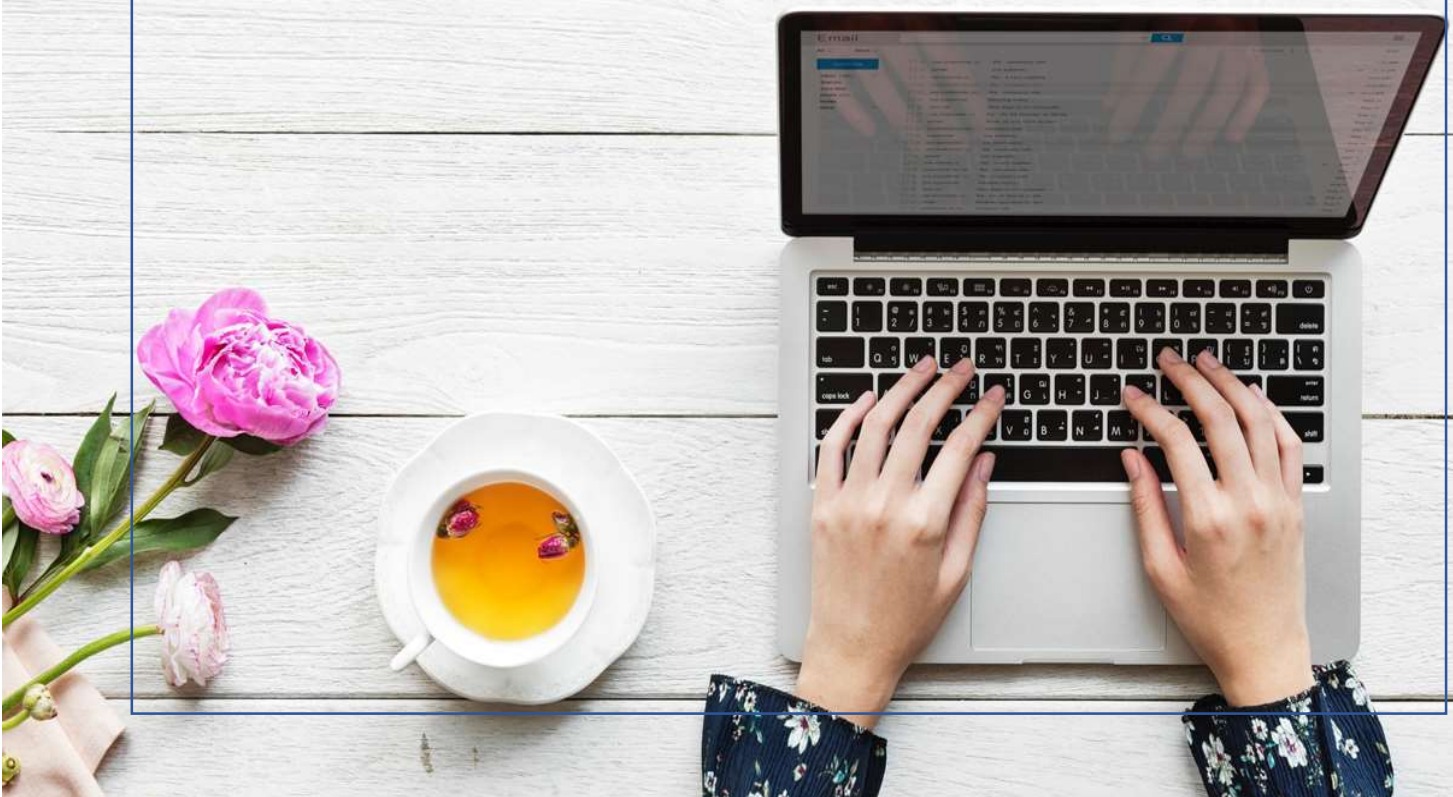


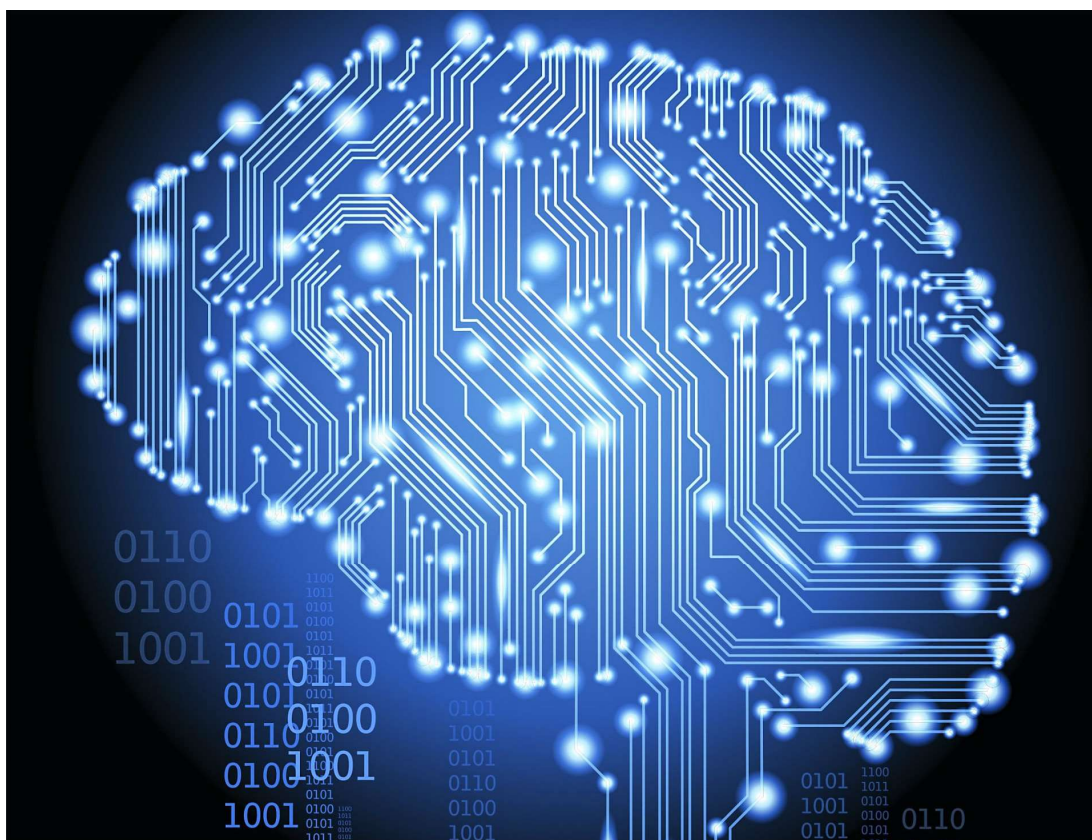
Πέτσα Γεωργία 3200155  
Παναγιώτης Τριανταφυλλίδης 3200199

Πρώτη εργασία στις Δομές Δεδομένων 2021-2022



## ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

Μέρος Α.....	3
Μέρος Β.....	5
Μέρος Γ.....	5
Σχόλια.....	6





## ΜΕΡΟΣ Α

Για την υλοποίηση της ουράς **FIFO** (IntQueueImpl) δουλέψαμε ως εξής:

- ✓ Αποφασίσαμε να βάλουμε έναν counter έτσι ώστε κάθε φορά που εισάγεται κάποιο στοιχείο στην ουρά, αυτός να αυξάνεται και, αντίστοιχα κάθε φορά που διαγράφεται ένα στοιχείο από αυτήν να μειώνεται. Έτσι στην υλοποίηση της `size()` δεν χρειάστηκε να διατρέξουμε επαναληπτικά την ουρά ούτε να χρησιμοποιήσουμε τον έλεγχο της `isEmpty()` με αποτέλεσμα να πετύχουμε χρόνο  $O(1)$ .
- ✓ Θεωρώντας την αρχή της ουράς ως `head` και το τέλος της ως `tail`, στην `put(T item)` κάναμε τους απαραίτητους ελέγχους και αναλόγως εισαγάγαμε το καινούριο στοιχείο στο τέλος της ουράς.
  - Πιο συγκεκριμένα, αν η ουρά είναι άδεια σημαίνει ότι το πρώτο της στοιχείο θα είναι και η αρχή της και το τέλος της. Σε αντίθετη περίπτωση πρώτα κάναμε την αναφορά του τελευταίου μέχρι πρότινος στοιχείου να δείχνει σε αυτό που θέλουμε να προσθέσουμε και μετά αλλάζουμε το `tail` έτσι ώστε να δείχνει στο τωρινό τελευταίο στοιχείο.


