Μυρτώ-Χριστίνα Ελευθέρου| 3170046

Χρυσούλα Οικονόμου|3170127

1η Εργασία

Μέρος Α

StringStackImpl:

- *isEmpty()*: Αν το head είναι *null*, τότε η στοίβα είναι άδεια και η μέθοδος επιστρέφει *true*, αλλιώς *false*.
- *push(String item)*: Ο νέος κόμβος εισέρχεται στην κορυφή της στοίβας, και ο δείκτης *head* δείχνει πλέον στον νέο κόμβο.
- *pop()*: Αν η στοίβα είναι άδεια, τότε εμφανίζεται *exception*. Διαφορετικά, γίνεται έλεγχος αν η στοίβα περιέχει μόνο ένα στοιχείο, αν ναι, το αφαιρούμε και επιστρέφουμε την τιμή του και στη συνέχεια ορίζουμε το *head* ίσο με *null*, αφού η στοίβα είναι πλέον άδεια, αν όχι, αφαιρούμε το στοιχείο που δείχνει το *head*(κορυφή της στοίβας) και το επιστρέφουμε.
- *peek()*: Όμοια με την *pop()*, αν η στοίβα είναι άδεια τότε εμφανίζεται *exception*. Αλλιώς η μέθοδος θα εμφανίσει το στοιχείο στην κορυφή της στοίβας.
- *printStack(PrintStream stream)*:Αν η στοίβα είναι άδεια, τότε εμφανίζεται αντίστοιχο μήνυμα, αλλιώς, ξεκινώντας από τον κόμβο στην κορυφή της στοίβας, εκτελείται ένα *while loop* μέχρις ότου ο τρέχων κόμβος να δείχνει σε *null*, που σημαίνει ότι έφτασε στον τελευταίο κόμβο. Σε κάθε επανάληψη το *currentNode* παίρνει την διεύθυνση του κόμβου στον οποίο δείχνει, δηλαδή τον αμέσως επόμενο κόμβο.
- *size()*: Στις μεθόδους εισαγωγής και εξαγωγής κόμβων(*push(), pop()*), ορίζουμε έναν μετρητή(*counter*), ο οποίος αυξάνεται ή μειώνεται κατά 1 αντίστοιχα. Η *size()*, επιστρέφει την τιμή του μετρητή ο οποίος υποδεικνύει το μέγεθος της στοίβας.

StringQueueImpl:

- *isEmpty()*:Αν το head είναι *null*, τότε η ουρά είναι άδεια και η μέθοδος επιστρέφει *true*, αλλιώς *false*.
- *put(String Item):*Αν η ουρά είναι άδεια, τότε τα *head* και *tail* θα δείχνουν στον νέο κόμβο που εισάγουμε, αλλιώς μόνο το *tail* θα δείχνει στον τελευταίο κόμβο που μπήκε στην ουρά.
- *get()*:Αν η ουρά είναι άδεια, εμφανίζεται *exception*, αλλιώς γίνεται έλεγχος αν η ουρά περιέχει μόνο ένα στοιχείο, αν ναι, το αφαιρούμε και επιστρέφουμε την τιμή του και στη συνέχεια ορίζουμε το *head* και το *tail* ίσο με *null*, αφού η ουρά είναι πλέον άδεια, αν όχι, αφαιρούμε το στοιχείο που δείχνει το *head*(αρχή της ουράς) και το επιστρέφουμε.
- *peek()*: Όμοια με την *get()*, αν η ουρά είναι άδεια τότε εμφανίζεται *exception*. Αλλιώς η μέθοδος θα εμφανίσει το στοιχείο αρχή της ουράς(εκεί που δείχνει το *head*).
- *printQueue(PrintStream stream)*:Αν η ουρά είναι άδεια, τότε εμφανίζεται αντίστοιχο μήνυμα, αλλιώς, ξεκινώντας από τον κόμβο το *head*, εκτελείται ένα *while loop* μέχρις ότου ο τρέχων κόμβος να δείχνει σε *null*, που σημαίνει ότι έφτασε στον τελευταίο κόμβο(*tail*). Σε κάθε επανάληψη το *currentNode* παίρνει την διεύθυνση του κόμβου στον οποίο δείχνει, δηλαδή τον αμέσως επόμενο κόμβο.
- *size()*: Στις μεθόδους εισαγωγής και εξαγωγής κόμβων(*put(), get()*), ορίζουμε έναν μετρητή(*counter*), ο οποίος αυξάνεται ή μειώνεται κατά 1 αντίστοιχα. Η *size()*, επιστρέφει την τιμή του μετρητή, ο οποίος υποδεικνύει το μέγεθος της ουράς.

Μέρος Γ

StringQueueWithOnePointer:

Σε αυτή την ουρά, επιτυγχάνουμε τη χρήση ενός μόνο δείκτη χρησιμοποιώντας κυκλική λίστα μονής σύνδεσης, αφού ο τελευταίος κόμβος(αυτός που εισήχθη τελευταίος, δείχνει στο πρώτο στοιχείο της ουράς.

