Αθανασίου Λυδία: p3170003

Θεμελίνα Κουζούμπαση: p3170076

ΜΕΡΟΣ Δ

ΜΕΡΟΣ Α:

Υλοποίηση πρώτης Διεπαφής (StringStack)-> StringStackImpl implements StringStack

Υλοποίηση του κατασκευαστή: public StringStackImpl() για τη δημιουργία νέας κενής στοίβας.

Υλοποίηση της μεθόδου: public boolean isEmpty() η οποία επιστρέφει true σε περίπτωση που η στοίβα είναι άδεια/δεν περιέχει κανένα στοιχείο.

Υλοποίηση της μεθόδου: public void push(String item) για να έχουμε την δυνατότητα να εισάγουμε αντικείμενα (τύπου String) στη στοίβα.

Υλοποίηση της μεθόδου: public String pop() για να μπορούμε να εξάγουμε στοιχείο από την λίστα σύμφωνα με την αρχή LIFO(Last In- First Out). Δηλαδή το τελευταίο στοιχείο που εισάχθηκε να διεξαχθεί (το οποίο και επιστρέφεται). Η συγκεκριμένη μέθοδος επιστρέφει exception σε περίπτωση που η στοίβα είναι άδεια και δεν υπάρχει στοιχείο ώστε να αφαιρεθεί.

Υλοποίηση της μεθόδου: public String peek() η οποία μας δίνει τη δυνατότητα να ελέγξουμε το τελευταίο στοιχείο της στοίβας που εισάχθηκε χωρίς να το αφαιρέσει (το οποίο και επιστρέφει). Η συγκεκριμένη μέθοδος επιστρέφει exception σε περίπτωση που η στοίβα είναι άδεια και δεν υπάρχει στοιχείο ώστε να επιστραφεί.

Υλοποίηση της μεθόδου: public void printStack(PrintStream stream) με την οποία εκτυπώνουμε τα στοιχεία της στοίβας.

Υλοποίηση της μεθόδου: public int size() ώστε να επιστρέφεται το μέγεθος της στοίβας διατηρώντας την πολυπλοκότητα της χαμηλή με την χρήση counter μεταβλητής η οποία αυξάνεται και μειώνεται κάθε φορά που εισαγάγεται ή εξάγεται αντιστοίχως στοιχείο από την στοίβα.

Υλοποίηση δεύτερης διεπαφής IntQueue-> IntQueueImpl implements IntQueue

Υλοποίηση του κατασκευαστή public IntQueueImpl() για τη δημιουργία νέας κενής ουράς.

Υλοποίηση της μεθόδου public void isEmpty() η οποία επιστρέφει true σε περίπτωση που η ουρά είναι άδεια/δεν περιέχει κανένα στοιχείο.

Υλοποίηση της μεθόδου: public void put(int item) για να έχουμε την δυνατότητα να προσθέτουμε αντικείμενα (τύπου int) στην ουρά.

Υλοποίηση της μεθόδου: public int get() για να μπορούμε να εξάγουμε στοιχείο από την ουρά σύμφωνα με την αρχή FIFO(First In – First Out). Δηλαδή το πρώτο στοιχείο που εισάχθηκε να διεξαχθεί (το οποίο και επιστρέφει). Η συγκεκριμένη μέθοδος επιστρέφει exception σε περίπτωση που η ουρά είναι άδεια και δεν υπάρχει στοιχείο ώστε να αφαιρεθεί.

Υλοποίηση της μεθόδου: public int peek() η οποία μας δίνει τη δυνατότητα να ελέγξουμε το πρώτο στοιχείο της ουράς που εισάχθηκε χωρίς να το αφαιρέσει (το οποίο και επιστρέφει). Η συγκεκριμένη μέθοδος επιστρέφει exception σε περίπτωση που η ουρά είναι άδεια και δεν υπάρχει στοιχείο ώστε να επιστραφεί.

Υλοποίηση της μεθόδου: public void printQueue(PrintStream stream) με την οποία εκτυπώνουμε τα στοιχεία της ουράς

Υλοποίηση της μεθόδου: public int size() ώστε να επιστρέφεται το μέγεθος της ουράς διατηρώντας την πολυπλοκότητα της χαμηλή με την χρήση counter μεταβλητής η οποία αυξάνεται και μειώνεται κάθε φορά που εισαγάγεται ή εξάγεται αντιστοίχως στοιχείο από την ουρά.

ΜΕΡΟΣ Β: Υλοποίηση προγράμματος πελάτη TagMatching

Στην αρχή του προγράμματος δημιουργούμε αναφορές τύπου File και BufferReader με τιμή null, τα οποία χρησιμοποιούμε αργότερα διαβάζοντας το αρχείο που μας δίνει ο χρήστης. Δημιουργούμε επίσης την στοίβα stack που θα χρησιμοποιήσουμε στο πρόγραμμα μας(StringStackImpl stack = new StringStackImpl()).

Με την χρήση try/catch βρίσκουμε τις exceptions σε περίπτωση που το πρόγραμμα αδυνατεί να βρει ή να ανοίξει το αρχείο αντιστοίχως. Και στις δύο περιπτώσεις αν βρεθεί exception τυπώνεται σχετικό μήνυμα.

