ΠΡΩΤΗ ΕΡΓΑΣΙΑ ΣΤΙΣ ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

ΜΠΙΛΙΑΣ ΓΕΩΡΓΙΟΣ 3200278

ΜΠΟΥΡΑΖΑΝΑΣ ΠΑΝΑΓΙΩΤΗΣ 3200268

Για το μέρος Α της εργασίας αρχικά φτιάξαμε μια κλάση Node όπου μέσα της δέχεται δεδομένα (**data**), επιστρέφει δεδομένα (**getData**) καθώς και μπορεί να οδηγηθεί στα επόμενα (**getNext**) και να θέσει επόμενα δεδομένα με την χρήση της next (**setNext**).

Αξίζει να σημειωθεί πως η υλοποίηση αυτή έγινε με την χρήση Τ τύπου, generics, για την εύκολη επαναχρησιμοποίηση του κώδικα καθώς και πως θα χρησιμοποιηθεί στην υλοποίηση της στοίβας και της ουράς.

Για την ουρά φτιάξαμε μια **IntQueueImpl** κλάση η οποία κάνει implement την διεπαφή **IntQueue** που μας έχει δοθεί. Δηλαδή, σε αυτή υλοποιήσαμε τις μεθόδους που θέλουμε να χρησιμοποιηθούν στην ουρά. Αρχικά, θέσαμε δυο Nodes, την **front** και **back**, οι οποίες θα μας βοηθήσουν στην υλοποίηση της ουράς μας. Τώρα, για την πρώτη μέθοδο της ουράς που αποκαλούμε isEmpty απλά επιστρέφουμε το front ως null, κάτι που σημαίνει πως η ουρά είναι άδεια. Στην συνέχεια, φτιάξαμε μια μέθοδο **put** όπου σε αυτή θα εισάγουμε ένα node. Γι' αυτή την μέθοδο αρχικά πήραμε την περίπτωση οπου η ουρά είναι κενή και θέσαμε και την **front** αλλά και την **back** ως n, αλλιώς του είπαμε να πάει πίσω (**back**) και να εισάγει εκεί το n. Ακολούθως, φτιάξαμε μια μέθοδο get η οποία λόγω του **FIFO** την βάλαμε να παίρνει το node που μπήκε νωρίτερα απ' όλα, δηλαδή το front και να το εξάγει απ' την ουρά. Αναλυτικότερά, λέμε εάν η ουρά έχει **front** ίσο με **back** πάει να πει πως έχει ένα μόνο στοιχείο που ανήκει και στα δυο οπότε τα θέτουμε και τα δυο ως null αλλιώς του λέμε να πάει το **front** στο επόμενο Node. Αξίζει να αναφερθεί πως επίσης βάλαμε σε περίπτωση που η ουρά είναι κενή να πετάει το κατάλληλο error.

Στην συνέχεια, φτιάξαμε και μια μέθοδο **peek** η οποία απλά επιστρέφει το παλαιότερο δεδομένο χωρίς να το εξάγει από την ουρά. Γι' αυτή την μέθοδο απλά του είπαμε εάν η ουρά είναι κενή να πετάξει το κατάλληλο error αλλιώς να πάρει το δεδομένο απ' την **front** και να το επιστρέψει. Επίσης, φτιάξαμε μια μέθοδο όπου εκτυπώνει όλα τα στοιχεία της ουράς απ' το παλαιότερο στο νεότερο. Γι' αυτή την μέθοδο απλά θέσαμε ένα **iterator** το οποίο ξεκινάει από την **front** και κάναμε μια while loop όσο το **iterator** είναι διάφορο του κενού να printάρει τα δεδομένα του και να το θέτει κάθε φορά ως το επόμενο Node. Τέλος, για την μέθοδο **size** απλά θέσαμε ένα **counter** όπου κάθε φορά που μπαίνει κάτι στην ουρά (**put**) να αυξάνεται κατά 1 ενώ όταν γίνεται εξαγωγή (**get**) να μειώνεται κατά 1 αντιστοίχως. Έτσι, μετά πήγαμε στην **size** μέθοδο και είπαμε εάν η ουρά είναι κενή τότε το μέγεθος είναι 0 αλλιώς είναι ίσο με το **counter**.

