Δομές Δεδομένων 2019-20

Project 1

Δημήτρης Δεληκούρας 3170034

Ευάγγελος-Ανδρέας Βενετσάνος 3180019

Μέρος Α:

Υλοποίηση Stack μέσω StringStackImpl

Υλοποίηση Constructor: Αρχικοποίηση ενός ψευδοκόμβου pseudo ίσο με το head της στοίβας καθώς και ενός μετρητή stacksize ως 0.

Υλοποίηση isEmpty: Σύγκριση του pseudo με τον head.

Υλοποίηση push: Αύξηση του μετρητή stacksize κατά 1 και αλλαγή αρχικά του head.next και στην συνέχεια του ίδιου του head της στοίβας στον καινούργιο κόμβο.

Υλοποίηση pop: Αρχικά γίνεται έλεγχος μέσω της μεθόδου isEmpty. Στην συνέχεια γίνεται μείωση του μετρητή stacksize κατά 1 και αλλαγή του head της στοίβας στο head.next. Επίσης επιστρέφεται το πρώην head.

Υλοποίηση peek: Αρχικά γίνεται έλεγχος μέσω της μεθόδου isEmpty. Στην συνέχεια γίνεται επιστροφή του head της στοίβας.

Υλοποίηση printStack: Αρχικά γίνεται έλεγχος μέσω της μεθόδου isEmpty. Στην συνέχεια γίνεται εμφάνιση όλων τον αντικειμένων, ξεκινώντας από το *"νεότερο"*. Μεταξύ 2 διαδοχικών τυπώνεται ο χαρακτήρας " , ".

Υλοποίηση size: Επιστροφή του μετρητή stacksize.

Υλοποίηση Queue μέσω StringQueueImpl

Υλοποίηση Constructor: Αρχικοποίηση ενός ψευδοκόμβου pseudo ίσο με το head και το tail της ουράς καθώς και ενός μετρητή queuesize ως 0.

Υλοποίηση isEmpty: Σύγκριση του head με τον tail.

Υλοποίηση put: Αύξηση του μετρητή queuesize κατά 1 και αλλαγή αρχικά του tail.next και στην συνέχεια του ίδιου του tail της ουράς στον καινούργιο κόμβο.

Υλοποίηση get: Αρχικά γίνεται έλεγχος μέσω της μεθόδου isEmpty. Στην συνέχεια γίνεται μείωση του μετρητή queuesize κατά 1 και αλλαγή του head της ουράς στο head.next. Επίσης επιστρέφεται το πρώην head.

Υλοποίηση peek: Αρχικά γίνεται έλεγχος μέσω της μεθόδου isEmpty. Στην συνέχεια γίνεται επιστροφή του head.next της ουράς.

Υλοποίηση printQueue: Αρχικά γίνεται έλεγχος μέσω της μεθόδου isEmpty. Στην συνέχεια γίνεται εμφάνιση όλων τον αντικειμένων, ξεκινώντας από το *"παλαιότερο"*. Μεταξύ 2 διαδοχικών τυπώνεται ο χαρακτήρας " , ".

Υλοποίηση size: Επιστροφή του μετρητή queuesize.

Μέρος Β:

Το αρχείο *όνομα_αρχείου.txt* στο οποίο περιέχονται τα δεδομένα του λαβύρινθου θα πρέπει να βρίσκεται στο ίδιο directory με τα αρχεία StringStack.java, StringStackImpl.java και Thiseas.java.

Με την εντολή ***java Thiseas όνομα_αρχείου.txt*** ξεκινά η εκτέλεση του προγράμματος.

Τα στοιχεία του αποθηκεύονται ως String[] τύπου:

{γραμμή, στήλη, τιμή (0 ή 1 ή E), δείκτης επίσκεψης (0 αν δεν το έχουμε επισκεφθεί ή 1 αν το έχουμε επισκεφθεί)}

Τα παραπάνω String[] τοποθετούνται στον πίνακα maze. Για τον υπολογισμό του σωστού μονοπατιού, το πρόγραμμα ελέγχει όλες τις γειτονικές θέσεις του σημείου στο οποίο *βρίσκεται* κάθε στιγμή (ξεκινώντας από την είσοδο E), ακολουθώντας την σειρά:

Δεξιά, Αριστερά, Κάτω, Πάνω.