Στο κύριο μέρος του προγράμματος διαβάζουμε το αρχείο με την χρήση BufferReader, γραμμή προς γραμμή και με τη χρήση ενός while loop. Χρησιμοποιώντας τις κλάσεις Pattern, Matcher(που έχω κάνει import) και τις μεθόδους αυτών, δημιουργώ patterns τα οποία διαβάζουν όλο το περιεχόμενο των tags (<tag> και </tag>) που μπορεί να είναι κεφαλαία, πεζά γράμματα και αριθμοί ("<[a-zA-Z0-9]+?(?=>)" και "</[a-zA-Z0-9]+?(?=>)" αντίστοιχα), πιάνοντας και το πρώτο “<” αλλά όχι το δεύτερο (πχ απ’το tag <html> πιάνουν το “<html”, αντίστοιχα απ’το </html> το “<html”).

Με τη χρήση εμφωλιασμένου στο πρώτο while loop, όταν το πρόγραμμα βρει tag της πρώτης μορφής (<tag>) το προσθέτει στη στοίβα.

Με τη χρήση δεύτερου εμφωλιασμένου στο πρώτο while loop όταν το πρόγραμμα βρει tag της δεύτερης μορφής (</tag>) ελέγχει αν το περιεχόμενο του (“tag”) είναι ίδιο με το τελευταίο στοιχείο που έχει εισαχθεί στη στοίβα (substring(matcher2.start() +2,matcher2.end()).equalsIgnoreCase(stack.peek())) και αν η συνθήκη είναι αληθής τότε το στοιχείο που εισάχθηκε τελευταίο στη στοίβα αφαιρείται (stack.pop()).

Το κύριο μέρος του προγράμματος είναι εμφωλιασμένο σε ένα try/catch σε περίπτωση που το πρόγραμμα δεν μπορεί να αναγνώσει γραμμή.

Το πρόγραμμα τερματίζει και αν το size της stack είναι ίσο με το μηδέν, πράγμα που σημαίνει ότι οι ετικέτες ταιριάζουν, εμφανίζει θετικό μήνυμα και αντίστοιχα αν το size είναι μεγαλύτερο του μηδενός (υπάρχουν ετικέτες στο κείμενο HTML που δεν έχουν κλείσει ή που έχουν κλείσει αλλά όχι με τη σωστή σειρά) εμφανίζει αντίστοιχο μήνυμα για να ενημερώσει τον χρήστη.

ΜΕΡΟΣ Γ: Υλοποίηση προγράμματος πελάτη NetBenefit

Στην αρχή του προγράμματος δημιουργούμε αναφορές τύπου File και BufferReader με τιμή null, τα οποία χρησιμοποιούμε αργότερα διαβάζοντας το αρχείο που μας δίνει ο χρήστης. Δημιουργούμε επίσης τις ουρές που θα χρησιμοποιήσουμε στο πρόγραμμα μας queue (για τις τιμές) και queue2 (για τις μετοχές) (StringStackImpl stack = new StringStackImpl()).

Με την χρήση try/catch βρίσκουμε τις exceptions σε περίπτωση που το πρόγραμμα αδυνατεί να βρει ή να ανοίξει το αρχείο αντιστοίχως. Και στις δύο περιπτώσεις αν βρεθεί exception τυπώνεται σχετικό μήνυμα.

Στο κύριο μέρος του προγράμματος διαβάζουμε το αρχείο με την χρήση BufferReader, γραμμή προς γραμμή και με τη χρήση ενός while loop. Ελέγχουμε με τη χρήση συνθήκης if τις δυο περιπτώσεις, αν δηλαδή έχουμε αγορά(buy) ή πώληση(sell).

Σε περίπτωση αγοράς αποθηκεύουμε όλες τις τιμές (μετοχές και ποσά) στην ουρά queue.

Σε περίπτωση πώλησης αρχικά διαχωρίζουμε τα στοιχεία της ουράς queue σε μετοχές και ποσά, μεταφέροντας τις μετοχές στην ουρά queue2. Άρα τώρα η queue έχει τα ποσά, ενώ η queue2 τις μετοχές.
Αποθηκεύουμε πόσες μετοχές πουλήθηκαν σε μία μεταβλητή (sell) και σε τι τιμή σε άλλη μεταβλητή (v). Στη συνέχεια υπολογίζουμε ξεχωριστά πόσες από τις υπάρχουσες μετοχές πουλήθηκαν ώστε να συμπληρωθεί η ποσότητα μετοχών που πωλήθηκε και αφαιρούμε σταδιακά τις μετοχές αυτές. Με ένα while loop υπολογίζουμε το κέρδος/ζημία σύμφωνα με τον τύπο που μας δόθηκε, αφαιρώντας κάθε φορά από τις ουρές τις μετοχές και τη τιμή τους αντίστοιχα.

Βγαίνοντας απ'την περίπτωση της αγοράς στην ουρά queue2 πλέον θα υπάρχουν μόνο οι μετοχές που δεν πωλήθηκαν και στην ουρά queue η τιμή που αγοράστηκαν.

Το κύριο μέρος του προγράμματος είναι εμφωλιασμένο σε ένα try/catch σε περίπτωση που το πρόγραμμα δεν μπορεί να αναγνώσει γραμμή.

Το πρόγραμμα τερματίζει και αν η μεταβλητή profit είναι θετική εμφανίζει θετικό μήνυμα και το ποσό του κέρδους και αντίστοιχα αν η μεταβλητή profit είναι μικρότερη του μηδενός εμφανίζει αντίστοιχο μήνυμα για να ενημερώσει τον χρήστη για τη ζημία.