική σας προσπάθεια σε κάθε Δράση.

8.1 Δράση 2

```
1  /**
2   * RPN Calculator V1 - as used in EA No3
3   * @author (Kleanthis Thramboulidis)
4   * @version (V1.0) - modified 5Mar24
5   */
6  import java.util.Stack;
7  import java.util.Scanner;
8
9  public class RPNCalc{
10     static String expresion;
11     static Scanner sc;
12
13     public static void main(String[] args) {
14         System.out.println("Reverse Polish Notation Calculator");
15         expresion = getExpression();
16         System.out.println("RPN Expression is : " + expresion);
17     }
18
19
20     static String getExpression() {
21         Scanner sc = new Scanner(System.in);
22         System.out.print("Expression to evaluate:");
23         var s = sc.nextLine();
24         sc.close();
25         return s;
26     }
27 }
```

8.2 Δράση 3

Σας δίνεται μέρος του πηγαίου κώδικα της πρώτης έκδοσης (Σχήματα 1 και 2) καθώς και επεξήγηση αυτού.

8.2.1 Η μέθοδος **main()**

Το **Σχήμα 1** δίνει το πρώτο μέρος μιας πρώτης έκδοσης του προγράμματος, που στην φάση αυτή είναι η κλάση RPNCalc. Η συμπεριφορά του προγράμματος όταν αυτό θα εκτελεστεί ορίζεται από την μέθοδο main().

Η *main* στην γραμμή 19 δημιουργεί μια στοίβα για ακεραίους ως ένα στιγμιότυπο της κλάσης **Stack** και αναθέτει την αναφορά του στην μεταβλητή st, η οποία έχει δηλωθεί ως data member κλάσης (αυτό προσδιορίζεται από το keyword static) (γραμμή 15).

Στη συνέχεια, γραμμή 20, καλεί την μέθοδο κλάσης getExpression() (ο ορισμός της βρίσκεται στο Σχήμα 2), η οποία παίρνει από τον χρήστη την προς υπολογισμό έκφραση, π.χ. την 12 24 + = , την βάζει σε ένα στιγμιότυπο της String και επιστρέφει στην main

την αναφορά (reference) του στιγμιοτύπου αυτού. Την αναφορά αυτή η main αναθέτει στην μεταβλητή expression που έχει δηλωθεί στη γραμμή 11 ως data member κλάσης.

Στη συνέχεια, γραμμή 21, δημιουργεί ένα στιγμιότυπο της κλάσης **Scanner** (*The* Scanner *class of the* java.*util package is used to read input data from different sources*, w3schools, Oracle) και αναθέτει την αναφορά του στην μεταβλητή κλάσης sc η οποία έχει δηλωθεί στην γραμμή 12.

Η γραμμή 22 ορίζει για τον scanner ως οριοθέτη (delimiter) των συνθετικών της έκφρασης τον χαρακτήρα κενό (" ").

```
6  import java.util.Stack;
7  import java.util.Scanner;
8
9  public class RPNCalc{
10     static final int OPERAND=1;
11     static String expresion;
12     static Scanner sc;
13     static String operand;
14     static int inputType;
15     static Stack<Integer> st;
16
17     public static void main(String[] args) {
18         System.out.println("Reverse Polish Notation Calculator");
19         st = new Stack<Integer>();
20         expresion = getExpression();
21         sc = new Scanner(expresion);
22         sc.useDelimiter(" ");
23         while(sc.hasNext()){
24             inputType = getOp(sc);
25             switch(inputType) {
26                 case OPERAND:
27                     putInStack(operand);
28                     break;
29                 case '+':
30                     add();
31                     break;
32                 case '=':
33                     displayResult();
34                     break;
35             }
36         }
37         sc.close();
38     }
```

Σχήμα 1. Η κλάση RpnCalc της 1ης έκδοσης (μέρος πρώτο).

Το στιγμιότυπο sc της Scanner θα χρησιμοποιηθεί στη συνέχεια:

a) από την while (γραμμή 23) για να ορίσει τον αριθμό των επαναλήψεων, μία επανάληψη για κάθε ένα συνθετικό της έκφρασης, και

b) από την getOp (γραμμή 24) για να πάρει αυτή ένα προς ένα τα συνθετικά της έκφρασης, που είναι αποθηκευμένη στο στιγμιότυπο expression της String.