Από αυτές θα επιλέξει την πρώτη που ικανοποιεί τις συνθήκες *τιμή* == 0 και *δείκτης επίσκεψης* == 0. Στην συνέχεια, ο *δείκτης επίσκεψης* του συγκεκριμένου String[] θα γίνει 1, ενώ το ίδιο θα γίνει push στην στοίβα dotStack τύπου StringStackImpl<String[]>. Η παραπάνω διαδικασία επαναλαμβάνεται για κάθε το string[] που προστίθεται στην στοίβα μέχρι το πρόγραμμα να βρεθεί σε αδιέξοδο. Τότε με την χρήση των μεθόδων pop και peek της στοίβας γυρνάμε πίσω στον κοντινότερο σταυροδρόμι και επιλέγουμε μια άλλη κατεύθυνση με τον τρόπο που αναφέραμε παραπάνω. Γίνεται, δηλαδή, backtracking.

Αν τώρα μετά από την επιλογή του κατάλληλου γειτονικού σημείου βρεθούμε σε σημείο με String[] τύπου:

{0, …, 0, 0}, {..., 0, 0, 0}, {n-1, …, 0, 0}, {..., m-1, 0, 0}

Όπου n και m είναι το σύνολο των γραμμών και στηλών του λαβυρίνθου αντίστοιχα. Θα βρισκόμαστε σε μια έξοδο του λαβυρίνθου οπότε τυπώνεται το κατάλληλο μήνυμα και το πρόγραμμα τερματίζεται.

Αλλιώς αν το πρόγραμμα βρεθεί ξανά στην είσοδο E του λαβύρινθου τότε θα έχουν ελεγχθεί όλα τα πιθανά μονοπάτια, δηλαδή δεν θα υπάρχει κάποια έξοδο. Και σε αυτή την περίπτωση τυπώνεται το κατάλληλο μήνυμα και το πρόγραμμα τερματίζεται.

Μέρος Γ:

Υλοποίηση Queue μέσω StringQueueWithOnePointer

Η χρήση ουράς με έναν δείκτη head, είναι εφικτή με την υλοποίηση κυκλικής συνδεδεμένης λίστας. Το τελευταίο αντικείμενο τύπου MyNode έχει ως next το πρώτο.

Υλοποίηση Constructor: Αρχικοποίηση ενός ψευδοκόμβου pseudo ίσο με το head της στοίβας καθώς και ενός μετρητή stacksize ως 0.Το επόμενο αντικείμενο του pseudo ορίζεται ως ο εαυτός του καθώς βρισκόμαστε στην τετριμμένη περίπτωση κυκλικής λίστας με 1 αντικείμενο.

Υλοποίηση isEmpty: Σύγκριση του head με τον ψευδοκόμβο. Αν το head είναι ο pseudo τότε η ουρά είναι κενή

Υλοποίηση put: Αν η ουρά είναι κενή τότε το νέο αντικείμενο παίρνει τη θέση του pseudo και έτσι έχουμε λίστα με 1 πραγματικό αντικείμενο τύπου MyNode. Αν η ουρά δεν είναι κενή, θέτουμε το next του head να είναι το νέο αντικείμενο και μετακινούμε το head στο νέο αντικείμενο. Σε κάθε περίπτωση αυξάνουμε το μετρητή queuesize κατά 1.

Υλοποίηση get:Γίνεται έλεγχος για το αν η ουρά είναι κενή. Αν ναι, έχουμε error καθώς δεν μπορούμε να αφαιρέσουμε αντικείμενο από κενή ουρά. Διαφορετικά,

Υλοποίηση peek: Αρχικά γίνεται έλεγχος μέσω της μεθόδου isEmpty. Στην συνέχεια γίνεται επιστροφή του head.next της ουράς.

Υλοποίηση printQueue:Αρχικά, αν η ουρά είναι κενή, τότε έχουμε error καθώς δεν υπάρχουν αντικείμενα να εμφανιστούν. Διαφορετικά, από το “νεότερο” αντικείμενο MyNode(head.next) , μέχρι το τελευταίο με ένα while loop κάνουμε print κάθε αντικείμενο και αλλάζουμε τον δείκτη μας ώστε να δείχνει στο επόμενο(το loop δεν περιλαμβάνει το παλαιότερο MyNode,οπότε το κάνουμε print εκτός του loop).

Υλοποίηση size: Επιστροφή του μετρητή queuesize.