Όσο αφορά την στοίβα κατά ανάλογο τρόπο φτιάξαμε την κλάση **StringStackImpl** η οποία κάνει αντιστοίχως implement την διεπαφή **StringStack**. Αρχικά, θέσαμε δυο Nodes **top** and **bottom** ως null. Στην συνέχεια με ανάλογο τρόπο με την ουρά έτσι και στη στοίβα φτιάξαμε μια μέθοδο **isEmpty** που επιστρέφει το bottom ως null, δηλαδή μια άδεια στοίβα. Στην συνέχεια,

υλοποιούμε την **push** όπου εισάγουμε ένα νέο Node στην στοίβα. Αυτό αν η στοίβα είναι κενή το κάνουμε θέτοντας και το **top** αλλά και το bottom ίσα με n αλλιώς πηγαίνοντας στο επόμενο Node το θέτουμε ως **top** και εισάγουμε το n σε αυτό. Μετά φτιάξαμε και μια **pop** όπου παίρνουμε το στοιχείο από την **top** και στην συνέχεια το εξάγουμε. Πιο αναλυτικά, αρχικά παίρνουμε την περίπτωση όπου η στοίβα είναι κενή. Εάν κάτι τέτοιο ισχύει πετάει το κατάλληλο error, αλλιώς παίρνουμε το στοιχείο και αν το **top** είναι ίσο με το bottom, δηλαδή υπάρχει μόνο ένα στοιχείο στην στοίβα, θέτουμε αυτά τα 2 ως null αλλιώς απλά μεταφέρουμε το **top** στο επόμενο Node.

Για την υλοποίηση της **peek** αλλά και της εκτύπωσης όλης της στοίβας λειτουργήσαμε ακριβώς με τον ίδιο τρόπο όπως με την ουρά απλώς αντί για το **front** της ουράς χρησιμοποιήσαμε το **top** της στοίβας. Τέλος, για την υλοποίηση της **size** λειτουργήσαμε πάλι με τον ακριβώς ίδιο τρόπο όπως και με την ουρά. Δηλαδή, αυξάνουμε κατά 1 τον counter στην **push** και μειώνουμε κατά 1 στην **pop** αντίστοιχα.

Το Β μέρος υλοποιήθηκε με την βοήθεια του Α μέρους της εργασίας και πιο συγκεκριμένα χρησιμοποιώντας την στοίβα. Πέρα από την στοίβα δημιουργήθηκαν επιπλέον δύο αρχεία/κλάσεις: η **TagMatching** και η **ReadHtml**. Στην **TagMatching** στην ουσία διαβάζουμε το αρχείο μέσω της **ReadHtml**, αφού πρώτα δημιουργήσουμε ένα καινούργιο αντικείμενό της. Επιπλέον από αυτήν εκτελείται το πρόγραμμα. Στην **ReadHtml**, αρχικά δημιουργείται μια στοίβα tag_stack και μια tag_stack_close. Μετά κάνουμε μια while loop όσο η line δεν είναι null. Μέσα σε αυτή την loop παίρνουμε αρχικά για κάθε line την trimάρουμε. Στην συνέχεια, παίρνουμε μια if όπου τσεκάρουμε εάν αυτή η τριμαρισμένη λέξη ξεκινάει από ‘’<’’ μπαίνει και ακολουθεί μια άλλη if όπου εάν δεν έχει closing tag την αποθηκεύει στην στοίβα των **tags** (**tag_stack**). Αλλιώς, εάν έχει **closing tag** το αποθηκεύει στην στοίβα των closing tags (**tag_stack_close**). Στην συνέχεια τσεκάρει αν το **top** της στοίβας ισοδυναμεί με αυτό το closing tag. Εάν κάτι τέτοιο ισχύει τότε αφαιρεί τα top από τις στοιβες των tags . Τέλος, εάν οι στοίβες είναι κενές σημαίνει πως η υλοποίηση του html αρχείου είναι σωστή και printάρει το κατάλληλο μήνυμα.

Το Γ μέρος υλοποιήθηκε με την βοήθεια του Α μέρους της εργασίας και πιο συγκεκριμένα με την χρήση της ουράς. Επίσης, όπως και στο Μέρος Β δημιουργήσαμε 2 κλάσεις: Την **NetBenefit** και την **ReadTxt**.