- *isEmpty()*: Αν η τιμή του *counter* ισούται με 0 επιστρέφει *true*, αλλιώς *false*.

- *put(String item)*: Αν η ουρά είναι άδεια τότε το *last* δείχνει στον κόμβο που εισάγουμε, ο οποίος είναι και η αρχή της ουράς, αλλιώς κάθε φορά που εισάγουμε ένα νέο κόμβο το *last* θα δείχνει σε αυτόν, και ο κόμβος αυτός θα δείχνει στον πρώτο κόμβο της ουράς.
- *get()*:Αν η ουρά είναι άδεια, τότε εμφανίζεται *exception*, αλλιώς αφαιρούμε τον πρώτο κόμβο, και ο τελευταίος κόμβος στον οποίο δείχνει το *last*, θα δείχνει πλέον εκεί που έδειχνε προηγουμένως ο πρώτος κόμβος.
- *peek()*:Αν η ουρά είναι άδεια, τότε εμφανίζεται *exception*, αλλιώς εμφανίζεται το πρώτο στοιχείο της ουράς.
- *printQueue(PrintStream stream)*:Αν η ουρά είναι άδεια, τότε εμφανίζεται αντίστοιχο μήνυμα, αλλιώς εκτελείται ένα *while loop* τόσες φορές, όσο είναι και το μέγεθος της λίστας την τρέχουσα στιγμή.
- *size()*: Στις μεθόδους εισαγωγής και εξαγωγής κόμβων(*put(), get()*), ορίζουμε έναν μετρητή(*counter*), ο οποίος αυξάνεται ή μειώνεται κατά 1 αντίστοιχα. Η *size()*, επιστρέφει την τιμή του μετρητή, ο οποίος υποδεικνύει το μέγεθος της ουράς.

Μέρος Β

- Το πρόγραμμα τρέχει από το *command line* και το αρχείο περιέχεται στον φάκελο *src* μαζί με τα υπόλοιπα αρχεία *.java*

Διαβάζοντας το αρχείο, αποθηκεύουμε όλα τα στοιχεία του στον πίνακα *tempMaze*[][]

Χρησιμοποιώντας την υλοποίηση της *StringStack*, δημιουργούμε δύο στοίβες, την *coords*, στην οποία αποθηκεύονται οι τιμές των συντεταγμένων τις οποίες έχουμε διασχίσει, και την *zeros*, στην οποία αποθηκεύεται ο αριθμός των μηδενικών που υπάρχουν γύρω από ένα συγκεκριμένο σημείο.

Στην συνέχεια εντοπίζουμε το αμέσως επόμενο 0 και αποθηκεύουμε τις τιμές του. Για κάθε κίνηση(δεξιά, αριστερά, πάνω, κάτω)

εκτελούμε ένα *while loop*, μέχρι να φτάσει σε κάποιο αδιέξοδο. Κατά την διάρκεια της επανάληψης, χρησιμοποιούμε τις 2 στοίβες για να αποθηκεύσουμε τις συντεταγμένες και τα 0 γύρω από τα σημεία, ενώ επίσης αποθηκεύουμε και την κίνηση την οποία κάναμε, έτσι ώστε να μην χρειαστεί να κάνει την ίδια κίνηση ξανά(π.χ. αν χρειαστεί να πάει πάνω ενώ στο σημείο αυτό είχε φτάσει ερχόμενος από πάνω, τότε να μην πάει προς τα πάνω αλλά να συνεχίσει την αναζήτηση για δεξιά ή αριστερά). Όταν φτάσει σε αδιέξοδο χρησιμοποιεί την στοίβα *zeros*, για να εντοπίσει το προηγούμενο στοιχείο το οποίο είχε γύρω του 3 ή 4 μηδενικά, δηλαδή, υπάρχει και άλλη κατεύθυνση την οποία μπορεί να ακολουθήσει. Παράλληλα χρησιμοποιεί την *coords* έτσι ώστε να μετατρέψει όλα τα 0 που οδήγησαν στο αδιέξοδο αυτό, σε 1 για να μην επιστρέψει εκεί σε μελλοντική αναζήτηση. Τα σημεία που έχουν λιγότερα από 3 μηδενικά, εξάγονται από την λίστα *zeros*, καθώς επίσης και οι συντεταγμένες τους εξάγονται από την *coords*. Όταν η τιμή ενός στοιχείου της στοίβας *zeros* ισούται με 3 ή 4, σταματάει την αναζήτηση και αποθηκεύει τις συντεταγμένες του σημείου αυτού στις μεταβλητές *row* και *col*. Αν το σημείο αυτό είναι το αρχικό σημείο από το οποίο ξεκινήσαμε τότε δεν αποθηκεύουμε την τελευταία κίνηση που κάναμε(δεξιά, αριστερά κλπ) έτσι ώστε να μπορεί να κινηθεί προς την ίδια κατεύθυνση αν χρειαστεί. Αν φτάσει σε κάποιο 0 το οποίο βρίσκεται στα άκρα του πίνακα, τότε η επανάληψη σταματάει και επιστρέφει τις συντεταγμένες του σημείου.