Data Structures.

1η Εργασία στις Δομές Δεδομένων

Μέρος Δ – *Report*

Χρήστος Πατρινόπουλος

3200150

Αλβιόνα Μάντσο

3200098

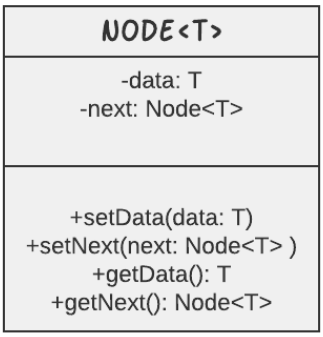
ΜΕΡΟΣ Α:

Υλοποιήθηκε το interface*: LinkedListInterface*

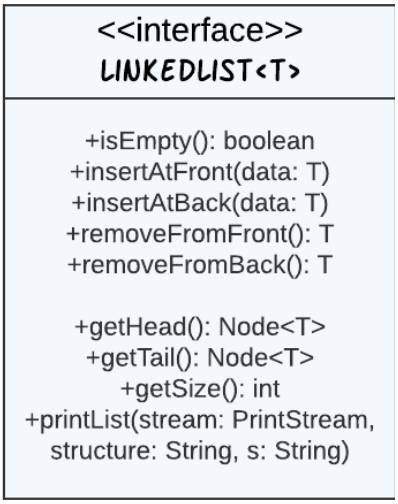
Υλοποιήθηκαν οι κλάσεις: *Node, LinkedList, StringStackImpl, IntQueueImpl*

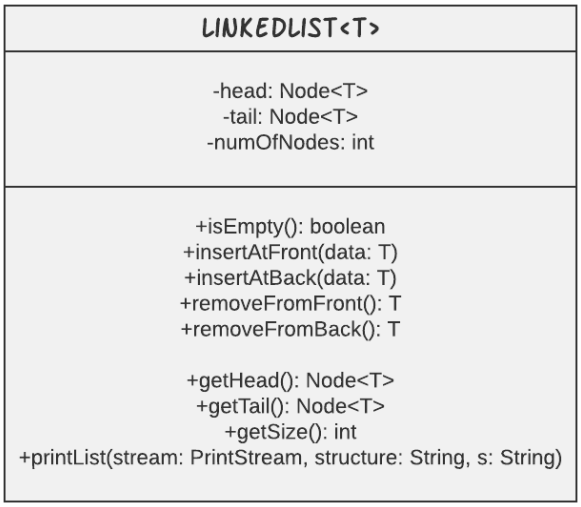
Τροποποιήθηκαν τα interfaces: *StringStack, IntQueue* για τη χρήση των τύπων **Generics** της Java.

Node: (generic) κλάση δημιουργίας κόμβων και απαραίτητες μέθοδοι.

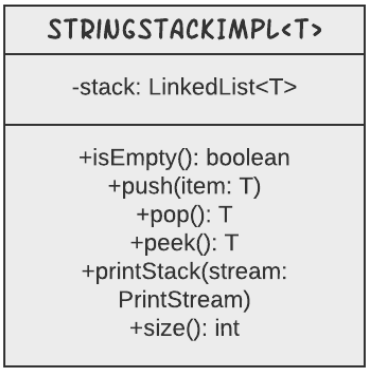


LinkedListInterface – LinkedList: υλοποίηση (generic) συνδεδεμένης λίστας με χρήση κόμβων της κλάσης Node. Χρησιμοποιήσαμε ως δείκτες πρώτου και τελευταίου κόμβου, τα head και tail αντίστοιχα. Υλοποιήθηκαν αναγκαίες μέθοδοι για είσοδο και έξοδο δεδομένων από την αρχή και το τέλος της λίστας, πρόσβαση στους κόμβους head και tail, έλεγχο κενής λίστας και εκτύπωσή της. Τέλος υλοποιήθηκε μέθοδος επιστροφής του μεγέθους της λίστας που τρέχει σε χρόνο O(1).

