Μέρος Δ - Αναφορά

Τανέρ Ιμάμ p3200057

Ιπποκράτης Παντελίδης p3210150

1) Μέρος Α και Γ

Στο Α μέρος φτιάξαμε τις κλάσεις StringStackImpl και StringQueueImpl χρησιμοποιώντας λίστα μονής σύνδεσης με generics.Υλοποιούν τις μεθόδους των interfaces StringStack και StringQueue αντίστοιχα.

Στην StringStackImpl εφαρμόσαμε την LIFO (Last In First Out)

- Φτιάξαμε μια κλάση myNode η οποία έχει μια μεταβλητή value που είναι η τιμή που έχει ο κόμβος και ένα δείκτη nextNode,ο οποίος δηλώνει το που δείχνει ο κόμβος
- Μετά φτιάχνουμε τον Constructor της myNode όπου αρχικοποιούμε το value με την τιμή που δίνεται(item) και τον δείκτη nextNode να δείχνει στο null.
- Ορίζουμε μια μεταβλητή size η οποία θα μετράει το μέγεθος της στοίβας και ένα δείκτη top που θα δείχνει πάντα στην κορυφή της στοίβας.
- Στην συνέχεια φτιάχνουμε τον Constructor της StringStackImpl όπου ορίζουμε τον δείκτη top ίσο null.
- Μέθοδος isEmpty(): Επιστρέφει true αν η στοίβα είναι άδεια,δηλαδη αν το top δείχνει στο null,επομένως δεν υπάρχει κανένα στοιχείο στην στοίβα.Σε αντίθετη περίπτωση επιστρέφει false.
- Μέθοδος push(item):Εισάγουμε ένα στοιχείο στην κορυφή της λίστας. Φτιάχνουμε έναν νέο κόμβο με value ίσο με το όρισμα item και κάνουμε το δείκτη nextNode του νέου κόμβου να δείχνει εκεί που δείχνει το top.Και τέλος κάνουμε το top να δείχνει στον νέο κόμβο και αυξάνουμε το size κατά 1.
- Μέθοδος pop():Ελέγχουμε αν η στοίβα είναι άδεια. Αν είναι άδεια τότε έχουμε NoSuchElementException.Αν δεν άδεια τότε αποθηκεύουμε σε μια μεταβλητή την τιμή του κόμβου που δείχνει το top και μετακινούμε το top στον επόμενο κόμβο.Τέλος επιστρέψαμε την παραπάνω μεταβλητή και μειώσαμε το size κατά 1.
- Μέθοδος peek(): Ελέγχουμε ξανά αν η στοίβα είναι άδεια. Αν είναι έχουμε NoSuchElementException.Αλλιώς επιστρέφουμε το value του κόμβου που δείχνει ο top,δηλαδή το τελευταίο στοιχείο της στοίβας.
- Μέθοδος print Stack():Χρησιμοποιώντας ένα while loop εκτυπώνουμε τα στοιχεία της στοίβας ξεκινώντας από το top.Δηλαδή όσο υπάρχουν στοιχεία στην λίστα(top != null) εκτυπώνουμε την τιμή που δείχνει το top και το μετακινούμε στον επόμενο κόμβο.

- Μέθοδος size(): Αν η στοίβα είναι άδεια επιστρέφει 0 αλλιώς επιστρέφει το μέγεθος της στοίβας το οποίο έχουμε υπολογίσει στην size.

Στην StringQueueImpl εφαρμόσαμε την FIFO(First In First Out)

