ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Δομές Δεδομένων - Εργασία 1

Τμήμα Πληροφορικής

Φθινοπωρινό Εξάμηνο 2018-2019

Διδάσκων: Ε. Μαρκάκης

Ουρές: Υλοποιήσεις ΑΤΔ και εφαρμογές

Σκοπός της εργασίας είναι η εξοικείωση με βασικούς αφηρημένους τύπους δεδομένων όπως οι στοίβες και οι ουρές FIFO. Η εργασία αποτελείται από 2 υλοποιήσεις ΑΤΔ (Μέρος Α) και 2 εφαρμογές (Μέρος Β και Γ). Διαβάστε προσεκτικά την εκφώνηση και τα ζητούμενα της εργασίας.

Μέρος Α [30 μονάδες]. Στο φάκελο «Εγγραφα/Εργασίες» του eclass, δίνονται οι διεπαφές StringStack και IntQueue, που δηλώνουν τις βασικές μεθόδους για μια στοίβα και μια ουρά FIFO, με στοιχεία τύπου *String* και *int* αντίστοιχα. Δημιουργήστε μία υλοποίηση των ΑΤΔ StringStack και IntQueue, δηλαδή γράψτε 2 κλάσεις που υλοποιούν τις 2 διεπαφές.

Οδηγίες υλοποίησης:

- Οι κλάσεις σας **πρέπει να λέγονται** StringStackImpl και IntQueueImpl.
- Η υλοποίηση και για τις 2 διεπαφές θα πρέπει να γίνει χρησιμοποιώντας λίστα μονής σύνδεσης.
- Κάθε μέθοδος εισαγωγής ή εξαγωγής στοιχείου θα πρέπει να ολοκληρώνεται σε χρόνο O(1), δηλαδή σε χρόνο ανεξάρτητο από τον αριθμό των αντικειμένων που είναι μέσα στην ουρά. Ομοίως, η μέθοδος size θα πρέπει να εκτελείται σε O(1).
- Όταν η στοίβα ή η ουρά είναι άδεια, οι μέθοδοι που διαβάζουν από την δομή θα πρέπει να πετάνε εξαίρεση τύπου NoSuchElementException. Η εξαίρεση NoSuchElementException ανήκει στην core βιβλιοθήκη της Java. Κάντε την import από το πακέτο **java.util**.
- Μπορείτε είτε να βασιστείτε στον κώδικα του εργαστηρίου και να τροποποιήσετε κατάλληλα τη λίστα μονής σύνδεσης που παρουσιάστηκε στο εργαστήριο 2 είτε να γράψετε εξ' ολοκλήρου τη δική σας λίστα είτε να χρησιμοποιήσετε μόνο αντικείμενα τύπου Node μέσα στην κλάση της ουράς για να φτιάξετε την λίστα. Για να αποκτήσετε

καλύτερη εξοικείωση προτείνουμε να ξεκινήσετε χωρίς τον κώδικα του εργαστηρίου και να γράψετε τις δικές σας κλάσεις (σίγουρα δεν θα χάσετε μονάδες όμως αν χρησιμοποιήσετε κάτι που έχετε δει στο εργαστήριο).

- **Προαιρετικά:** μπορείτε να κάνετε την υλοποίηση σας με χρήση generics για να μπορείτε να χειρίζεστε ουρές και στοίβες με οποιονδήποτε τύπο αντικειμένων. Υπάρχει ένα 10% bonus σε όσους χρησιμοποιήσουν generics για το Μέρος Α. Αν κάνετε χρήση generics, πρέπει να τροποποιήσετε κατάλληλα τα interfaces που σας δίνονται για να δουλεύουν για οποιονδήποτε τύπο δεδομένων.
- **Δεν επιτρέπεται να χρησιμοποιήσετε έτοιμες υλοποιήσεις δομών τύπου λίστας, στοίβας, και ουράς, από την βιβλιοθήκη της Java** (π.χ. Vector, ArrayList κλπ).

Μέρος Β [30 μονάδες]. Χρησιμοποιώντας την υλοποίηση της στοίβας του μέρους Α, γράψτε ένα πρόγραμμα-πελάτη το οποίο θα κάνει έλεγχο των ετικετών (tags) σε ένα HTML αρχείο. Σε κάθε HTML έγγραφο, οι ετικέτες οριοθετούν τμήματα κειμένου. Μια ετικέτα ανοίγματος έχει τη μορφή <tag_name> ενώ η αντίστοιχη ετικέτα κλεισίματος είναι η </tag_name>. Κάποιες απλές και συνηθισμένες ετικέτες είναι οι εξής:

- body, για να ξεκινήσει το κυρίως σώμα του εγγράφου
- h1, για την επικεφαλίδα
- center, για στοίχιση στο κέντρο
- ol, για αριθμημένη λίστα στοιχείων
- li, για ένα στοιχείο μιας λίστας