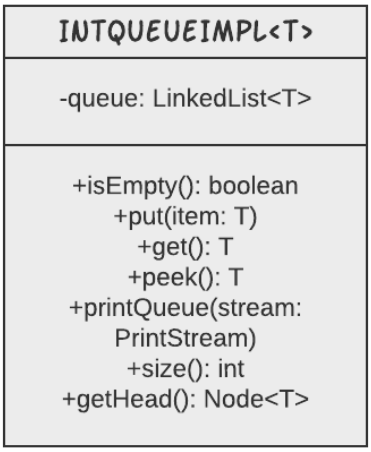


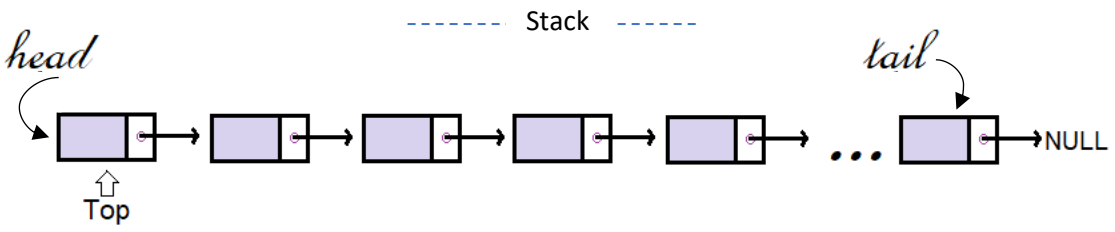


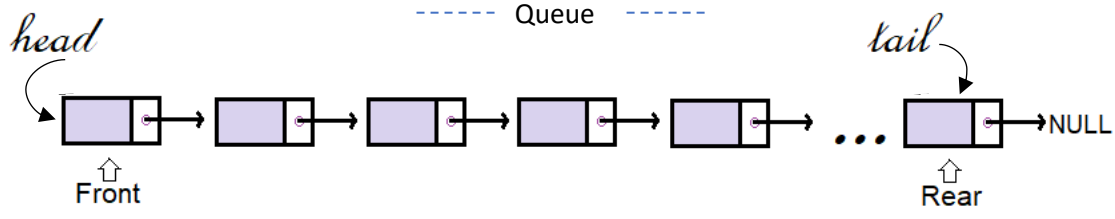
StringStackImpl: υλοποίηση (generic) στοίβας με χρήση αντικειμένου συνδεδεμένης λίστας (LinkedList object). Θεωρήσαμε ως κορυφή της στοίβας την αρχή (front) της λίστας, ώστε η ώθηση και η απώθηση δεδομένων να γίνονται μόνο από την αρχή και σε χρόνο O(1).



IntQueueImpl: υλοποίηση (generic) ουράς με χρήση αντικειμένου συνδεδεμένης λίστας (LinkedList object). Θεωρήσαμε ως θέση εισαγωγής της ουράς το τέλος της λίστας και ως θέση εξαγωγής την αρχή της λίστας, ώστε και οι δύο διαδικασίες να τρέχουν σε χρόνο O(1). Σε σχέση με το αντίστοιχο interface, προστέθηκε μέθοδος που επιστρέφει τον κόμβο αρχής της ουράς (κεφαλή της λίστας) χωρίς να τον αφαιρεί από την ουρά (χρήσιμο στο μέρος Γ της εργασίας).







*Το όνομα-path του αρχείου δίνεται ως argument στη γραμμή εντολών κατά την εκτέλεση.

