Δομές Δεδομένων – Εργασία 1

Τμήμα Πληροφορικής
Φθινοπωρινό Εξάμηνο 2021-2022
Υπεύθυνοι φοιτητές:
Μαρία Κονταράτου (3200078)
Γεώργιος Κουμουνδούρος (3200083)
Διδάσκων: Ε.Μαρκάκης

ΑΘΗΝΑ, ΝΟΕΜΒΡΙΟΣ 2021

ΜΕΡΟΣ Δ – ΑΝΑΦΟΡΑ ΠΑΡΑΔΟΣΗΣ

Α. Επεξήγηση διεπαφών μέρους Α

Για να υλοποιήσουμε τις μεθόδους, χρησιμοποιήσαμε την class Node, την οποία είχαμε δουλέψει στο 2ο εργαστήριο για λίστες.

StringStackImpl:

- public boolean is Empty():
  Επιστρέφει το head της εκάστοτε μονής λίστας, η οποία αφού είναι κενή θα είναι null

- public void push(String item)
  Με όρισμα κάποιο string, η μέθοδος σχηματίζει ένα νέο node με τα στοιχεία απ'το προαναφερθέν string. Σκοπός είναι να βάλει το item στη στοίβα. Αν είναι ήδη κενή, εύκολα το τοποθετεί ως πρώτο στοιχείο. Αν πάλι όχι, το τοποθετεί πριν τον κόμβο. Σε κάθε περίπτωση, αυξάνει και το μέγεθος κατά 1.

-public String pop() throws NoSuchElementException()
  Με σκοπό να αφαιρέσει και να επιστρέψει το item στο πάνω μέρος της στοίβας, αν η λίστα είναι κενή, πετάει NoSuchElementException(). Αλλιώς, λέμε στον δείκτη να δείχνει στον κόμβο που θέλουμε να αφαιρέσουμε και επιστρέφουμε το item (data) του κόμβου, αφαιρώντας φυσικά και το μέγεθος κατά 1.

-public String peek() throws NoSuchElementException()
  Θέλουμε απλά να επιστρέψουμε το πρώτο στοιχείο στην κορυφή της στοίβας. Αν η λίστα είναι κενή, πετάει NoSuchElementException(). Αλλιώς επιστρέφει το item του πρώτου κόμβου.

-public void printStack(PrintStream stream)
  Τυπώνει τα δεδομένα ένα προς ένα όσο είναι γεμάτη η στοίβα και μετά προχωράει στο επόμενο.

-public int size()
  Επιστρέφει το μέγεθος της στοίβας

IntQueueImpl:

- public boolean is Empty():
  Επιστρέφει το head της εκάστοτε ουράς, η οποία αφού είναι κενή θα είναι null

- public void put(int item)
  Δημιουργεί ένα node με τα δεδομένα item που έχουν δωθεί. Αν η ουρά είναι άδεια, τότε δημιουργεί έναν κόμβο βάση του Node που έφτιαξε. Αλλιώς, βάζει τον δείκτη του tail να δείχνει στο Node που κατασκευάσαμε. Επίσης, αυξάνει το μέγεθος κατά 1.

-public String get() throws NoSuchElementException()
  Πάντα με σκοπό να αφαιρέσουμε το τελευταίο στοιχείο της ουράς, αν είναι άδεια πετάει NoSuchElementException(). Αν πάλι όχι, επιστρέφει τα δεδομένα του κόμβου, αλλάζοντας τον δείκτη για να δείχνει στον κόμβο που θέλουμε να διαγράψουμε.

-public String peek() throws NoSuchElementException()
  Επιστρέφει γενικά τα δεδομένα του κόμβου, εκτός αν η ουρά είναι κενή στην περίπτωση που πετάει NoSuchElementException().

-public void printQueue(PrintStream stream)
  Τυπώνει τα δεδομένα των κόμβων ένα προς ένα όσο είναι γεμάτη η ουρά και μετά προχωράει στο επόμενο κόμβο.

