ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ – ΕΡΓΑΣΙΑ 1

Βελισσαρίδης Γεώργιος (P3210255)

Μέρος Α & Γ

Γενικά

Για την υλοποίηση των τριών δομών δεδομένων χρησιμοποιήθηκε το generic type <Item>.

Επίσης χρησιμοποιήθηκε η εσωτερική κλάση Node, με πεδία ένα Item item, ένα Node next και έναν κατασκευαστή που αρχικοποιεί το item με το όρισμα της και το next σε null.

StringStackImlp

Δηλώθηκε ένας δείκτης σε Node με όνομα head που δείχνει στο στοιχείο της στοίβας που προστέθηκε πιο πρόσφατα και αρχικοποιήθηκε σε null. Δηλώθηκε ένας int με όνομα size, που μετρά πόσα στοιχεία έχει ανά πάσα στιγμή η στοίβα και αρχικοποιήθηκε με 0.

Η μέθοδος isEmpty() επιστρέφει true μόνο αν το head ισούται με null, δηλαδή όταν δεν υπάρχει κανένα στοιχείο στη στοίβα.

Η μέθοδος push(Item) προσθέτει ένα νέο Item στην κορυφή της στοίβας. Δημιουργείται ένα νέο Node oldNode που δείχνει στο head, ενώ στο head ανατίθεται νέο Node που περιέχει το Item που περάστηκε ως όρισμα. Το size αυξάνεται κατά ένα.

Η μέθοδος pop() αφαιρεί το Item που προστέθηκε τελευταίο στη στοίβα και το επιστρέφει. Αναθέτει το δείκτη head στο head.next, ώστε να αφαιρεθεί από τη στοίβα το τελευταίο στοιχείο και μειώνει το size κατά ένα. Αν δεν υπάρχει κανένα Item στη στοίβα κατά την κλήση της εγείρει NoSuchElementException.

Η μέθοδος peek() επιστρέφει το τελευταίο Item που προστέθηκε στη στοίβα χωρίς να το αφαιρεί από αυτή.

Η printStack() δημιουργεί το Node traverser που διασχίζει τη στοίβα ξεκινώντας από το head και εκτυπώνει κάθε Item.

StringQueueImpl

Δηλώθηκαν δύο δείκτες σε Node με όνομα head και tail που δείχνουν στο στοιχείο της ουράς που προστέθηκε λιγότερο και περισσότερα πρόσφατα αντίστοιχα, και αρχικοποιήθηκαν σε null. Δηλώθηκε ένας int με όνομα size, που μετρά πόσα στοιχεία έχει ανά πάσα στιγμή η ουρά και αρχικοποιήθηκε με 0.

Η μέθοδος put(Item) προσθέτει ένα νέο Item στο τέλος της ουράς. Στην περίπτωση που η ουρά δεν έχει κανένα στοιχείο, αρχικοποιεί νέο Node με το Item που περάστηκε ως όρισμα και το αναθέτει στο tail, το οποίο στη συνέχεια αναθέτει στο head. Στην περίπτωση που η ουρά περιέχει ήδη στοιχεία, δημιουργείται ο δείκτης oldTail που δείχνει στο tail, ενώ στο tail ανατίθεται νέο Node με το Item που περάστηκε ως όρισμα και το ίδιο το tail ανατίθεται στο oldTail.next. Το size αυξάνεται κατά ένα.

Η μέθοδος get() αφαιρεί το Item που προστέθηκε πρώτο στην ουρά και το επιστρέφει. Αναθέτει το head.next στο head, ώστε να αφαιρεθεί το Node με το πρώτο Item από την ουρά. Επίσης, αν μετά την αφαίρεση του στοιχείου η ουρά είναι άδεια, κάνει και το tail=null ώστε να μην διατηρείται άσκοπα αναφορά στο Node που αφαιρέθηκε. Μειώνει το size κατά ένα ενώ αν δεν υπάρχει κανένα Item στην ουρά κατά την κλήση της εγείρει NoSuchElementException.

Οι άλλες μέθοδοι υλοποιήθηκαν με ίδιο τρόπο με την StringstackImpl.

StringQueueWithOnePointer

Ο τρόπος για να χρησιμοποιηθεί μόνο ένας pointer είναι να χρησιμοποιηθεί κυκλική ουρά και ο pointer δείχνει στο tail της ουράς. Έτσι, όταν ένα Item γίνεται get, έχουμε πρόσβαση σε αυτό μέσω του tail.next, που δείχνει στην αρχή της ουράς, ενώ όταν ένα Item γίνεται put, τοποθετείται στο τέλος της κυκλικής ουράς και ανατίθεται στο tail.

Δηλώθηκε ένας δείκτης σε Node με όνομα tail που δείχνει στο τελευταίο στοιχείο της κυκλικής ουράς, δηλαδή στο στοιχείο που προστέθηκε πιο πρόσφατα και αρχικοποιήθηκε σε null. Δηλώθηκε ένας int με όνομα size, που μετρά πόσα στοιχεία έχει ανά πάσα στιγμή η στοίβα και αρχικοποιήθηκε με 0.

Η μέθοδος put(Item) τοποθετεί ένα νέο στοιχείο στο πέρας της κυκλικής ουράς. Αν η ουρά είναι άδεια, τότε αναθέτει στο tail ένα νέο Node που περιέχει το Item που περάστηκε σαν όρισμα, και θέτει το tail.next στο tail, δηλαδή στον εαυτό του, κάνοντας έτσι την ουρά κυκλική. Διαφορετικά, δημιουργεί ένα νέο Node που περιέχει το Item που περάστηκε σαν όρισμα και θέτει το next του στο tail.next, που είναι το πρώτο Node της ουράς. Τέλος, θέτει το tail.next να δείχνει σε αυτό το νέο Node και το ίδιο το tail να δείχνει πλέον σε αυτό. Αυξάνει το size κατά ένα.