Ιδανικά, κάθε HTML έγγραφο έχει *ταιριασμένες* ετικέτες, αν δηλαδή υπάρχει κάπου η ετικέτα , θα πρέπει αργότερα στο αρχείο να εμφανίζεται η ετικέτα , που σηματοδοτεί το τέλος της στοίχισης στο κέντρο. Το ίδιο πρέπει να συμβαίνει για όλες τις ετικέτες. Επίσης, οι ετικέτες μπορεί να είναι φωλιασμένες μέσα σε ένα αρχείο και συνεπώς πρέπει να κλείνουν με την σωστή σειρά. Αν έχουμε φωλιασμένες ετικέτες ανοίγματος, θα πρέπει να κλείσει πρώτα η τελευταία ετικέτα που άνοιξε προτού κλείσουν οι υπόλοιπες. Αν και αρκετοί browsers παρουσιάζουν ανοχή στην ύπαρξη κάποιων μη ταιριασμένων ετικετών, θα θέλαμε να μην έχουμε τέτοιες ετικέτες σε αρχεία που επεξεργαζόμαστε.
Γράψτε ένα πρόγραμμα το οποίο θα διαβάζει ένα HTML αρχείο και θα αποφασίζει αν το αρχείο έχει ταιριασμένες ετικέτες.

Οδηγίες υλοποίησης:

- Το πρόγραμμα σας **πρέπει να λέγεται** TagMatching.java.
- Θα πρέπει να χρησιμοποιήσετε την υλοποίηση της στοίβας από το Μέρος Α.
- Για το διάβασμα του html αρχείου, μπορείτε να χρησιμοποιήσετε έτοιμες μεθόδους της Java. Μπορείτε επίσης να χρησιμοποιήσετε έτοιμες μεθόδους για να επεξεργαστείτε μεταβλητές τύπου String. Πέρα από το διάβασμα όμως του αρχείου, το υπόλοιπο πρόγραμμα θα πρέπει να κανει χρήση της στοίβας για να αποφασίσει για το ταίριασμα των ετικετών. Θα πρέπει επίσης να τυπώνετε αντίστοιχο μήνυμα για το αν το αρχείο έχει ή όχι ταιριασμένες ετικέτες.
- Η εργασία σας θα εξεταστεί σε html αρχεία με απλές ετικέτες ανοίγματος και κλεισίματος. Π.χ. μην ανησυχείτε για tags τυπου link . Επίσης δεν χρειάζεται να ελέγχετε αν το html αρχείο εχει άλλα συντακτικά λάθη. Ασχοληθείτε μόνο με το ταίριασμα των ετικετών.
- Το μονοπάτι για το προς εξέταση html αρχείο θα δίνεται ως όρισμα, π.χ. η εκτέλεση του προγράμματος από τη γραμμή εντολών θα πρέπει να είναι ως εξής

> java TagMatching path_to_html_file.html

Μέρος Γ [30 μονάδες]. Χρησιμοποιώντας την υλοποίηση της ουράς του μέρους Α, γράψτε ένα πρόγραμμα-πελάτη, το οποίο λύνει το εξής λογιστικό πρόβλημα: όταν θέλουμε να υπολογίσουμε το καθαρό κέρδος που προκύπτει από πώληση μετοχών, υπάρχει αμφισημία ως προς το ποιες μετοχές θεωρούμε ότι πουλάμε (όταν έχουμε μετοχές που έχουν αγοραστεί σε διαφορετικές τιμές). Για παράδειγμα, έστω ότι τη χρονική στιγμή t1, αγοράσαμε 50 μετοχές από κάποια εταιρεία, σε τιμή 25 ευρώ. Μετέπειτα, τη χρονική στιγμή t2, αγοράσαμε άλλες 40 μετοχές από την ίδια εταιρεία σε τιμή 22 ευρώ, και τη χρονική στιγμή t3, αγοράσαμε ακόμα 30 μετοχές σε τιμή 32 ευρώ. Έστω ότι αργότερα πουλάμε 110 μετοχές (από τις 120 που έχουμε) σε τιμή 30 ευρώ. Η λογιστική αρχή που χρησιμοποιείται για να καθορίσουμε το καθαρό κέρδος εκείνη τη στιγμή από την πώληση στηρίζεται στο πρωτόκολλο FIFO (τα περισσότερα λογισμικά διαχείρισης χαρτοφυλακίων χρησιμοποιούν αυτή την αρχή). Στο παράδειγμά μας επομένως, θεωρούμε ότι από τις 110 μετοχές που πουλάμε, οι πρώτες 50 προέρχονται απο αυτές που αγοράστηκαν την στιγμή t1, οι επόμενες 40 από αυτές που αγοράστηκαν την στιγμή t2, και οι επόμενες 20 από την τελευταία αγορά. Έτσι το κέρδος μας εκείνη τη στιγμή θεωρούμε ότι είναι:

$$50\ (30 - 25) + 40\ (30 - 22) + 20\ (30 - 33) = 250 + 320 - 60 = 510$$

Γράψτε ένα πρόγραμμα το οποίο θα διαβάζει από ένα .txt αρχείο μια ακολουθία από διαδοχικές συναλλαγές της μορφής:

```
buy 50 price 25
buy 40 price 22
buy 30 price 33
sell 110 price 30
buy 25 price 35
sell 30 price 40
...
```

και θα τυπώνει το συνολικό κέρδος ή ζημιά προκύπτει ακριβώς μετά από την τελευταία πώληση. Το παραπάνω αρχείο απεικονίζει τις συναλλαγές με τη σειρά που γίνονται, δηλαδή στο παράδειγμα αυτό έγιναν πρώτα 3 αγορές, μετά πώληση, μετά μια αγορά και μετά ξανά πώληση. Σημειώστε ότι δεν υπολογίζουμε στο κέρδος τυχόν μετοχές που έχουν περισσέψει και δεν έχουμε πουλήσει (αυτές θα παρθούν υπόψη σε μελλοντικές πωλήσεις).