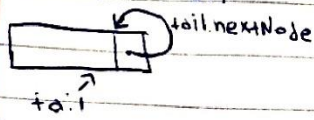
- Με τον ίδιο τρόπο φτιάχνουμε την κλάση Node και τον Constructor της, έχουμε την size,όμως έχουμε δυο δείκτες, τον head που δείχνει πάντα στην αρχή της ουράς και το tail που δείχνει στο τέλος.
- Στην συνέχεια φτιάχνουμε τον Constructor της StringQueueImpl όπου ορίζουμε το head και το tail ίσο με null.
- Μέθοδος isEmpty(): Επιστρέφει true αν η ουρά είναι άδεια, δηλαδή αν το head δείχνει στο null,επομένως δεν υπάρχει κανένα στοιχείο στην ουρά. Σε αντίθετη περίπτωση επιστρέφει false.
- Μέθοδος Put(item): Εισάγουμε ένα στοιχείο στο τέλος της λίστας. Φτιάχνουμε έναν νέο κόμβο με value ίσο με το όρισμα item και ελέγχουμε αν η ουρά είναι άδεια. Αν είναι, κάνουμε τους δείκτες head και tail να δείχνουν στον νέο κόμβο(έχουμε ένα στοιχείο στην ουρά).Αλλιώς κάνουμε τον δείκτη nextNode του tail να δείχνει στον νέο κόμβο και το tail στον ίδιο. Τέλος αυξάνουμε το size κατά 1.
- Μέθοδος get():Παίρνουμε ένα στοιχείο από την αρχή της ουράς. Ελέγχουμε αν η ουρά είναι άδεια. Αν είναι, τότε έχουμε NoSuchElementException.Αν όχι, τότε αποθηκεύουμε σε μια μεταβλητή την τιμή του κόμβου που δείχνει το head(πρώτο στοιχείο) και μετακινούμε το head στον επόμενο κόμβο. Επιπλέον αν το head δείχνει μετά την μετακίνηση στο null, η ουρά άδειασε οπότε κάνουμε και το tail να δείχνει στο null.Τέλος επιστρέψαμε την παραπάνω μεταβλητή και μειώσαμε το size κατά 1.
- Μέθοδος peek(): Ελέγχουμε ξανά αν η στοίβα είναι άδεια. Αν είναι έχουμε NoSuchElementException.Αλλιώς επιστρέφουμε το value του κόμβου που δείχνει ο head,δηλαδή το παλαιότερο στοιχείο της ουράς.
- Μέθοδος printQueue(): Χρησιμοποιώντας ένα while loop εκτυπώνουμε τα στοιχεία της ουράς ξεκινώντας από την αρχή. Δηλαδή όσο υπάρχουν περισσότερα από ένα στοιχεία στην λίστα εκτυπώνουμε την τιμή που δείχνει το head και το μετακινούμε στον επόμενο κόμβο. Τέλος όταν μείνει ένα στοιχείο στην ουρά το εκτυπώνουμε.
- Μέθοδος size(): Αν η ουρά είναι άδεια επιστρέφει 0 αλλιώς επιστρέφει το μέγεθος της το οποίο έχουμε υπολογίσει στην size.

Στο Γ μέρος φτιάξαμε το αρχείο StringQueueWithOnePointer χρησιμοποιώντας λίστα κυκλικής σύνδεσης με generics.Ειδικότερα παραλείψαμε τον δείκτη head και χρησιμοποιήσαμε μόνο τον δείκτη tail της ουράς. Υλοποιεί τις μεθόδους του interface StringQueue.

Με τον ίδιο τρόπο έχουμε την κλάση myNode

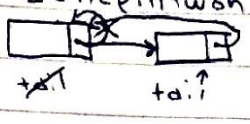
- Μέθοδος isEmpty(): Επιστρέφει true αν η στοίβα είναι άδεια, δηλ. αν το tail δείχνει στο null, αλλιώς επιστρέφει false
- Μέθοδος put(item): φτιάχνουμε νέο κόμβο όπως τις άλλες. Ελέγχουμε αν είναι άδεια, αν έχει 1 ή παραπάνω απο 1 στοιχεία

1η περίπτωση



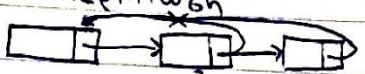
(Κάναμε το tail και τον επόμενο να δείχνουν στο node)

2η περίπτωση



Κάνουμε το επόμενο του tail να δείχνει στον νέο κόμβο και επόμενο του κόμβου στην δεχή Μετακινούμε το tail στο τέλος

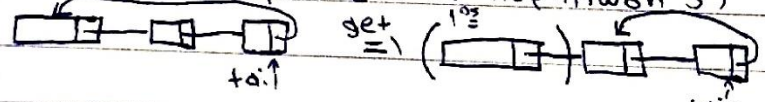
3η περίπτωση



Κάνουμε το επόμενο του νέου κόμβου να δείχνει στην δεχή (κυκλική) (1 μετά το τέλος) και το επόμενο του tail το node. Μετακινούμε το tail