Π.χ: java TagMatching tags.html

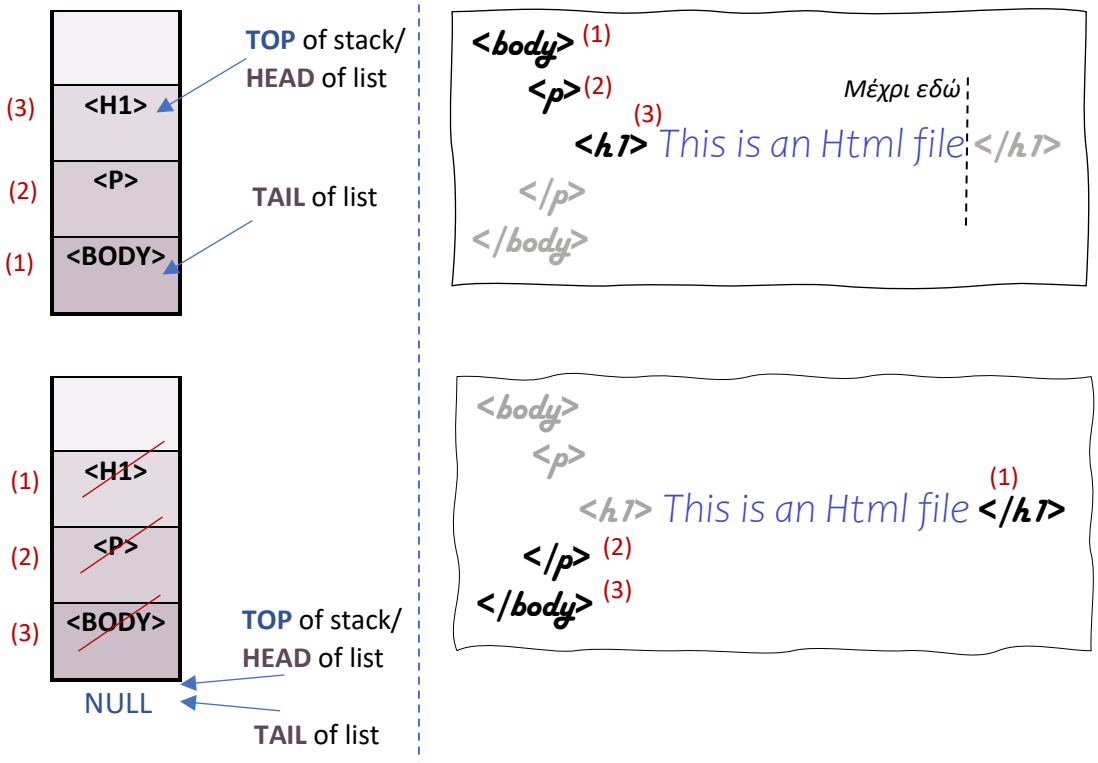
ΜΕΡΟΣ Β:

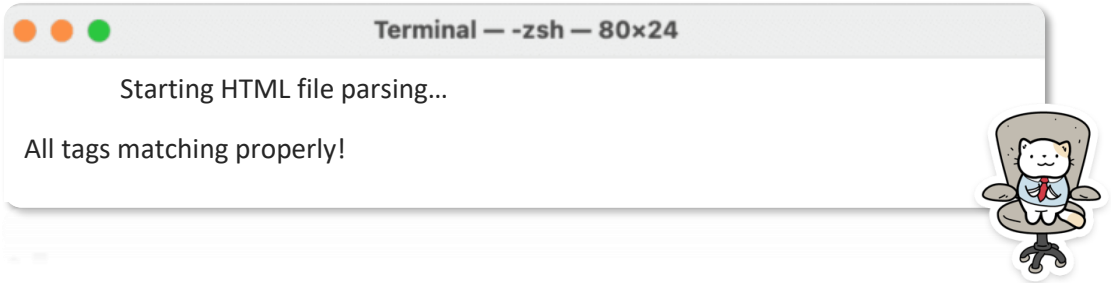
Υλοποιήθηκε η κλάση: *TagMatching**

TagMatching: Χρησιμοποιήσαμε αντικείμενο της κλάσης StringStackImpl ως δομή με είσοδο String δεδομένων. Η είσοδος στη στοίβα είναι τα opening tags του HTML “<>”, με τη σειρά που διαβάζονται από το αρχείο, εκμεταλλευόμενοι το γεγονός ότι τα tags που ανοίγουν τελευταία πρέπει να κλείσουν πρώτα (λειτουργία LIFO της στοίβας).

