



Δομές Δεδομένων - Εργασία 1

Τμήμα Πληροφορικής

Φθινοπωρινό Εξάμηνο 2021-2022

Διδάσκων: Ε. Μαρκάκης

Διονύσης Ρηγάτος (3200262)

Αθανάσιος Τριφώνης (3200298)

Μέρος Α: Για την υλοποίηση της στοίβας κατασκευάσαμε την κλάση StringStackImpl η οποία εφαρμόζει το interface `StringStack`. Αρχικά δηλώνεται μία μεταβλητή αντικειμένου `Node` με όνομα `first` η οποία αναπαριστά την κορυφή της στοίβας και μία μεταβλητή `int` `size` η οποία αναπαριστά το μέγεθος της στοίβας.

Μέθοδοι:

- `push(T data)`: Δημιουργεί έναν νέο κόμβο με τα δεδομένα `data` και τον τοποθετεί στην κορυφή της στοίβας. Έπειτα αυξάνει το μέγεθος της στοίβας κατά μία μονάδα.
- `pop()`: Εμφανίζει σφάλμα `NoSuchElementException` σε περίπτωση που η στοίβα είναι άδεια. Σε διαφορετική περίπτωση αποθηκεύει τα δεδομένα του κόμβου στην κορυφή της στοίβας για να τα επιστρέψει, μετακινεί την κορυφή μία θέση πιο κάτω στην στοίβα (αφήνοντας έτσι το κόμβο που βρισκόταν εκεί να διαγραφεί από τον `garbage collector`) και μειώνει το μέγεθος της στοίβας κατά 1. Τέλος επιστρέφει τα δεδομένα που είχε αποθηκεύσει.
- `peek()`: Εμφανίζει σφάλμα `NoSuchElementException` σε περίπτωση που η στοίβα είναι άδεια. Διαφορετικά επιστρέφει τα δεδομένα του κόμβου στην κορυφή της στοίβας χωρίς να βγάλει τον κόμβο από την στοίβα.
- `size()`: Επιστρέφει το μέγεθος της στοίβας.
- `isEmpty()`: Επιστρέφει `true` αν η στοίβα είναι άδεια ή `false` αν υπάρχουν κόμβοι στην στοίβα.
- `printStack(PrintStream stream)`: Εμφανίζει μήνυμα “The stack is empty” σε περίπτωση που είναι άδεια η στοίβα, διαφορετικά εμφανίζει όλους τους κόμβους από την κορυφή μέχρι τον πάτο της στοίβας με βελάκια ενδιάμεσα και ο πάτος δείχνει `null`.

Για την υλοποίηση της ουράς κατασκευάσαμε την κλάση IntQueueImpl η οποία εφαρμόζει το interface `IntQueue`. Αρχικά δηλώνονται δύο μεταβλητές αντικειμένου

Node ως head και tail οι οποίες αναπαριστούν τα δύο άκρα της ουράς την αρχή και το τέλος αντίστοιχα. Επίσης δηλώνεται και μία μεταβλητή int size η οποία δείχνει το μέγεθος της ουράς.

Μέθοδοι:

- put(T data): Δημιουργεί έναν δείκτη old_tail που δείχνει στο τρέχον tail. Έπειτα δημιουργεί ένα νέο κόμβο με τα δεδομένα data, ο οποίος δείχνει σε null και τοποθετεί το tail στον νέο κόμβο. Αν η ουρά είναι άδεια τοποθετεί το head στον νέο κόμβο επίσης, διαφορετικά αν υπάρχουν κ άλλοι κόμβοι συνδέει την old tail με τον νέο κόμβο. Τέλος αυξάνει το μέγεθος της ουράς κατά μία μονάδα.
- get(): Αν η ουρά είναι άδεια επιστρέφει σφάλμα NoSuchElementException. Διαφορετικά επιστρέφει τα δεδομένα του κόμβου που καταδεικνύεται από το head και είναι αυτός που βρίσκεται το μεγαλύτερο διάστημα στην ουρά. Αποθηκεύει τα δεδομένα του head στην data, έπειτα μετακινεί το head στον επόμενο κόμβο (αφήνοντας έτσι τον garbage collector να διαγράψει το παλιό head), μειώνει το μέγεθος της ουράς κατά 1 και επιστρέφει την data με τα δεδομένα του κόμβου που εξήλθε από την ουρά.
- peek(): Αν η ουρά είναι άδεια επιστρέφει σφάλμα NoSuchElementException. Διαφορετικά επιστρέφει τα δεδομένα του κόμβου που βρίσκεται περισσότερο χρόνο στην ουρά (head) χωρίς να τον εξάγει.
- size(): Επιστρέφει το μέγεθος της ουράς.
- isEmpty(): Επιστρέφει true αν η ουρά είναι άδεια αλλιώς false αν υπάρχουν nodes σε αυτή.
- printQueue(PrintStream stream): Αν η ουρά είναι άδεια τυπώνει το μήνυμα "The queue is empty!". Διαφορετικά διατρέχει όλη την ουρά από το παλιότερο Node (head) μέχρι το πιο πρόσφατο (tail) και τα τυπώνει με αυτή την σειρά.