- Μέθοδος get(): Αν είναι άδεια έχουμε Exception. Αν έχει ένα στοιχείο δηλ. (το tail.next και tail.next.next είναι ίδα) τότε κάνει το tail και το tail.next Node ίδα με null (0 στοιχεία)

Αλλιώς (π.χ είμαστε στην περίπτωση 3)



και έτσι επιτυγχάνεται η λογική διαγραφή του πρώτου κόμβου

- Μέθοδος peek(): Το tail βρίσκεται πάντα στον τελευταίο κόμβο, επομένως για να εμφανίσουμε το πρώτο στοιχείο παίρνουμε το value του επόμενου κόμβου του tail
- Μέθοδος printQueue: Όσο υπάρχουν παράπανω από 1 στοιχεία εμφανίζουμε κάθε φορά το value του tail.next Node δηλαδή την δεχή και μετά μετακινούμε τον επόμενο κόμβο. Στο τέλος όταν μείνει 1 στοιχείο το tail, το εκτυπώνουμε
- Μέθοδος size(): Ίδια με τις προηγούμενες

2) Μέρος Β

Το Β μέρος της εργασίας κάνει αναζήτηση σε ένα λαβύρινθο με σκοπό να βρει την έξοδο. Εμείς φτιάξαμε μια κλάση Thiseas η οποία περιέχει μια μέθοδο που αφαιρεί τα κενά από ένα string,μια μέθοδο Thiseas που διαβάζει το αρχείο που δίνει ο χρήστης και εξάγει ορισμένες πληροφορίες, μια βοηθητική κλάση Position,μια μέθοδο Solver και την main μας. Αρχικά η μέθοδος Thiseas παίρνει σαν όρισμα ένα αρχείο και εμείς σύμφωνα με αυτό φτιάχνουμε δυο πίνακες χαρακτήρων, έναν μονοδιάστατο 4 θέσεων που στις πρώτες δυο θέσεις έχει τον αριθμό των γραμμών και στηλών αντίστοιχα, στην τρίτη την γραμμή που βρίσκεται το Ε και στην τέταρτη την στήλη, και ένα δισδιάστατο που περιέχει το λαβύρινθο. Κάνουμε μερικούς ελέγχους για να δούμε αν τα δεδομένα είναι σωστά και ξεκινάμε την υλοποίηση. Χρησιμοποιήσαμε μια στοίβα χρησιμοποιώντας generics όπου περιέχει στοιχεία της κλάσης Position,δηλαδή μιας βοηθητικής κλάσης που μας βοηθάει να διαχειριστούμε τις συντεταγμένες του λαβυρίνθου. Ειδικότερα κάθε φορά που εφόσον ικανοποιούνται οι απαραίτητες συνθήκες υπάρχει γειτονικό κελί(δεξιά, αριστερά, κάτω, πάνω αλλά όχι διαγώνια),το κάνουμε push στην στοίβα. Σε περίπτωση που δε μπορούμε να μετακινηθούμε σε επόμενο κελί τότε κάνουμε pop() το τελευταίο Position, και με την βοήθεια ενος πίνακα tag που έχει σαν δείκτη το μέγεθος της στοίβας αποθηκεύει τις κινήσεις της(πχ δεξιά ,κάτω πάνω, αριστερά)και επανέρχεται στο σημείο που ήταν και πριν. Μετα ξαναβλέπω μήπως υπήρχε κάποια άλλη επιλογή μετακίνησης πέρα από την προηγουμένη. Αν πάλι δεν βρούμε κάνουμε επαναληπτικά pop() μέχρι να βρούμε. Όταν ικανοποιούνται οι έλεγχοι εξόδου τότε έχουμε την έξοδο, ενώ αν όχι τότε η στοίβα μας έχει αδειάσει(stack.isEmpty()) και δεν υπάρχει έξοδος. Ουσιαστικά η στοίβα στην υλοποίηση μας μας δείχνει το μονοπάτι που ακολουθεί το πρόγραμμα μας μέχρι να βρει η όχι την έξοδο, και μας βοηθάει μέσω του μεγέθους της να μπορούμε να κρατάμε τις προηγούμενες κινήσεις υλοποιώντας έτσι το backtracking.