Αντίστοιχα στην `get()` ελέγχουμε αν είναι κενή η ουρά δημιουργώντας `exception`, αλλιώς αφαιρούμε την κεφαλή (`head`) και επιστρέφουμε τα δεδομένα αυτής.

- Πιο συγκεκριμένα, πάλι ελέγχουμε αν η ουρά είναι άδεια και δημιουργούμε `exception`. Σε αντίθετη περίπτωση βάζουμε τα δεδομένα του στοιχείου προς διαγραφή σε μία μεταβλητή `data` για να τα επιστρέψουμε. Αν έχει ένα στοιχείο τότε αυτό πρέπει να διαγραφεί επομένως κάνουμε τα `head` και `tail` να είναι `null`. Αν έχει παραπάνω από ένα στοιχεία τότε κάνουμε τη κεφαλή (`head`) να δείχνει στο επόμενο στοιχείο της ουράς, διαγράφοντας έτσι το στοιχείο που θέλουμε και επιστρέφοντας τα δεδομένα αυτού.


Για την υλοποίηση της στοίβας **LIFO** (StringStackImpl) δουλέψαμε ως εξής:

- ✓ Αποφασίσαμε να βάλουμε έναν counter έτσι ώστε κάθε φορά που εισάγεται κάποιο στοιχείο στην στοίβα, αυτός να αυξάνεται και, αντίστοιχα κάθε φορά που διαγράφεται ένα στοιχείο από αυτήν να μειώνεται. Έτσι στην υλοποίηση της `size()` δεν χρειάστηκε να διατρέξουμε επαναληπτικά την στοίβα ούτε να χρησιμοποιήσουμε τον έλεγχο της `isEmpty()` με αποτέλεσμα να πετύχουμε χρόνο  $O(1)$ .
- ✓ Θεωρώντας τον «πάτο» της στοίβας ως `tail` και την κορυφή της ως `head` (έτσι ώστε να αφαιρούμε κάθε φορά το `head`), στην `push(T item)` κάναμε τους απαραίτητους ελέγχους και αναλόγως εισαγάγαμε το καινούριο στοιχείο στο τέλος της στοίβας.



 Πιο συγκεκριμένα, αν η στοίβα είναι άδεια σημαίνει ότι το νέο της στοιχείο θα είναι και η αρχή της και το τέλος της. Σε αντίθετη περίπτωση πρώτα κάναμε την αναφορά του προς εισαγωγή στοιχείου να δείχνει στο head (δηλαδή το αμέσως προηγούμενο) και μετά το head να γίνεται ίσο με το νέο τελευταίο στοιχείο.

Αντίστοιχα στην pop() ελέγχουμε αν είναι κενή η στοίβα δημιουργώντας exception, αλλιώς αφαιρούμε την κεφαλή (head) και επιστρέφουμε τα δεδομένα αυτής.

 Πιο συγκεκριμένα, πάλι ελέγχουμε αν η στοίβα είναι άδεια και δημιουργούμε exception. Σε αντίθετη περίπτωση βάζουμε τα δεδομένα του head προς διαγραφή σε μία μεταβλητή data για να τα επιστρέψουμε. Αν έχει ένα στοιχείο τότε αυτό πρέπει να διαγραφεί επομένως κάνουμε τα head και tail να είναι null. Αν έχει παραπάνω από ένα στοιχεία τότε απλώς θέτουμε το head να είναι το αμέσως επόμενο στοιχείου του (head = head.getNext();). Οι κοινές υλοποιήσεις τόσο στην ουρά όσο και στην στοίβα είναι ότι χρησιμοποιήσαμε generics και ίδιο κώδικα για την printStack και printQueue. Στις 2 τελευταίες μεθόδους τυπώνουμε τα στοιχεία ένα ένα όλης της δομής με τη μέθοδο println() του αντικειμένου stream.

## ΜΕΡΟΣ Β

Ο τρόπος που επιλέξαμε να δουλέψουμε για το μέρος Β είναι ο εξής:

Βάλαμε μία μεταβλητή Boolean `ok = true`; Ούτως ώστε αν εντοπίσουμε κάποιο λάθος tag μέσα στο αρχείο, αυτή να γίνει false και να σταματήσει το διάβασμα του.