Όταν εντοπίζεται closing tag “</ >”, θα πρέπει να βρίσκεται στην κορυφή της στοίβας το αντίστοιχο opening tag. Σε περίπτωση που η στοίβα είναι άδεια, το πρόγραμμα τερματίζει και ενημερώνει το χρήστη με το μήνυμα «Tags are not correctly placed!». Αν όχι, ελέγχεται μέσω της peek() το opening tag στην κορυφή και αν δεν ταιριάζουν εμφανίζει το ίδιο μήνυμα, αλλιώς γίνεται ταίριασμα και άρα απώθηση του opening tag. Αν το πρόγραμμα τερματίσει χωρίς την ύπαρξη αταίριαστων tags στο HTML αρχείο, τυπώνεται το μήνυμα «All tags matching properly!».

Ελέγχεται κατάλληλα η ορθότητα ανοίγματος και κλεισίματος του HTML αρχείου που δίνεται στο πρόγραμμα.





*Το όνομα-path του αρχείου δίνεται ως argument στη γραμμή εντολών κατά την εκτέλεση.

Π.χ: java NetBenefit shares.txt

ΜΕΡΟΣ Γ:

Υλοποιήθηκε η κλάση: *NetBenefit**

NetBenefit: Χρησιμοποιήσαμε αντικείμενο της κλάσης IntQueueImpl ως δομή με είσοδο int δεδομένων. Παρακάτω αναφέρουμε την 1η μας σκέψη για την υλοποίηση και στη συνέχεια τη **2η ιδέα** που είναι και εκείνη η οποία προτιμήθηκε λόγω των πλεονεκτημάτων της.

1η ιδέα υλοποίησης

(όχι βέλτιστη, οδηγεί σε σπατάλη μνήμης λόγω υπέρογκου πλήθους στοιχείων)

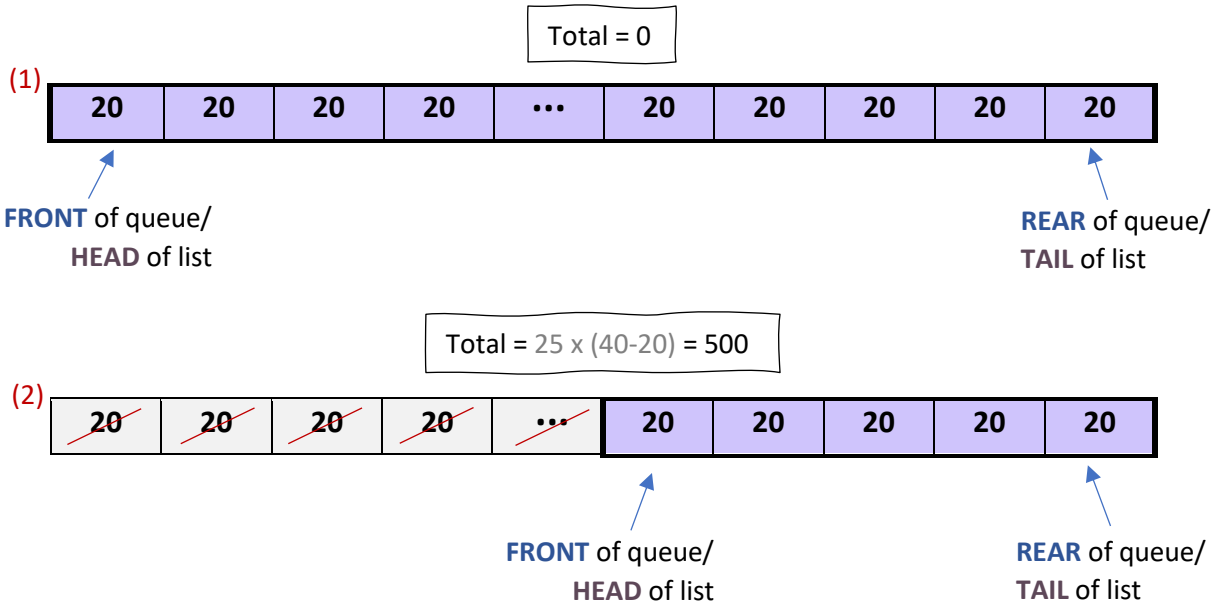
Για κάθε συναλλαγή αγοράς, εισάγεται ως δεδομένο στην ουρά η αξία της μετοχής, τόσες φορές όσες και ο αριθμός των μετοχών που αγοράζονται.