Η μέθοδος get() αφαιρεί το Item που προστέθηκε πρώτο στην ουρά και το επιστρέφει. Αν η ουρά έχει ένα μόνο στοιχείο, το αφαιρεί και θέτει το tail ίσο με null. Διαφορετικά θέτει το tail.next (το Node που θέλουμε να αφαιρέσουμε) στο tail.next.next (το επόμενο από το Node που θέλουμε να αφαιρέσουμε). Μειώνει το size κατά ένα και αν δεν υπάρχει κανένα Item στην ουρά κατά την κλήση της εγείρει NoSuchElementException.

Μέρος Β

Αρχικά διαβάζεται το path για το .txt αρχείο που δίνεται ως όρισμα κατά την εκτέλεση του Thiseas.java, με τέτοιο τρόπο ώστε να υποστηρίζονται τα κενά. Στη συνέχεια δημιουργείται File με το δοσμένο path, αν υπάρχει. Με έναν Scanner αποθηκεύεται η πρώτη γραμμή σε String (αν υπάρχει). Το String αυτό γίνεται split με βάση τον χαρακτήρα κενού, και ελέγχεται ότι έχει ακριβώς δύο στοιχεία. Τα στοιχεία αυτά ανατίθεται στις στατικές μεταβλητές rows και columns του λαβυρίνθου, αν είναι αριθμητικοί χαρακτήρες. Ακολούθως, αντίστοιχη διαδικασία λαμβάνει χώρα για την απόσπαση των συντεταγμένων του χαρακτήρα ‘E’. Αν οποιοσδήποτε από τους ελέγχους αυτούς δεν περάσει, εμφανίζεται αντίστοιχο μήνυμα και η εφαρμογή τερματίζει.

Το character array maze αρχικοποιείται με μεγέθη τα rows και columns που αποσπάστηκαν από την πρώτη σειρά του .txt και η δηλώνεται ακέραια μεταβλητή countE που αρχικοποιείται σε 0. Ο Scanner διασχίζει σειριακά τα στοιχεία του .txt, μετατρέπει το καθένα σε character και το αναθέτει στην αντίστοιχη θέση του array maze. Αν το στοιχείο είναι κάποιο από τα προβλεπόμενα, συνεχίζει ενώ αν δεν είναι εμφανίζει αντίστοιχο μήνυμα και η εφαρμογή τερματίζει. Μία από τις περιπτώσεις ειδικού χαρακτήρα αποτελεί το ελληνικό ‘Ε’, που περιλαμβανόταν και στο .txt αρχείο sample-input.txt που δόθηκε μαζί με την εργασία. Ενώ το Intellij IDEA, με το οποίο υλοποίησα την εργασία έτρεχε κανονικά και για ελληνικό ‘Ε’, εκτός από αγγλικό, με τους αντίστοιχους ελέγχους, όταν το έτρεξα από command line το ελληνικό ‘Ε’ δεν αναγνωριζόταν, για αυτό αποφάσισα να μη το δέχομαι σαν valid είσοδο. Αν κατά τη διάρκεια γεμίσματος του πίνακα τελειώσει το .txt αρχείο, εμφανίζεται μήνυμα ότι οι διαστάσεις που δόθηκαν δεν αντιστοιχούν στα στοιχεία του λαβυρίνθου και η εφαρμογή τερματίζει. Στη συνέχεια γίνονται έλεγχοι για το αν υπάρχουν υπερβολικά πολλά στοιχεία, αν δεν υπάρχει ο χαρακτήρας ‘Ε’ ή υπάρχουν πολλαπλοί και για το αν ο χαρακτήρας ‘Ε’ βρίσκεται στην προσδιορισμένη θέση.

Σχετικά με την υλοποίηση της αναζήτησης εξόδου από τον λαβύρινθο με backtracking, δημιουργείται ένα mazeStack, που χρησιμοποιεί την υλοποίηση στοίβας του Α μέρους και δέχεται αντικείμενα τύπου Point. Αρχικά, προστίθεται το Point που αντιστοιχεί στις συντεταγμένες του χαρακτήρα ‘Ε’. Όσο η στοίβα δεν είναι άδεια γίνονται οι εξής έλεγχοι:

- Αν βρισκόμαστε σε ακριανή θέση (έλεγχος με την στατική μέθοδο atEdge) και το στοιχείο είναι ‘0’, βρήκαμε λύση, την οποία εμφανίζουμε και η εφαρμογή τερματίζει.
- Αν η θέση ακριβώς πάνω από την τρέχουσα βρίσκεται εντός των διαστάσεων του λαβυρίνθου (έλεγχος με στατική μέθοδο inBounds) και το στοιχείο είναι ‘0’, τότε

προσθέτουμε την τρέχουσα θέση στη στοίβα, θέτουμε το στοιχείο της αντίστοιχης θέσης του πίνακα στο χαρακτήρα ‘1’ και μετακινούμαστε στη νέα θέση.

- Κάνουμε αντίστοιχους ελέγχους για τις θέσεις αμέσως αριστερά, κάτω και δεξιά από την τρέχουσα θέση.
- Αν δεν υπάρχει γειτονική θέση με χαρακτήρα ‘0’ θέτουμε το στοιχείο της αντίστοιχης θέσης του πίνακα στο χαρακτήρα ‘1’ και αναθέτουμε στην τρέχουσα θέση τη θέση που κάνουμε pop από τη στοίβα.

Αν η στοίβα αδειάσει, σημαίνει ότι δεν υπάρχει έξοδος από το λαβύρινθο, εμφανίζεται αντίστοιχο μήνυμα και η εφαρμογή τερματίζει.