Arithmetic Calculator

Γράψτε ένα πρόγραμμα σε java το οποίο επιλύει οποιαδήποτε σύνθετη αριθμητική έκφραση η οποία μπορεί να περιέχει

1. **Παρενθέσεις:** δηλώνουν την προτεραιότητα των πράξεων.
2. **Τα σύμβολα των πράξεων (τελεστές):**
    - **+** (*πρόσθεση*), (π.χ. 3.3+2.2)
    - **-** (*αφαίρεση*), (π.χ. 3.3-2.2)
    - **x** ή ***** (*πολλαπλασιασμός*), (π.χ. 3.3x2.2 ή 3.3*2.2)
    - **/** (*διαίρεση*), (π.χ. 3.3/2.2)
    - **^** (*ύψωση σε δύναμη*), (π.χ. 3.3^2.2)
3. **Θετικούς αριθμούς (ακέραιους ή κινητής υποδιαστολής):** Όλοι οι υπολογισμοί γίνονται μεταξύ αριθμών κινητής υποδιαστολής.
4. **Κενούς χαρακτήρες** (`[ \t\n\r]`) μεταξύ παρενθέσεων, τελεστών και αριθμών.

Η κατάταξη των τελεστών από την υψηλότερη προς την χαμηλότερη προτεραιότητα δίνεται παρακάτω. **Εκφράσεις που περιέχονται μέσα σε παρενθέσεις έχουν προτεραιότητα έναντι οποιουδήποτε τελεστή**.

```
^(υψηλή)   /(μέση+)    *x(μέση-)   -(χαμηλή+)   +(χαμηλή-)
```

Παράδειγμα αριθμητικής έκφρασης είναι η παρακάτω:

```
5 + (((3.3 + 6.6) * 9.2 ) + 12.546) * 2.323 +
( ( ( 33.3 + 2342.1 ) * 55.555 )  - 10000.009 ) +   11.334 * 2.3  ^3.5
```

Προκειμένου να επιλύσετε το πρόβλημα καλείστε να δημιουργήσετε το ισοδύναμο δυαδικό δένδρο υπολογισμού της παρακάτω έκφρασης (παράδειγμα δίνεται σε επόμενη σελίδα). Οι κανόνες που διέπουν το δένδρο είναι οι εξής:

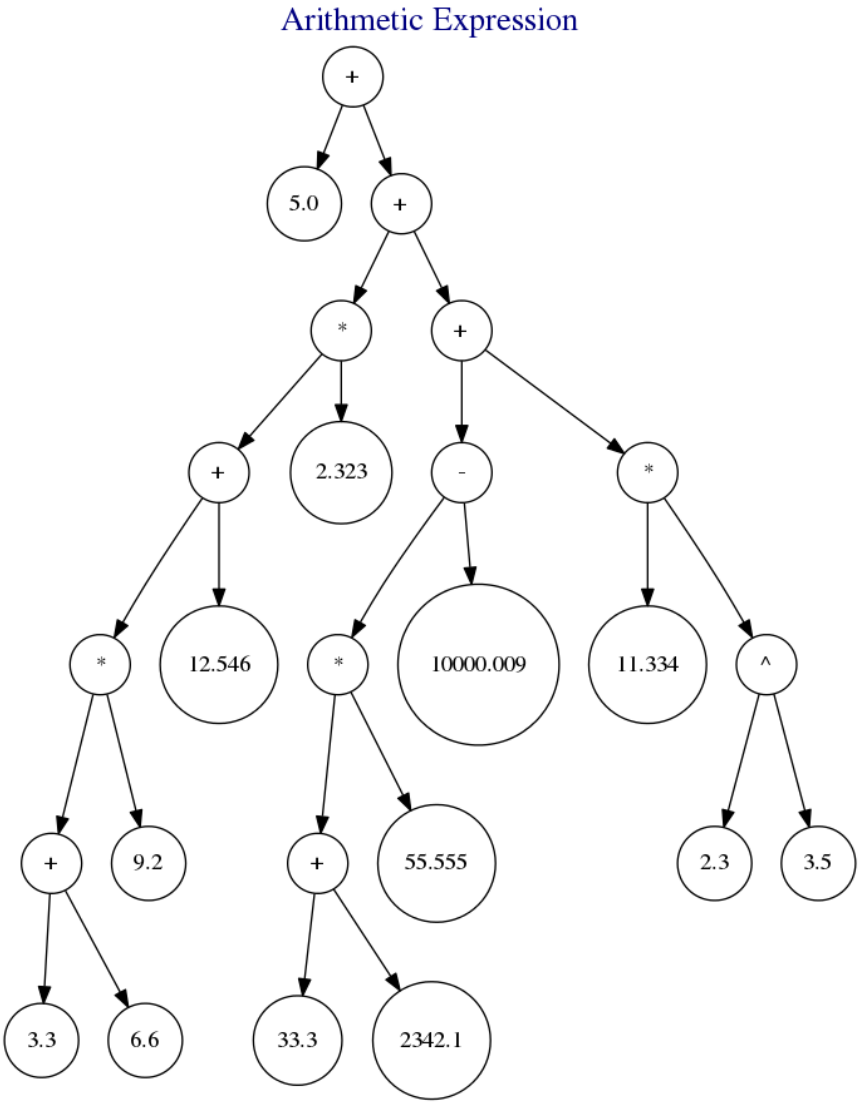
- Για κάθε αριθμητική έκφραση, αναζητούνται όλοι οι τελεστές που βρίσκονται εκτός παρενθέσεων. Επιλέγουμε έναν από τους διαθέσιμους τελεστές με τη χαμηλότερη διαθέσιμη προτεραιότητα. Σπάμε την αριθμητική έκφραση σε δύο υπο-αριθμητικές εκφράσεις (αριστερή και δεξιά) οι οποίες ενώνονται με τον τελεστή.
- Όταν συναντούμε παρενθέσεις, θα πρέπει να τις αφαιρέσουμε και να αναλύσουμε την έκφραση εντός των παρενθέσεων. Η αφαίρεση των παρενθέσεων θα πρέπει να γίνει αφού προηγουμένως έχουμε εξετάσει όλους τους αριθμητικούς τελεστές εκτός παρενθέσεων.
- Η διαδικασία επαναλαμβάνεται αναδρομικά έως ότου όλες οι υπο-εκφράσεις καταλήγουν να περιέχουν μόνο αριθμούς. Στην περίπτωση που σε μία έκφραση ο πρώτος και ο τελευταίος χαρακτήρας είναι παρενθέσεις άνοιγμα ‘(‘ και κλείσιμο ‘)’ παρένθεσης, αφαιρούμε τις παρενθέσεις πριν προχωρήσουμε περαιτέρω στην ανάλυση της έκφρασης.

Το πρόγραμμα σας διαβάζει μία αριθμητική έκφραση από το πληκτρολόγιο. Για την αριθμητική έκφραση που διαβάζει πρέπει

A. Να ελέγξει ότι πρόκειται για σωστά δομημένη αριθμητική έκφραση. Συγκεκριμένα θα πρέπει να ελέγχει ότι στην έκφραση
    a. κλείνουν όλες οι παρενθέσεις που έχουν ανοίξει προηγουμένως

b. δεν υπάρχουν σύμβολα κλεισίματος παρένθεσης **‘)’** χωρίς να προηγείται σύμβολο ανοίγματος της παρένθεσης.
c. δεν περιέχονται άλλοι χαρακτήρες εκτός από αριθμοί (σταθερής και κινητής υποδιαστολής) χαρακτήρες (whitespace), παρενθέσεις και οι τελεστές (**+,-,/,*,x,^**). Η υποστιαστολή σημειώνεται με την τελεία **‘.’**.