-public int size()
  Επιστρέφει το μέγεθος της ουράς

B.Tag Matching

Για την υλοποίηση του μέρους Β χρησιμοποιήσαμε την στοίβα StringStackImpl του Α ερωτήματος. Αρχικά το πρόγραμμα-πελάτη δέχεται ένα αρχείο html από τον χρήστη και το διαβάζει γραμμή-γραμμή ψάχνοντας για tags της html. Με το StringTokenizer όταν εντοπίζεται το σύμβολο “<” με το οποίο ξεκινάνε όλα τα tags της html, η σειρά διασπάται σε tokens. Με την μεταβλητή tok κρατάμε το κομμάτι της σειράς από το “<” και μετά και έτσι με την επανάληψη while ελέγχουμε έναν έναν τους χαρακτήρες ώσπου να εντοπίσουμε το “>”. Με το sum κρατάμε τους χαρακτήρες από το “<” μέχρι το “>”(π.χ. body>) , αν ο πρώτος χαρακτήρας μετά το “<” είναι διάφορος του “/” τότε κάνουμε push στη λίστα το sum αλλιώς αν το sum που έχουμε ισοδυναμεί με το τελευταίο στοιχείο που έγινε push στη λίστα(+ αυτό το σύμβολο “/” γιατί τα tags κλεισίματος μετά το “<” περιέχουν αυτό το σύμβολο, π.χ. <body> </body>) τότε έχει ταιριαστεί σωστά ένα ζεύγος tags. Τέλος, αν μετά το διάβασμα του html file και αν δεν έχει παρεμβληθεί κάποιο λάθος, όπου εμφανίζεται το μήνυμα "Error reading line ...", τότε αν η στοίβα είναι άδεια αυτό σημαίνει πως τα tags του αρχείου είναι ταιριασμένα σωστά και εμφανίζεται το μήνυμα "tags are matching", αλλιώς αν δεν είναι άδεια σημαίνει πως τα tags δεν είναι σωστά ταιριασμένα και εμφανίζεται το μήνυμα "tags are not matching".

Στον φάκελο src μαζί με τα αρχεία java, δημιουργήσαμε και ένα αρχείο html (tagMatching.html) στο οποίο κάναμε αλλαγές ώστε να δούμε αν το πρόγραμμά μας τρέχει.

Από command line, το πρόγραμμα τρέχει με τις εξής δύο εντολές:
>javac TagMatching.java
>java TagMatching tagMatching.html

C. NetBenefit

Για το παρόν πρόγραμμα-πελάτη χρησιμοποιήσαμε την υλοποίηση μεθόδων των ουρών FIFO του πρώτου μέρους της εργασίας. Σε πρώτο στάδιο, διαβάζουμε το αρχείο με τη βοήθεια έτοιμων κλάσεων της Java (FileReader, BufferedReader) και πετάμε IOException σε περίπτωση που δεν βρεθεί. Δημιουργήσαμε δύο βασικές ουρές, queueBuy και queuePrice στις οποίες αποθηκεύονται στη μεν ποσότητα των μετοχών και στη δε την τιμή.

Αρχικά, διαβάζουμε σειρά προς σειρά το txt αρχείο που έχει δοθεί από τον χρήστη και με τη βοήθεια κλάσεων της Java (regex.Pattern, regex.Matcher) απομονώνουμε τους αριθμούς και στη συνέχεια τους τοποθετούμε σε μία λίστα (list) μεγέθους δύο όπου τους μετατρέπει από string σε integer. Εάν η γραμμή είναι της μορφής «buy x price y» τότε τοποθετούμε τα δεδομένα στις αντίστοιχες ουρές μέσω της μεθόδου put. Αν πάλι η γραμμή είναι της μορφής «sell x price y» δημιουργούμε δύο μεταβλητές sell και price στις οποίες φορτώνουμε τις αντίστοιχες τιμές.

Σε αυτό το στάδιο, όσο οι μετοχές μας είναι θετικές (σε άλλη περίπτωση πετάμε NoSuchElementException και μήνυμα λάθους) έχουμε δύο περιπτώσεις, αν οι μετοχές που βρίσκονται στην αρχή της ουράς είναι μικρότερες/ίσες με την ποσότητα που θέλουμε να πουλήσουμε ή αν είναι μεγαλύτερες. Στην πρώτη περίπτωση, υπολογίζουμε το κέρδος ενώ παράλληλα αφαιρούμε τις με την μέθοδο get() την τιμή του συγκεκριμένου κόμβου καθώς έχουμε πουλήσει τις μετοχές που είχαν την συγκεκριμένη τιμή. Επίσης, αφαιρούμε από τη μεταβλητή sell τις μετοχές εκείνες που έχουμε πουλήσει και ήταν πρώτες στην ουρά πάλι με τη χρήση της μεθόδου get(). Στη δεύτερη περίπτωση, υπολογίζουμε πάλι το κέρδος ενώ παράλληλα αφαιρούμε την ποσότητα από το head στοιχείο των μετοχών και κάνουμε break όταν δεν έχουμε άλλες μετοχές (sell=0).

Αν διαβάσουμε και το αρχείο το οποίο ανοίξαμε, το κλείνουμε και εκτυπώνουμε το συνολικό Net Benefit.

Στον φάκελο src μαζί με τα αρχεία java, δημιουργήσαμε και ένα αρχείο txt (netBenefit.txt) στο οποίο κάναμε αλλαγές ώστε να δούμε αν το πρόγραμμά μας τρέχει.

Από command line, το πρόγραμμα τρέχει με τις εξής δύο εντολές:
>javac NetBenefit.java
>java NetBenefit netBenefit.txt