(1) Π.χ. buy ***30*** price ***20*** ⇨ ***30*** φορές εισαγωγή του αριθμού «***20***»

Για κάθε συναλλαγή πώλησης, εξάγεται από την ουρά ο αριθμός των μετοχών που πωλούνται και ενημερώνεται η μεταβλητή «total» που αντιστοιχεί στο καθαρό κέρδος/ζημία. Αν οι μετοχές δεν επαρκούν -χρήση της size()- τότε τερματίζεται το πρόγραμμα και ενημερώνεται ο χρήστης με το μήνυμα «Not sufficient shares; transaction cannot occur! Ending NetBenefit...».

(2) Π.χ. sell ***25*** price ***40*** ⇨ ***25*** φορές εξαγωγή & total = total + (***40*** – ***20***)

Κατά τον τερματισμό του προγράμματος εμφανίζεται το συνολικό καθαρό κέρδος/ζημία και τυπώνεται το μήνυμα «Profit: “total” / Loss: “total” / No profit/loss: 0».



2η ιδέα υλοποίησης

(με κωδικοποίηση, εξαιρετικά καλύτερη η διαχείριση της μνήμης)

Για κάθε συναλλαγή αγοράς, εισάγεται ως δεδομένο στην ουρά πρώτα το πλήθος των αγοραζόμενων μετοχών και στη συνέχεια η τιμή της μονάδας (2 κόμβοι). Παράλληλα κρατάμε το συνολικό αριθμό των μετοχών που αγοράζονται (διαθέσιμες μετοχές).

(1) Π.χ. (a) buy ***50*** price ***15***, (b) buy ***30*** price ***20***

Για κάθε συναλλαγή πώλησης, ελέγχεται αν οι αγορασμένες (διαθέσιμες) μετοχές επαρκούν.

Αν ναι, επαναληπτικά εξάγονται οι κόμβοι του αριθμού των μετοχών και της αντίστοιχης τιμής των μετοχών (2 κόμβοι κάθε φορά), έως ότου οι μετοχές που απομένουν για να εξαχθούν είναι είτε λιγότερες από τον αριθμό μετοχών που περιλαμβάνονται στην αρχή της ουράς, είτε 0 το πλήθος.

- Στην πρώτη περίπτωση, ανανεώνουμε τον αριθμό των μετοχών που βρίσκονται στην αρχή της ουράς, αφαιρώντας δηλαδή μόνο τον αριθμό των μετοχών που απομένουν να αγοραστούν. Δεν επηρεάζεται ο κόμβος της τιμής.
- Στη δεύτερη περίπτωση, 0 το πλήθος, τελειώνει η διαδικασία.

Ενημερώνεται η μεταβλητή «total» που αντιστοιχεί στο καθαρό κέρδος/ζημία, καθώς και οι διαθέσιμες προς πώληση μετοχές.

(2) Π.χ. sell ***60*** price ***25***

Αν όχι, οι μετοχές δεν επαρκούν και τότε τερματίζεται το πρόγραμμα και ενημερώνεται ο χρήστης με το μήνυμα «Not sufficient shares; transaction cannot occur! Ending NetBenefit...».

Κατά τον τερματισμό του προγράμματος εμφανίζεται το συνολικό καθαρό κέρδος/ζημία και τυπώνεται το μήνυμα «Profit: “total” / Loss: “total” / No profit/loss: 0».

Ελέγχεται κατάλληλα η ορθότητα ανοίγματος και κλεισίματος του αρχείου που δίνεται στο πρόγραμμα.