Β. να δημιουργήσει το ισοδύναμο δυαδικό δένδρο που αντιστοιχεί στη δοθείσα αριθμητική έκφραση. Η κλάση του δένδρου θα πρέπει να περιέχει τουλάχιστον τις εξής μεθόδους:
1. **toDotString:** ~~εκτυπώνει~~ επιστρέφει ένα αλφαριθμητικό που αντιστοιχεί στο καστασκευασμένο δένδρο. Το αλφαριθμητικό αυτό στη συνέχεια, μπορεί να χρησιμοποιηθεί από το πρόγραμμα dot/graphviz προκειμένου να εκτυπώσει το διάγραμμα του δένδρου σε μορφή PNG, SVG ή άλλη. ~~Εάν κληθεί η **toDotString** για ένα υποδένδρο του τελικού δένδρου θα πρέπει να επιστρέψει την ισοδύναμη έκφραση για το δένδρο αυτό~~.
2. **toString:** καλούμενη από τη ρίζα του δένδρου επιστρέφει ένα αλφαριθμητικό που περιέχει την ίδια ή μία **ισοδύναμη** αριθμητική έκφραση με αυτή που αντιστοιχεί στην αρχική δοθείσα έκφραση. Η έκφραση θα πρέπει να πηγάζει από το δένδρο που δημιουργήσατε και να μην αποθηκεύει αυτούσια την αρχική έκφραση την οποία και εκτυπώνει. Το αλφαριθμητικό που προκύπτει από την μέθοδο **toString()** μπορεί να διαφέρει από το αρχικά δοθέν αλφαριθμητικό, αλλά θα πρέπει να είναι αριθμητικά ισοδύναμα. ~~Εάν κληθεί η **toString** για ένα υποδένδρο του τελικού δένδρου θα πρέπει να επιστρέψει την ισοδύναμη έκφραση για το δένδρο αυτό~~.
3. **calculate:** Υπολογίζει την αριθμητική τιμή της έκφρασης που αντιστοιχεί στο υποδένδρο αυτό. Καλούμενη από την ρίζα του δένδρου επιστρέφει την ισοδύναμη αριθμητική τιμή (αποτέλεσμα) της αρχικής αριθμητικής έκφρασης.



Διάβασμα γραμμής από το πληκτρολόγιο

Ο κώδικας για να διαβάσετε μία γραμμή από το πληκτρολόγιο και να την αποθηκεύσετε σε ένα String δίνεται παρακάτω:

```
public class ReadLine {

  public static void main(String []args) {
    java.util.Scanner sc = new java.util.Scanner(System.in);
    System.out.print("Enter math expression: ");
    String line = sc.nextLine();
    System.out.println("Math expresssion is: "+line);
  }
}
```

Αρχείο dot

Παράδειγμα αρχείου **dot** για το παραπάνω διάγραμμα είναι το εξής:

```
digraph ArithmeticExpressionTree {
    label="Arithmetic Expression"
    1028566121 [label="+", shape=circle, color=black]
    1028566121 -> 1118140819
    1118140819 [label="5.0", shape=circle, color=black]
    1028566121 -> 1975012498
    1975012498 [label="+", shape=circle, color=black]
    1975012498 -> 1808253012
    1808253012 [label="*", shape=circle, color=black]
    1808253012 -> 589431969
    589431969 [label="+", shape=circle, color=black]
    589431969 -> 1252169911
    1252169911 [label="*", shape=circle, color=black]
    1252169911 -> 2101973421
    2101973421 [label="+", shape=circle, color=black]
    2101973421 -> 685325104
    685325104 [label="3.3", shape=circle, color=black]
    2101973421 -> 460141958
    460141958 [label="6.6", shape=circle, color=black]
    1252169911 -> 1163157884
    1163157884 [label="9.2", shape=circle, color=black]
    589431969 -> 1956725890
    1956725890 [label="12.546", shape=circle, color=black]
    1808253012 -> 356573597
    356573597 [label="2.323", shape=circle, color=black]
    1975012498 -> 1735600054
    1735600054 [label="+", shape=circle, color=black]
    1735600054 -> 21685669
    21685669 [label="-", shape=circle, color=black]
    21685669 -> 2133927002
    2133927002 [label="*", shape=circle, color=black]
    2133927002 -> 1836019240
    1836019240 [label="+", shape=circle, color=black]
    1836019240 -> 325040804
    325040804 [label="33.3", shape=circle, color=black]
    1836019240 -> 1173230247
    1173230247 [label="2342.1", shape=circle, color=black]
    2133927002 -> 856419764
    856419764 [label="55.555", shape=circle, color=black]
    21685669 -> 621009875
    621009875 [label="10000.009", shape=circle, color=black]
    1735600054 -> 1265094477
    1265094477 [label="*", shape=circle, color=black]
    1265094477 -> 2125039532
    2125039532 [label="11.334", shape=circle, color=black]
    1265094477 -> 312714112
    312714112 [label="^", shape=circle, color=black]
    312714112 -> 692404036
    692404036 [label="2.3", shape=circle, color=black]
    312714112 -> 1554874502
    1554874502 [label="3.5", shape=circle, color=black]
}
```