Η main στις γραμμές 25-35 χειρίζεται τα συνθετικά της έκφρασης κατάλληλα, όπως έκανε και στην C υλοποίηση.

Τέλος, στην γραμμή 37, κλείνει τον Scanner.

8.2.2 Η μέθοδος **getOp()**

Το **Σχήμα 2** δίνει το δεύτερο μέρος της κλάσης RpnCalc.

Η βασική λειτουργία της getExpression() αναφέρθηκε παραπάνω. Πιο αναλυτικά εξηγείται στην άλλη υπο-ενότητα.

```
41      static String getExpression() {
42      Scanner sc = new Scanner(System.in);
43      System.out.print("Expression to evaluate:");
44      var s = sc.nextLine();
45      sc.close();
46      return s;
47  }
48
49  private static int getOp(Scanner sc) {
50      String input;
51      input = sc.next();
52      if(Character.isDigit(input.charAt(0))){
53          operand=input;
54          return OPERAND;
55          }
56      else
57          return input.charAt(0);
58       }
59
60  static void putInStack(String s){
61      st.push(Integer.parseInt(s));
62  }
63
64  private static void add() {
65      st.push(st.pop()+st.pop());
66  }
67
68  private static void displayResult() {
69      System.out.println("result=" +st.pop());
70  }
71 }
```

Σχήμα 2. Η κλάση RpnCalc της 1ης έκδοσης (μέρος δεύτερο).

Η getOp, όπως θα παρατηρήσετε, ακολουθεί την ίδια λογική με την getOp() της C υλοποίησης.

Με την γραμμή 51, η μέθοδος παίρνει από την προς υπολογισμό έκφραση, που έχει δώσει ο χρήστης, το επόμενο συνθετικό της και ενημερώνει την μεταβλητή input να δείχνει σε αυτό.
Στη συνέχεια, γραμμές 52-57, ελέγχει αν το συνθετικό αυτό είναι operand ή operator και επιστρέφει την πληροφορία αυτή στην main. Ενημερώνει ταυτόχρονα (γραμμή 53) και την αναφορά operand στην περίπτωση που το συνθετικό είναι τελεστής.

Για την επικοινωνία της getOp με την main έχουν δηλωθεί οι γνωστές σας μεταβλητές operand και inputType, οι οποίες έχουν οριστεί ως data members κλάσης (keyword static).

Προσέξτε πως η μέθοδος ορίζεται ως private static καθώς ορίζει συμπεριφορά κλάσης. Επιστρέφει δε int.

8.2.3 Η μέθοδος **getExpression()**

Η μέθοδος αξιοποιεί την κλάση Scanner της βασικής βιβλιοθήκης της Java για να πάρει από την βασική είσοδο του συστήματος, στην οποία έχουμε πρόσβαση με την αναφορά System.in, την προς υπολογισμό έκφραση.

Πιο συγκεκριμένα, στην γραμμή 42 δημιουργεί ένα στιγμιότυπο της Scanner, του οποίου αξιοποιεί την υπηρεσία nextLine για να πάρει από την βασική είσοδο την προς υπολογισμό έκφραση.

Την προς υπολογισμό έκφραση αποθηκεύει ως ένα στιγμιότυπο της String (γραμμή 44). Στη συνέχεια, γραμμή 46, κλείνει τον scanner και επιστρέφει την αναφορά του στην main.

8.3 Δράση 5

Η πρόταση

```
static int [] stack;
```

δηλώνει την stack ως αναφορά σε πίνακα ακεραίων.

Η δημιουργία της στοίβας (πίνακα ακεραίων) θα γίνει μέσα στην main() με την παρακάτω πρόταση.

```
stack = new int[20];
```

Με βάση τον ορισμό αυτόν της στοίβας θα ορίσετε τις μεθόδους push and pop.

Ελέγξτε τη λειτουργικότητα της εφαρμογής σας.