Μέρος Β: Σε αυτό το μέρος υλοποιούμε την κλάση TagMatching η οποία ελέγχει εαν ένα αρχείο HTML είναι valid/invalid βάσει του εαν όλα τα tags ανοίγουν και κλείνουν με την σωστή σειρά. Με την χρήση των BufferedReader και FileReader ανοίγουμε και διαβάζουμε το αρχείο HTML γραμμή-γραμμή, του οποίου το όνομα δώθηκε σε μορφή String από το command line (args[0]). Βεβαιώνουμε οτι το command line input είναι έγκυρο και οτι το αρχείο υπάρχει με την χρήση της Try-Except, η οποία σε αυτή την περίπτωση εμφανίζει συγκεκριμένο μήνυμα στον χρήστη. Σε πιο γενικά exceptions, εμφανίζει το ανάλογο error message.


Εφόσον το input stream ανοίξει, δημιουργούμε ένα νέο Stack Object στο οποίο θα αποθηκεύουμε τα ανοιχτά tags που διαβάζουμε και μια boolean μεταβλητή με την οποία θα ελέγχουμε την εγκυρότητα του αρχείου. Εφόσον το αρχείο περιέχει κείμενο, δημιουργούμε ένα νέο αντικείμενο Matcher και Pattern του Regex module τα οποία θα χρησιμοποιήσουμε για να αναγνωρίσουμε εάν ένα string είναι έγκυρο HTML tag.* Το group 1 θα κρατάει τα ανοιχτά tags, ενώ το group 2 τα κλειστά. Όσο υπάρχουν γραμμές στο αρχείο, κρατάμε την τρέχουσα γραμμή και την "ψάχνουμε" για tags με την χρήση Regular Expressions και τα groups που φτιάξαμε πριν. Στην περίπτωση


όπου ένα tag ανήκει στο 1^ο group, το κάνουμε push στην στοίβα. Αλλιώς, εάν το tag είναι κλεισίματος, ελέγχουμε εάν είναι όμοιο του τελευταίου ανοιχτού tag κάνοντας peek στην στοίβα και χρησιμοποιώντας String Manipulation έτσι ώστε να μπορούμε να ελέγχουμε equality. Σε αυτή την περίπτωση, εφόσον είναι όμοια, κάνουμε pop και θέτουμε το invalid tag σε false καθώς μέχρι στιγμής το αρχείο μας είναι έγκυρο. Εάν τα tags δεν ταιριάζουν, θέτουμε την invalid true και σπάμε την while έτσι ώστε να τερματίσει η ανάγνωση του αρχείου, το οποίο είμαστε πλέον βέβαιοι ότι δεν είναι έγκυρο αρχείο HTML.


Τέλος, κλείνουμε το buffer stream και εμφανίζουμε στον χρήστη ανάλογο μήνυμα το οποίο καθορίζεται από το εάν η στοίβα δεν είναι άδεια (δηλαδή υπάρχουν ανοιχτά tags ακόμα) ή το εάν η invalid είναι true.

*RegEx Analysis: ()

Γκρούπ: 1^ο: Ανοιχτά Tags, 2^ο: Κλειστά Tags

: Τα matching words πρέπει να ξεκινάνε με < και να τελειώνουν σε >, η να ξεκινάνε με </ και να τελειώνουν με > ανάλογα το είδος του tag.

: Χαρακτήρες που δεν πρέπει να βρίσκονται εντός του σετ.

