1η Εργασία Δομές Δεδομένων - Report

Μιλτιάδης Τσόλκας : 3200213

Χαράλαμπος Καρακώστας : 3200065

ΜΕΡΟΣ Α

Για την υλοποίηση των διεπαφών, σκεφτήκαμε ότι κάθε διεπαφή θα δημιουργεί μία λίστα που λειτουργεί είτε ως ουρά (**IntQueueImpl** κάνοντας implement την **IntQueue** που δόθηκε και τις μεθόδους της) είτε ως στοίβα (**StringStackImpl** κάνοντας implement την **StringStack** που δόθηκε και τις μεθόδους της). Σε κάθε μία από τις διεπαφές οι κόμβοι της λίστας υλοποιούνται με private κλάσεις που ορίζουμε σε κάθε διεπαφή, ανάλογα με το αν το περιεχόμενο που ζητείται πρέπει να είναι ακέραιος ή συμβολοσειρά. Η υλοποίηση μας δεν έχει πραγματοποιηθεί με χρήση generics.

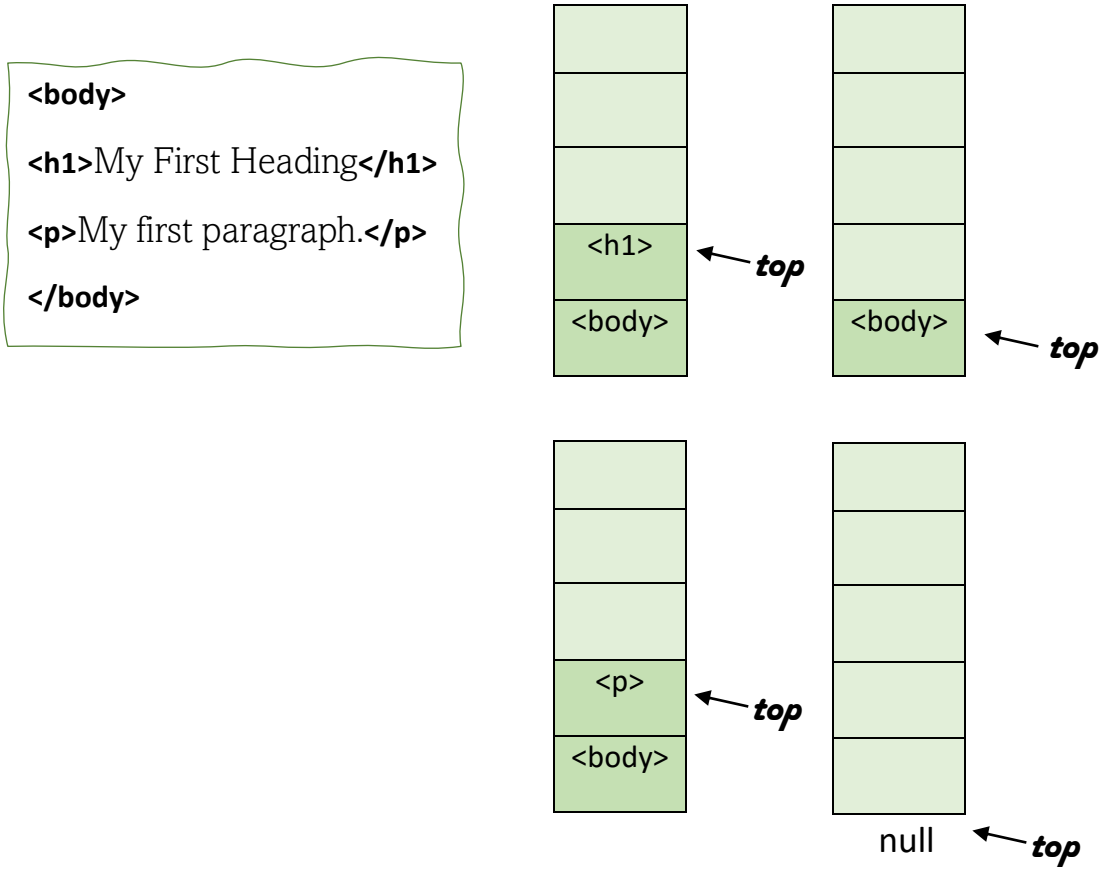
Όσον αφορά την υλοποίηση της ουράς δημιουργούμε αρχικά μία private κλάση Node, ώστε να φτιάχνουμε αντικείμενα που να λειτουργούν ως κόμβοι στους οποίους θα αποθηκεύονται integer τιμές. Η μέθοδος **isEmpty** επιστρέφει λογική τιμή True ή False ανάλογα με το αν το head είναι κενό. Με την μέθοδο **put** γίνεται εισαγωγή στοιχείου στο tail της ουράς, ενώ η **get** αφαιρεί στοιχείο και το επιστρέφει από το head της ουράς αφού πρώτα ελέγξει ότι δεν είναι άδεια, αλλιώς έχουμε εξαίρεση. Η μέθοδος **peek** επιστρέφει το στοιχείο που βρίσκεται στο head της ουράς αν η ουρά δεν είναι άδεια, ειδάλλως έχουμε εξαίρεση. Η μέθοδος **printQueue** εκτυπώνει τα στοιχεία της ουράς από το head ως το tail ή ένα μήνυμα ότι είναι άδεια. Τέλος, η μέθοδος **size** επιστρέφει το μέγεθος της ουράς το οποίο ανανεώνεται κάθε φορά που προστίθεται ή αφαιρείται ένα στοιχείο στην ουρά, έτσι ώστε να μη χρειαστεί να την διαπεράσει για να ολοκληρωθεί σε χρόνο O(1) ο υπολογισμός του μεγέθους.

