Εργασία 1

<u>Συντελεστές</u>

Καραγιάννης Γεώργιος (3190074)

Τάτας Αλέξανδρος (3190196)

Μέρος Α(LIFO)

Στην υλοποίηση της στοίβας από string χρησιμοποιήσαμε το πρωτόκολλο LIFO. Συγκεκριμένα, υλοποιήσαμε τις μεθόδους isEmpty(έλεγχος αν η στοίβα είναι κενή), push(προώθηση ενός string object στην στοίβα), pop(αφαιρεί κάθε φορά και επιστρέφει το string που βρίσκεται στην κορυφή της στοίβας), peek(μονάχα επιστρέφει το στοιχείο κορυφής για πληροφόρηση), printStack(τυπώνει κάθε φορά τα στοιχεία της στοίβας με την σειρά που τοποθετηθήκαν) και size(επιστρέφει το μέγεθος σε στοιχεία της στοίβας ανά δεδομένη στιγμή). Αναλυτικότερα, για κάποιες βασικές μεθόδους:

- isEmpty():Ελέγχει αν ο δείκτης στοίβας δεν δείχνει σε κανένα στοιχείο και έτσι καταλαβαίνουμε αν είναι άδεια.
- push():Αν η στοίβα είναι άδεια βάζει το δείκτη στοίβας να δείχνει το πρώτο στοιχείο και αυτός να έχει σαν επόμενο κόμβο το “τίποτα”. Αν η στοίβα περιέχει ήδη στοιχείο βάζουμε το νέο στοιχείο να κοιτάει(μέσω του next) το παλιό και μεταφέρουμε τον δείκτη στοίβας στο νέο στοιχείο.(Ταυτόχρονα, ενημερώνεται και το πεδίο size).
- pop():Αν η στοίβα είναι κενή την ώρα που πάμε να αφαιρέσουμε στοιχείο πετιέται εξαίρεση κενής στοίβας. Αν η στοίβα περιέχει >=1 στοιχεία και πάμε να αφαιρέσουμε το τελευταίο που προστέθηκε προηγουμένως τότε βάζουμε τον δείκτη στοίβας να δείχνει στο προ-τελευταίο στοιχείο(μέσω πεδίου next) και επιστρέφουμε τα δεδομένα του κατηργημένου κόμβου. (Ταυτόχρονα, ενημερώνεται και το πεδίο size).
- peek(): Αν η στοίβα είναι κενή πετιέται μήνυμα εξαίρεσης κενής στοίβας. Αντίθετα, επιστρέφονται τα δεδομένα του κόμβου στον οποίο δείχνει ο δείκτης στοίβας.

Μέρος Α(FIFO)

Στην υλοποίηση της ουράς από integers χρησιμοποιήσαμε το πρωτόκολλο FIFO. Συγκεκριμένα, υλοποιήσαμε τις μεθόδους isEmpty(έλεγχος αν η ουρά είναι κενή), put(προσθέτει έναν ακέραιο στην ουρά), get(αφαιρεί τον παλαιότερο ακέραιο από την ουρά και τον επιστρέφει), peek(επιστρέφει τον παλαιότερο ακέραιο της ουράς χωρίς να τον καταργεί), printQueue(τυπώνει την ουρά ανά πάσα στιγμή με τα στοιχεία στην σειρά που είναι τοποθετημένα), size(επιστρέφει το μέγεθος της ουράς αν πάσα στιγμή).Αναλυτικότερα, για κάποιες βασικές μεθόδους:

- isEmpty():Ελέγχει αν ο δείκτης first δείχνει σε κάποιο στοιχείο και ανάλογα επιστρέφει true η false(αφού αν ο first ή ο last δείχνουν σε null τότε δεν υπάρχει στοιχείο στην ουρά).
- put():Αν η ουρά είναι κενή βάζουμε τον first και τον last να δείχνουν στο στοιχείο που προσθέσαμε(αφού θα είναι και μοναδικό προς το παρόν).Αντίθετα, βάζουμε το στοιχείο που δείχνει ο last να έχει ως επόμενο(μέσω του next πεδίου) τον ακέραιο που εμείς προσθέτουμε και έπειτα αλλάζουμε τον δείκτη last ώστε να δείχνει στο νέο στοιχείο. (Ταυτόχρονα, ενημερώνεται και το πεδίο size).
- get():Αν η ουρά είναι κενή τότε πετιέται μήνυμα εξαίρεσης κενής ουράς. Αντίθετα, βάζουμε τον δείκτη first να δείχνει στο επόμενο στοιχείο από αυτό που έδειχνε μέχρι προηγουμένως ο first(έτσι καταργείται ο παλαιότερος κόμβος). (Ταυτόχρονα, ενημερώνεται και το πεδίο size).
- peek(): Αν η ουρά είναι κενή τότε πετιέται μήνυμα εξαίρεσης κενής ουράς. Αντίθετα, επιστρέφονται τα δεδομένα του κόμβου στον οποίο δείχνει ο first.