: Από μία ή παραπάνω φορές.

|: OR

Μέρος Γ: Για τον υπολογισμό του κέρδους / ζημίας από τις αγορές και πωλήσεις μετοχών εκτός από την κλάση NetBenefit χρησιμοποιήθηκε επίσης η κλάση Bundle η οποία κατασκευάζει ένα αντικείμενο με μεταβλητές int price για την τιμή των μετοχών και int shares για το πλήθος των μετοχών, οι οποίες λαμβάνονται από τον constructor, μέσω string manipulation καθώς διαβάζουμε το txt αρχείο στην NetBenefit σε κάθε συναλλαγή αγοράς. Ο λόγος ήταν ώστε να αποθηκεύονται στην ουρά FIFO όλες οι αγορές μετοχών.

Μέθοδοι Bundle:

- getShares(): Επιστρέφει το πλήθος των μετοχών της συγκεκριμένης αγοράς.
- setShares(int shares): Χρησιμοποιείται για να μεταβάλλουμε το πλήθος των μετοχών μίας συγκεκριμένης αγοράς μέσα στην ουρά, από την οποία έχουν πουληθεί μετοχές.
- getPrice(): Επιστρέφει την τιμή που αγοράστηκε η κάθε μετοχή στην συγκεκριμένη αγορά.

Η κλάση NetBenefit ανοίγει το αρχείο txt και το διαβάζει γραμμή γραμμή. Κάνει split τις λέξεις στα κενά που έχουν ανάμεσα τους και τις αποθηκεύει σε ένα String array με όνομα split, στη συνέχεια μετατρέπει τα string που αντιστοιχούν στο πλήθος και στην τιμή σε integers, τα αποθηκεύει στις μεταβλητές int amountOfShares και int price αντίστοιχα και ελέγχει αν είναι έγκυρες οι τιμές τους. Σε περίπτωση που δεν είναι έγκυρες τυπώνει σφάλμα IllegalArgumentException.

Σε περίπτωση που οι παραπάνω τιμές είναι έγκυρες γίνεται έλεγχος στην πρώτη λέξη του input η οποία καθορίζει αν πρόκειται για αγορά ή πώληση. Όταν βλέπουμε αγορά εισάγουμε στην ουρά ένα αντικείμενο Bundle με το αντίστοιχο πλήθος και τιμή.

Έπειτα αυξάνουμε την μεταβλητή `ownedShares` που χρησιμοποιούμε ως μετρητή του πλήθους αγορασμένων μετοχών που έχουμε στην κατοχή μας, με το πλήθος των μετοχών που αγοράσαμε.

Όταν έχουμε πώληση κάνουμε μία σειρά επαναληπτικών πράξεων μέσα σε `while loop`, καθώς θα πρέπει για τον αριθμό μετοχών που θέλουμε να πουλήσουμε να πηγαίνουμε τμηματικά από τις παλιότερες μετοχές στην ουρά και να πουλάμε, να υπολογίζουμε το κέρδος / ζημία με ως την διαφορά στην τιμή πώλησης με την τιμή αγοράς και να μειώνουμε την μεταβλητή `ownedShares` ανάλογα, όπως και την μεταβλητή `amountOfShares` που είναι οι μετοχές προς πώληση οι οποίες αφού υπολογιστεί ένα μέρος τους με βάση την παλιότερη αγορά, θα πρέπει να συνεχιστεί η διαδικασία με επόμενες αγορές μέχρι να πουληθούν όλες οι μετοχές στην συγκεκριμένη αγορά. Σε κάθε σημείο στο οποίο πουλάμε όλες τις μετοχές από την παλιότερη αγορά τότε εξάγουμε την αγορά από την ουρά. Όταν σε μία αγορά έχουμε πουλήσει ένα μέρος από τις μετοχές της, ενημερώνουμε το πλήθος τους μέσω της μεθόδου `setShares(int shares)`.

Σε περίπτωση που υπάρξει κάποια συναλλαγή πώλησης με πλήθος μετοχών μεγαλύτερο από το συνολικό πλήθος μετοχών που έχουμε στην διάθεση μας αγορασμένες το πρόγραμμα εμφανίζει ένα σχετικό σφάλμα πως το ποσό που πάμε να πουλήσουμε είναι μεγαλύτερο από τις διαθέσιμες μετοχές που έχουμε.

Αφού τρέξουμε όλες τις συναλλαγές εμφανίζεται το `int profit` ως ακέραιος αριθμός που αναπαριστά το κέρδος ή την ζημία που υπάρχει από τις συναλλαγές.