Παρακάτω δίνεται το ίδιο αρχείο, όπως αυτό εμφανίζεται στο kate.

```
digraph ArithmeticExpressionTree {
    fontcolor="navy";
    fontsize=20;
    labelloc="t";
    label="Arithmetic Expression"
    1028566121 [label="+", shape=circle, color=black]
    1028566121 -> 1118140819
    1118140819 [label="5.0", shape=circle, color=black]
    1028566121 -> 1975012498
    1975012498 [label="+", shape=circle, color=black]
    1975012498 -> 1808253012
    1808253012 [label="*", shape=circle, color=black]
    1808253012 -> 589431969
    589431969 [label="+", shape=circle, color=black]
    589431969 -> 1252169911
    1252169911 [label="*", shape=circle, color=black]
    1252169911 -> 2101973421
    2101973421 [label="+", shape=circle, color=black]
    2101973421 -> 685325104
    685325104 [label="3.3", shape=circle, color=black]
    2101973421 -> 460141958
    460141958 [label="6.6", shape=circle, color=black]
    1252169911 -> 1163157884
    1163157884 [label="9.2", shape=circle, color=black]
    589431969 -> 1956725890
    1956725890 [label="12.546", shape=circle, color=black]
    1808253012 -> 356573597
    356573597 [label="2.323", shape=circle, color=black]
    1975012498 -> 1735600054
    1735600054 [label="+", shape=circle, color=black]
    1735600054 -> 21685669
    21685669 [label="-", shape=circle, color=black]
    21685669 -> 2133927002
    2133927002 [label="*", shape=circle, color=black]
    2133927002 -> 1836019240
    1836019240 [label="+", shape=circle, color=black]
    1836019240 -> 325040804
    325040804 [label="33.3", shape=circle, color=black]
    1836019240 -> 1173230247
    1173230247 [label="2342.1", shape=circle, color=black]
    2133927002 -> 856419764
    856419764 [label="55.555", shape=circle, color=black]
    21685669 -> 621009875
    621009875 [label="10000.009", shape=circle, color=black]
    1735600054 -> 1265094477
    1265094477 [label="*", shape=circle, color=black]
    1265094477 -> 2125039532
    2125039532 [label="11.334", shape=circle, color=black]
    1265094477 -> 312714112
    312714112 [label="^", shape=circle, color=black]
    312714112 -> 692404036
    692404036 [label="2.3", shape=circle, color=black]
    312714112 -> 1554874502
    1554874502 [label="3.5", shape=circle, color=black]
}
```

Κώδικας για την εγγραφή ενός **dot String** σε αρχείο και κλήση του προγράμματος dot για την παραγωγή εικόνας PNG.

```
Scanner sc = new Scanner(System.in);
System.out.print("Enter expression: ");
String expr = sc.nextLine();
NodeTree tree = new NodeTree(expr);
try {
  PrintWriter pfile = new PrintWriter("ArithmeticExpression.dot");
  pfile.println(tree.toDotString());
  pfile.close();
  System.out.println("PRINT DOT FILE OK!");

  Process p = Runtime.getRuntime().exec("dot -Tpng ArithmeticExpression.dot " +
                                        "-o ArithmeticExpression.png");
  p.waitFor();
  System.out.println("PRINT PNG FILE OK!");
} catch(Exception ex) {
  System.err.println("Unable to write dotString!!!");
  ex.printStackTrace();
  System.exit(1);
}
```

Τρόπος Αποστολής

Πακετάρετε **ΜΟΝΟ** τα αρχεία java αφού τα συμπιέσετε σε ένα αρχείο ZIP με όνομα το όνομα και το ΑΕΜ κάθε μέλους της ομάδας σας (*π.χ. GiorgosThanos_1234_PeterGordon_1235.zip*). Τα αρχεία σας θα πρέπει να περιέχονται στην ιεραρχία καταλόγων που ορίζεται από το **package ce325.hw1**. Αποστείλετε την δουλειά σας με e-mail στην διεύθυνση **ce325.course@gmail.com** ως εξής:

- Τίτλος (subject): **CE325 hw01**
- συνημμένο το παραπάνω αρχείο ZIP.
- Στο σώμα του μηνύματος τα ονόματα και τα ΑΕΜ της ομάδας σας.

Εργασίες που δεν είναι συνεπείς με τους παραπάνω περιορισμούς δεν αξιολογούνται.