***Να σημειωθεί ότι η υλοποίηση της κλάσης Node έγινε με χρήση generics τα οποία εκμεταλλεύονται οι IntQueue και StringStack για να χρησιμοποιήσουν κόμβους με δεδομένα τύπου int και string αντίστοιχα**

Μέρος Β

Για το μέρος Β χρησιμοποιήσαμε την δομή StringStack ώστε να υλοποιήσουμε ένα μικρό πρόγραμμα που ταυτοποιεί εάν ένα html αρχείο έχει matching tags. Αρχικά, χρησιμοποιήσαμε την κλάση Scanner ώστε να διαβάσουμε το αρχείο html και να το μετατρέψουμε σε string ώστε να μπορέσουμε να το χειριστούμε κατάλληλα. Τώρα, πάνω στο string χρησιμοποιούμε regular expressions για να μπορέσουμε να απομονώσουμε τα tags που περιέχονται στο string. Τα tags μπαίνουν με την σειρά που εμφανίζονται σε μια ουρά από strings(έχουμε φτιάξει και την SupportQueue που είναι κλώνος της IntQueue αλλά για String data και με μια επιπλέον βοηθητική μέθοδο peekStart). Αυτό που ουσιαστικά κάνουμε διατρέχοντας την ουρά (με next)είναι όταν βρίσκουμε opening tag να κάνουμε push το tag σε μια στοίβα από strings και όταν πέτυχουμε closing tag τότε να κάνουν pop από την στοίβα το νεότερο στοιχείο. Αν στο τέλος η στοίβα μείνει άδεια τότε αυτό σημαίνει ότι όλα τα tags είναι matching.

Μέρος Γ

Για το μέρος Γ χρησιμοποιήσαμε την δομή IntQueue ώστε να υλοποιήσουμε τον υπολογιστή κέρδους η ζημίας από την αγοραπωλησία μέτοχων. Αρχικά, διαβάζουμε το txt αρχείο γραμμή προς γραμμή και ανάλογα με το αν το πρώτο γράμμα της σειράς είναι ‘b’(=buy) διαχειριζόμαστε αγορά μετοχών ενώ αν είναι ‘s’(=sell) τότε διαχειριζόμαστε πώληση μετοχών. Για την απομνημόνευση των ποσοτήτων μετοχών και των τιμών τους διατηρούμε δυο παράλληλες ουρές (stockAmmountQueue για την ποσότητα μετοχών που αγοράσαμε σε μια συγκεκριμένη χρονική στιγμή και stockPriceQueue για την αντιστοιχούμενη τιμή της μονάδας). Οπότε, για κάθε αγορά x μετοχών με y τιμής μετοχής κάνουμε put στις δυο ουρές x και y αντίστοιχα(ώστε η αντιστοιχία κόμβων να είναι 1-1). Αφότου γίνει αυτό ενημερώνουμε το πεδίο balance με το δαπανηθέν ποσό(το Balance ξεκινάει με τιμή 0 για να μπορέσουμε στο τέλος να συμπεράνουμε αν είμαστε συν η πλην και ποσό). Στην περίπτωση που συναντήσουμε γραμμή του txt που απευθύνεται σε πώληση μετοχών τότε κάνουμε get() και από τις δυο ουρές(ειδικά από την ammount) όλο το σύνολο των μετοχών(καθώς και την αντίστοιχη τιμή) που αγοράστηκαν την παλαιότερη στιγμή και υπολογίζουμε τα έσοδα που

προέκυψαν από την πώληση προσθέτοντας τα στο balance. Πρέπει να σημειωθεί ότι σε περίπτωση που αν το ποσό που απομένει να πουληθεί είναι μικρότερο από το επόμενο σε σειρά πακέτο μετοχών που αγοράστηκε τότε πρέπει να αφαιρέσουμε από εκείνο το πακέτο(παλαιότερο κόμβο της stockAmmountQueue) μόνο το πλήθος μετοχών που μας απομένει για πώληση.

***Να σημειωθεί επίσης ότι το τρέξιμο του μέρους β και γ γίνεται μέσω cmd και γι' αυτό χρησιμοποιούμε PATH για να περάσουμε όρισμα στο πρόγραμμα**