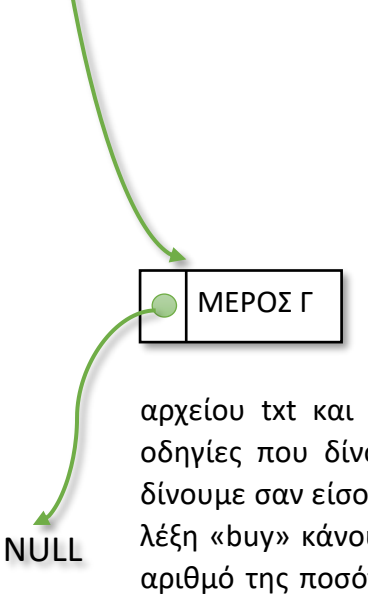
Σχετικά με την υλοποίηση της στοίβας η κλάση **Node** δημιουργείται έτσι ώστε τα αντικείμενα Node να αποθηκεύουν τιμές τύπου String. Οι μέθοδοι **isEmpty**, **peek**, **printStack** και **size** προσαρμόζουν την ίδια υλοποίηση σε στοίβα με αυτήν που πραγματοποιούν οι αντίστοιχες μέθοδοι που αναλύσαμε παραπάνω για την ουρά. Η μέθοδος **push** κάνει ώθηση στοιχείου στην κορυφή της στοίβας, ενώ η μέθοδος **pop** κάνει απώθηση του αντίστοιχου στοιχείου, αν βέβαια η στοίβα δεν είναι κενή και στις 2 περιπτώσεις.

ΜΕΡΟΣ Β

Για το μέρος Β φτιάξαμε το αρχείο **TagMatching.java** το οποίο χρησιμοποιεί την **StringStackImpl** για να δημιουργεί στοίβα στην οποία θα αποθηκεύονται τα αντίστοιχα tag του HTML αρχείου που θα ανοίγει για να εξετάσει την ορθότητα του προγράμματος. Χρησιμοποιούμε τις οδηγίες που δίνονται στην εκφώνηση για τα **command line arguments**, για να

δίνουμε σαν είσοδο το όνομα του αρχείου. Σκεφτήκαμε πως πρέπει, αφού αναγνωρίσουμε ποιες από τις λέξεις του HTML αρχείου είναι tag, να τις αποθηκεύουμε κάθε φορά με μία εντολή ώθησης στην κορυφή της στοίβας που έχουμε φτιάξει στην αρχή του κώδικα. Ελέγχουμε για κάθε tag αν είναι tag ανοίγματος ή κλεισίματος. Επειδή κάθε tag που ανοίγει τελευταίο πρέπει να κλείνει πρώτο, εξού και η χρήση στοίβας της οποίας η λογική είναι **Last In First Out (LIFO)**, το κάθε tag που διαβάζουμε αν είναι ανοίγματος το ωθούμε στην στοίβα ενώ αν είναι κλεισίματος ελέγχουμε αν το τελευταίο tag που ωθήθηκε στην κορυφή της στοίβας είναι το αντίστοιχο tag ανοίγματος. Αν υπάρχει αντιστοιχία τότε γίνεται απώθηση του tag από την κορυφή της στοίβας, αλλιώς δεν γίνεται απώθηση με αποτέλεσμα να μην αδειάζει ποτέ η στοίβα και άρα να καταλαβαίνουμε στο τέλος του προγράμματος αν το HTML αρχείο είναι ορθό ή όχι (δηλαδή αν κλείνουν όλα τα tag που ανοίγουν).





Για το μέρος Γ δημιουργήσαμε το αρχείο **NetBenefit.java** το οποίο χρησιμοποιεί την **IntQueueImpl** για να δημιουργεί ουρά στην οποία θα αποθηκεύονται τα αριθμητικά στοιχεία του λογιστικού αρχείου txt και θα κάνει τους αντίστοιχους υπολογισμούς. Χρησιμοποιούμε τις οδηγίες που δίνονται στην εκφώνηση για τα **command line arguments,** για να δίνουμε σαν είσοδο το όνομα του αρχείου. Όταν η εκάστοτε γραμμή ξεκινάει με την λέξη «buy» κάνουμε προσθήκη σε μία ουρά που έχουμε δημιουργήσει πρώτα τον αριθμό της ποσότητας και μετά τον αριθμό της τιμής. Εάν η γραμμή ξεκινάει από «sell» αυτό σημαίνει ότι θα πρέπει να πουληθεί η ποσότητα των μετοχών (από τις μετοχές που βρίσκονται στην ουρά) που αναφέρεται μετά το «sell», στην τιμή που αναφέρεται μετά το «price», με την σειρά με την οποία μπήκαν, εξού και η χρήση ουράς της οποίας η λογική είναι **First In First Out (FIFO).** Ύστερα γίνεται αφαίρεση από την ουρά των μετοχών που πουλήθηκαν και της τιμής τους, ενώ θεωρήσαμε ότι πρέπει να παραμένουν όσες περίσσεψαν και η τιμή τους.

BUY 50 PRICE 25

SELL 30 PRICE 40