Αρχικά αποφασίσαμε να σπάμε όλη τη γραμμή που διαβάζουμε γράμμα-γράμμα σε έναν πίνακα `chars`. Έπειτα βλέπουμε αν κάποιο από τα `chars` είναι tag ανοίγματος (`<...>`) ή κλεισίματος (`></...>`) και αναλόγως πράττουμε:

- Αν είναι ανοίγματος, τότε απλά κάνει push το tag στην στοίβα `stackTags` χωρίς τα εισαγωγικά (Δηλαδή εάν έχουμε το tag `<h1>` στη στοίβα θα γίνει pushed το `h1`)
- Αν είναι κλεισίματος, τότε ελέγχουμε αν η στοίβα είναι άδεια και σε αυτή την περίπτωση βγάζουμε σφάλμα, αλλιώς ουσιαστικά διαγράφουμε το `"/` από το tag ώστε να δούμε αν είναι ίδιο με το τελευταίο tag που έχει μπει στη στοίβα. Αν αυτό ισχύει, τότε καλούμε τη μέθοδο `pop` για να αφαιρεθεί το τελευταίο στοιχείο και συνεχίζουμε την επανάληψη, αλλιώς βγάζουμε σφάλμα. (Αφού το τελευταίο tag δεν θα είναι ίδιο με το tag κλεισίματος άρα υπάρχει λάθος «ιεραρχία»).

## ΜΕΡΟΣ Γ

Ο τρόπος που αποφασίσαμε να δουλέψουμε σ' αυτό το σκέλος διαφέρει λίγο από τον προηγούμενο. Αρχικοποιούμε μία μεταβλητή `benefit = 0` η οποία θα μας δώσει στο τέλος το κέρδος ή την ζημία. Έπειτα «σπάμε» την κάθε γραμμή σε tokens (δηλαδή ανά λέξη) και εφόσον ξέρουμε ότι δεν θα υπάρχουν συντακτικά ή άλλα λάθη ελέγχουμε αν το token που έχουμε εκείνη τη στιγμή ισούται με τη λέξη `buy` ή `sell`.

- Στην πρώτη περίπτωση πηγαίνουμε στο επόμενο token για να πάρουμε τον αριθμό των μετοχών προς αγορά, και τον περνάμε σε μία μεταβλητή `sum`. Έπειτα «προσπερνάμε» τη λέξη `price` έτσι ώστε να έχουμε την τιμή της αγοράς, την μετατρέπουμε σε `integer` και τέλος βάζουμε στην ουρά την τιμή που έχουμε πάρει τόσες φορές όσες είναι και οι μετοχές.
- Στην δεύτερη, παίρνουμε το πλήθος των μετοχών στην ουρά και το ελέγχουμε με το πλήθος των επιθυμητών πωλήσεων. Αν οι πωλήσεις ξεπερνάνε τις ήδη υπάρχουσες μετοχές, τότε πρόκειται για σφάλμα. Αλλιώς πάλι «προσπερνάμε» τη λέξη `price` έτσι ώστε να πάρουμε την τιμή της πώλησης σε μία μεταβλητή `price`. Έπειτα βγάζουμε από την ουρά την τιμή της παλαιότερης μετοχής και την αφαιρούμε από την τιμή πώλησης μέχρι να φτάσουμε τον αριθμό των προς πώληση μετοχών. Το άθροισμα των παραπάνω αφαιρέσεων θα μας δώσει το `benefit`.

Στο τέλος του προγράμματος και εφόσον δεν έχει προκύψει κάποιο σφάλμα εμφανίζουμε κατάλληλο μήνυμα για κέρδος ή ζημία στον χρήστη.

**Υποσημείωση:** Για την σωστή ανάγνωση του αρχείου παρακαλούμε να μην υπάρχουν κενές γραμμές στο αρχείο διότι θα δημιουργηθεί `NoSuchElementException` και δεν θα λειτουργήσει σωστά ο κώδικας.



## ΣΧΟΛΙΑ

1. Στην εκτέλεση του κώδικα τόσο στο Μέρος Β όσο και στο Μέρος Γ χρησιμοποιήσαμε `command line arguments`.
2. Χρησιμοποιήσαμε στο Μέρος Α `generics`
3. Για δική μας επαλήθευση έχουμε χρησιμοποιήσει στο Μέρος Β κάποια `html` αρχεία καθώς και στο Μέρος Γ `txt` αρχεία, τα οποία έχουν την ίδια δομή με τα αντίστοιχα της εκφώνησης.