Οδηγίες υλοποίησης:

- Το πρόγραμμά σας **πρέπει να λέγεται** *NetBenefit.java*.
- Για το διάβασμα του αρχείου εισόδου, μπορείτε να χρησιμοποιήσετε έτοιμες μεθόδους της Java για να διαβάσετε και να επεξεργαστείτε την ακολουθία των συναλλαγών από το αρχείο. Στη συνέχεια θα πρέπει να χρησιμοποιήσετε την υλοποίηση της ουράς FIFO για να υπολογίσετε το κέρδος.
- Θεωρούμε ότι σε όλες τις γραμμές του αρχείου εισόδου, ο αριθμός των μετοχών και οι τιμές είναι ακέραιοι αριθμοί. Δεν απαιτείται να ελέγξετε ότι όλα τα στοιχεία του αρχείου εισόδου είναι σε σωστή μορφή. Θα πρέπει όμως να τυπώνετε μήνυμα λάθους αν μέσα στο αρχείο υπάρχει πώληση μεγαλύτερου αριθμού μετοχών από αυτές που έχουν αγοραστεί μέχρι εκείνη τη στιγμή.
- Το μονοπάτι προς το αρχείο εισόδου θα δίνεται ως όρισμα, δηλαδή η εκτέλεση του πρόγραμματος από τη γραμμή εντολών θα γίνεται ως εξής

> java NetBenefit path_to_text_file.txt

Μέρος Δ - Αναφορά παράδοσης [10 μονάδες]. Ετοιμάστε μία σύντομη αναφορά σε pdf αρχείο (μην παραδώσετε Word ή txt αρχεία!) με όνομα project1-report.pdf, στην οποία θα αναφερθείτε στα εξής:

a. Εξηγήστε συνοπτικά πώς υλοποιήσατε τις διεπαφές στο μέρος Α (άνω όριο 2 σελίδες).
b. Για το μέρος Β, εξηγήστε πώς χρησιμοποιήσατε την υλοποίηση από το μέρος Α για να φτιάξετε το πρόγραμμα που ζητείται (άνω όριο 2 σελίδες).
c. Ομοίως για το μέρος Γ (άνω όριο 2 σελίδες).

Το συνολικό μέγεθος της αναφοράς θα πρέπει να είναι τουλάχιστον 2 σελίδες και το πολύ 6 σελίδες.

Οδηγίες Παράδοσης

Η εργασία σας θα πρέπει να μην έχει συντακτικά λάθη και να μπορεί να μεταγλωττίζεται. Εργασίες που δεν μεταγλωττίζονται χάνουν το **50%** της συνολικής αξίας.

Η εργασία θα αποτελείται από:

1. Τον πηγαίο κώδικα (source code). Τοποθετήστε σε ένα φάκελο με όνομα **src** τα αρχεία java που έχετε φτιάξει. Ενδεικτικά θα πρέπει να περιέχει (χρησιμοποιήστε τα όνοματα των κλάσεων όπως ακριβώς δίνονται στην εκφώνηση):
   - Τις διεπαφές StringStack και IntQueue, οι οποίες είναι ήδη διαθέσιμες στο eclass.
   - Όλες τις υλοποιήσεις των διεπαφών από το μέρος Α.
   - Τα προγράμματα TagMatching.java και NetBenefit.java.

   Επιπλέον, φροντίστε να συμπεριλάβετε όποια άλλα αρχεία πηγαίου κώδικα φτιάξατε και απαιτούνται για να μεταγλωττίζεται η εργασία σας. Φροντίστε επίσης να προσθέσετε επεξηγηματικά σχόλια όπου κρίνετε απαραίτητο στον κώδικά σας.

2. Την αναφορά παράδοσης

Όλα τα παραπάνω αρχεία θα πρέπει να μπουν σε ένα αρχείο zip. Το όνομα που θα δώσετε στο αρχείο αυτό θα είναι ο αριθμός μητρώου σας πχ. 3030056_3030066.zip ή 3030056.zip (αν δεν είστε σε ομάδα). Στη συνέχεια, θα υποβάλλετε το zip αρχείο σας στην περιοχή του μαθήματος «Εργασίες» στο e-class. Δεν χρειάζεται υποβολή και από τους 2 φοιτητές μιας ομάδας.

Η προθεσμία παράδοσης της εργασίας είναι Τετάρτη, 14 Νοεμβρίου 2018 και ώρα 23:59.