Μαρία - Ιωάννα Μουζουράκη p3200105

Εργασία 1

Μέρος Δ - Αναφορά παράδοσης

1. Μέρος Α :

StringStackImpl :

isEmpty: Επιστρέφει true αν το μέγεθος της στοίβας ισούται με 0.

push: Δημιουργούμε έναν νέο κόμβο και του δίνουμε την τιμή που δώθηκε ως παράμετρος στην κλήση της push (node.setData(item)).Αν η στοίβα ήταν άδεια ,ο κόμβος που ήταν head της στοίβας θα είναι τώρα ο επόμενος του κόμβου που προσθέτουμε (node.setNext(head)).Θέτουμε ως head της στοίβας τον κόμβο που προσθέτουμε (head = node).

pop: Αν η στοίβα είναι άδεια,εμφανίζει error.Αποθηκεύουμε στη μεταβλητή remove_item την τιμή του head (T removed_item = head.getData()) και την επιστρέψουμε (return removed_item).Αφαιρούμε το head (head = head.getNext()), οπότε το μέγεθος της στοίβας μειώνεται.

peek: Αν η στοίβα είναι άδεια,εμφανίζει error.Επιστρέφουμε την τιμή του head (return head.getData()).

printStack: Ξεκινώντας από το head , όσο ο κόμβος στον οποίο βρισκόμαστε έχει επόμενο κόμβο (node.getNext() != null) , τυπώνουμε την τιμή του (stream.println(node.getData())) και προχωράμε στον επόμενο κόμβο (node = node.getNext()).Αν η στοίβα είναι άδεια τυπώνει αντίστοιχο μήνυμα.

size: Επιστρέφει το μέγεθος της στοίβας , το οποίο αυξάνεται ή μειώνεται όταν βάζουμε ή βγάζουμε κόμβους από αυτήν.

StringQeueImpl :

isEmpty : Επιστρέφει true αν το μέγεθος της ουράς ισούται με 0

put: Δημιουργούμε έναν νέο κόμβο και του δίνουμε την τιμή που δώθηκε ως παράμετρος στην κλήση της put (node.setData(item)).Αυτός ο κόμβος είναι πλέον το tail της ουράς (tail = node).Εάν η ουρά ήταν άδεια, ο κόμβος θα είναι και το head (head = node).Αν είχε μόνο έναν κόμβο , τότε ο κόμβος που προσθέτουμε θα είναι ο επόμενος του head (head = node) , ενώ αν δεν ισχύει καμία από τις παραπάνω περιπτώσεις θα είναι ο επόμενος του tail (tail.setNext(node)).

get: Αν η ουρά είναι άδεια,εμφανίζει error.Αποθηκεύουμε στη μεταβλητή removed_item την τιμή του head (T removed_item = head.getData()) και την επιστρέψουμε (return removed_item).Αφαιρούμε το head (Αν έχει παραπάνω από έναν κόμβο : head = head.getNext() ,διαφορετικά : (head = null)), οπότε το μέγεθος της ουράς μειώνεται.

peek: Αν η ουρά είναι άδεια,εμφανίζει error.Επιστρέφουμε την τιμή του head (return head.getData()).

printQueue: Ξεκινώντας από το head , όσο ο κόμβος στον οποίο βρισκόμαστε έχει επόμενο κόμβο (node.getNext() != null) , τυπώνουμε την τιμή του (stream.println(node.getData())) και προχωράμε στον επόμενο κόμβο (node = node.getNext()).Αν η ουρά είναι άδεια τυπώνει αντίστοιχο μήνυμα.

size: Επιστρέφει το μέγεθος της ουράς, το οποίο αυξάνεται ή μειώνεται όταν βάζουμε ή βγάζουμε κόμβους από αυτήν.

Μέρος Γ :

StringQeueWithOnePointer :

Για να αποφύγουμε την χρήση δύο μεταβλητών , χρησιμοποιήσαμε λίστα κυκλικής σύνδεσης , θέτωντας το head επόμενο κόμβο του tail.Οπότε όπου χρησιμοποιούμε το tail.getNext() ,πηγαίνουμε στο head της ουράς.Οι μαθόδοι isEmpty, peek, printQueue και size είναι

ίδιες με αυτές της StringQeueImpl.

put :Δημιουργούμε έναν νέο κόμβο και του δίνουμε την τιμή που δώθηκε ως παράμετρος στην κλήση της put (node.setData(item)).Αυτός ο κόμβος είναι πλέον το tail της ουράς (tail = node).Εάν η ουρά έχει μόνο έναν κόμβο ο κόμβος που ήταν το tail είναι τώρα head (node.setNext(tail)) και ο κόμβος που προστίθεται είναι το νέο tail (tail.setNext(node)).Αν έχει παραπάνω από έναν κόμβο , ο κόμβος που προστέθηκε είναι το νέο tail οπότε το συνδέουμε με το head (node.setNext(tail.getNext())), και συδέουμε μαζί του τον κόμβο που ήταν tail (tail.setNext(node)).

get: Αν η ουρά είναι άδεια,εμφανίζει error.Αποθηκεύουμε στη μεταβλητή removed_item την τιμή του head (T removed_item = tail.getNext().getData()) και την επιστρέψουμε (return removed_item).Αφαιρούμε τον κόμβο που ήταν το tail ως εξής : Αν υπάρχει μόνο ένας κόμβος στην ουρά πλέον δεν θα υπάρχει κάνενα οπότε tail = null. Αν υπάρχουν δύο , θα μείνει το head οπότε δεν θα συνδέεται πλέον με κάποιν άλλον άρα tail.setNext(null). Αν δεν ισχύει κάποια από τις παραπάνω περιπτώσεις ο κόμβος που βρίσκεται μετά το head (tail.getNext().getNext()) είναι πλέον το head οπότε tail.setNext(tail.getNext().getNext()).

2. Μέρος Β :

Από την εκφώνηση της εργασίας γνωρίζουμε ότι εάν βρεθούμε σε αδιέξοδο θα πρέπει να γυρίσουμε πίσω , το οποίο σημαίνει πως θα πρέπει να θυμώμαστε τις προηγούμενες κινήσεις που κάναμε . Για αυτόν τον σκοπό χρησιμοποιούμε την υλοποίησης της στοίβας από το Μέρος Α.Κάθε φορά που εκτελεί ο χρήστης μία κίνηση , οι συντεταγμένες της κίνησης αποθηκεύονται σε μία στοίβα.Οπότε πάνω-πάνω στην στοίβα θα υπάρχει η πιο πρόσφατη κίνηση.Όταν φτάνουμε σε αδιέξοδο ,γυρνάμε σταδιακά προς τα πίσω , πέρνοντας από την στοίβα τις συντεταγμένες της προηγούμενης κίνησης , οι οποίες θα είναι οι πρώτες δύο τιμές της στοίβας .Αφού κάνουμε μία κίνηση προς

τα πίσω αφαιρούμε από την στοίβα τις δύο αυτές συντεταγμένες και ρωτάμε τον χρήστη αν θέλει να συνεχίσιε να κινέιται προς τα πίσω . Αν ναι , συνεχίζουμε την ίδια διαδικασία.