Η **NetBenefit** είναι μια κλάση που απλώς περιέχει την main, η οποία χρησιμοποιείται για να εισάγει το αρχείο txt που θέλουμε να διαβαστεί στην ReadTxt και να γίνει η διαδικασία που επιθυμούμε. Αυτό γίνεται δημιουργώντας ένα νέο αντικείμενο **ReadTxt** και στην συνέχεια καλώντας την μέθοδο **LoadFile** της **ReadTxt** και εισάγοντας το αρχείο txt εκεί.

Η **ReadTxt** είναι μια κλάση που απλώς περιέχει μέσα της μια μέθοδο **LoadFile** οπού εκεί γίνεται όλη η διαδικασία. Στην **LoadFile** ουσιαστικά δημιουργήσαμε 2 ουρές, μια για την ποσότητα των μέτοχων που αγοράζονται (**stock_queue**) και μια για τις τιμές όπου αγοράστηκαν αυτές οι μετοχές (**price_queue**). Αυτές οι ουρές γίνονται με την βοήθεια της **IntQueueImpl** απτό μέρος Α.

Στην συνέχεια, αφού έχουμε τσεκάρει ότι το αρχείο txt βρέθηκε (1η try) και ότι ανοίχθηκε επιτυχώς (2η try) αρχικά κάνουμε split σε διαφορετικά words τις λέξεις ανάμεσα στα κενά.

Αφού το κάνουμε αυτό παίρνουμε την πρώτη λέξη που είναι είτε **buy** είτε **sell** και παίρνουμε μια if για κάθε μια απ' τις 2 αυτές περιπτώσεις. Εάν αυτή η πρώτη word που έχουμε θέσει ως **type** είναι buy τότε παίρνουμε την πρώτη λέξη (**ποσότητα μέτοχων/stock**) και την βάζουμε στην **stock_queue** καθώς και την τρίτη λέξη (**τιμή μέτοχων/price**) και την βάζουμε μέσα στην **price_queue**. Επίσης, δημιουργούμε και μια μεταβλητή **total_stock** όπου κάθε φορά που γίνεται μια αγορά προστίθεται σ' αυτή η ποσότητα των μέτοχων που μόλις αγοράστηκαν ώστε να γνωρίζουμε την ποσότητα των μέτοχων που κατέχουμε την ώρα που κάνουμε **sell**. Αλλιώς εάν το type είναι **sell** τότε γίνεται η διαδικασία της πώλησης μέτοχων.

Εάν απτή άλλη η πρώτη λέξη στο txt αρχείο είναι sell τότε αρχικά δημιουργήσαμε μια loop όσο το stock που θέλουμε να πουλήσουμε δεν έχει πουληθεί (**sell_stock != 0**). Στην συνέχεια παίρνουμε ως παράμετρο εάν δεν υπάρχει προηγούμενο stock, δηλαδή αν δεν έχει περισσέψει stock από την get μέθοδο (εάν είχε γίνει). Έτσι, εάν δεν υπάρχει υπολειπόμενο stock τότε παίρνουμε και θέτουμε ως τωρινό stock με την χρήση της get() (**current_stock**) και ως τωρινή price με τον ανάλογο τρόπο με την χρήση της get μεθόδου της άλλης ουράς (**current_price**).

Μετά εκεί μέσα παίρνουμε άλλη παράμετρο όπου λεμέ, εάν η τωρινή τιμή που πήραμε από την ουρά είναι μεγαλύτερη από αυτή που θέλουμε να πουλήσουμε και κάνουμε τις ανάλογες πράξεις. Δηλαδή, κάνουμε την πράξη για να βρούμε το κέρδος της συγκεκριμένης sell διαδικασίας, αφαιρούμε απτό συνολικό stock που έχουμε κρατήσει (**total stock**) αυτό που μόλις πουλήθηκε, κρατάμε το περισσευούμενο stock και τιμή σε δυο μεταβλητές (**remaining_stock και remaining_price**) και θέτουμε το stock που θέλουμε να πουλήσουμε ίσο με το 0, αφού έγινε η διαδικασία πώλησης όλου του stock που θέλαμε να πουλήσουμε. Αλλιώς, αν το stock που έχουμε (**current_stock**) είναι μικρότερο ή ίσο από αυτό που θέλουμε να πουλήσουμε (**sell_stock**) υπολογίζουμε το κέρδος ή ζημιά της πώλησης, απλά αφαιρούμε απτό συνολικό stock (**total_stock**) αυτό που μπορούμε να πουλήσουμε και στην προκείμενη περίπτωση επειδή το stock που πήραμε από την ουρά είναι μικρότερο ή ίσο με αυτό που θέλουμε να πουλήσουμε, το αφαιρούμε κατά την τιμή αυτού που πήραμε από την ουρά (**current_stock**). Επειδή όμως όπως προαναφέραμε είναι μικρότερο ή ίσο αυτού που θέλουμε να πουλήσουμε η διαδικασία πώλησης ίσως να μην έχει τελειώσει. Γι' αυτό το λόγο κάνουμε μια διαδικασία αφαίρεσης απτό ποσό stocks που θέλουμε να πουλήσουμε κατά αυτό που ήδη πουλήθηκε ώστε αν αυτό που πουλήθηκε είναι μικρότερο αυτού που θέλαμε να πουλήσουμε να συνεχιστεί η διαδικασία πώλησης.

Τώρα για την περίπτωση όπου υπάρχει περισσευούμενο stock (**remaining_stock**) κάνουμε πάλι με παρόμοιο τρόπο την διαδικασία πώλησης με μικρές διάφορες. Δηλαδή, για την πράξη του κέρδους ή ζημίας αντί να κάνουμε χρήση της τιμής και του stock που παίρνουμε απ' την ουρά κάνουμε την χρήση των μεταβλητών περισσευούμενου stock και την τιμή του (**remaining_stock** και **remaining_price**). Επίσης, στην περίπτωση που το ποσό που έχουμε να πουλήσουμε είναι μεγαλύτερο ή ίσο αυτού που έχει περισσέψει, αντί να κάνουμε αφαίρεση πάλι με το stock που παίρνουμε από την ουρά, την κάνουμε με την χρήση του περισσευούμενου stock (**remaining_stock**). Αντίστοιχα, στην περίπτωση που το ποσό μέτοχων που θέλουμε να πουλήσουμε είναι μικρότερο αυτού που έχει περισσέψει τότε αφαιρούμε το περισσευούμενο stock (**remaining_stock**) κατά το ποσό που πουλήθηκε (**sell_stock**).

Τέλος, θέσαμε μια μεταβλητή **final_result** όπου προστίθεται το αποτέλεσμα της κάθε διαδικασίας sell και στο τέλος printάρουμε αυτό το κέρδος ή ζημιά με το κατάλληλο μήνυμα.

Για το τρέξιμο του κώδικα στο IDE του Eclipse πρέπει να αλλάξετε το path στο NetBenefit.java και στο TagMatching.java αντίστοιχα. Αυτό επιτυγχάνεται πατώντας δεξί κλικ στο αντίστοιχο file > Run As > Run Configurations… > Arguments > και βάζετε στο Program Arguments το path που βρίσκεται το αρχείο txt ή html αντίστοιχα. Πχ. Εάν θέλουμε να τρέξουμε το NetBenefit.java στην προκείμενη περίπτωση θα πατήσουμε δεξί κλικ πάνω του, θα κάνουμε την παραπάνω διαδικασία και θα βάλουμε ως path το C:\Users\georg\OneDrive\Desktop\3200278_3200268\src\NetBenefit.txt

Για το τρέξιμο του cmd θα πρέπει πρώτα να κάνετε το γνωστό cd και compile και στην συνέχεια θα κάνετε ακριβώς αυτό που δίνετε και στην εκφώνηση με το path σας. Πχ για το NetBenefit θα κάνουμε

java NetBenefit C:\Users\georg\OneDrive\Desktop\3200278_3200268\src\NetBenefit.txt

Εμάς ο κώδικας τρέχει κανονικά και όλα φαίνονται να λειτουργούν ομαλά. Για οποιοδήποτε πρόβλημα αν είναι δυνατόν επικοινωνήστε μαζί μας.

Τα email μας είναι: p3200278@aueb.gr, p3200268@aueb.gr