1η Εργασία Δομών Δεδομένων

Το αρχείο CircularQueue.java αρχικα κάνει implement το interface Queue<E> οπου E το element που χρησιμοποιείται αυτη την στιγμή.

Fields

΄Υστερα, δημιουργούμε ως μεταβλητές το αρχικό μέγεθος του circular queue, καθώς και το front, το rear, τον αριθμό των στοιχείων που περιέχονται στο queue κάθε στιγμή (μεταβλητή current_size) και το array E[] το οποίο είναι ουσιαστικά η ίδια η queue.

Constructor

Μετά από αυτό υπάρχει το constructor μέσα στο οποίο οι μεταβλητές front και rear αρχικοποιούνται ως -1 και το current_size ως 0 για να δείξουμε ότι η queue είναι κενή όταν πρωτοδημιουργήται. Επίσης εντός του constructor η queue αρχικοποιήται στο αρχικό της μέγεθος αμέσως μετά τη δημιουργία της.

Methods

isEmpty()

Στη συνέχεια έχουμε τη μέθοδο isEmpty η οποία επιστρέφει boolean αποτέλεσμα (true/false) με βάση το αν η μεταβλητή current_size είναι 0, δηλαδή αν το queue είναι άδειο (δεν έχει στοιχεία). Αντίστοιχη είναι και η μέθοδος αμέσως μετά, ονόματι isFull.

isFull()

Η isFull είναι μια μέθοδος η οποία βασίζει το αποτέλεσμα της στο αν ο αριθμός των στοιχείων είναι μεγαλύτερος ή ίσος του μεγέθους της queue (μιλώντας για τη στιγμή του ελέγχου πάντα, αφού και τα δύο αυτά μεγέθοι είναι μεταβλητά).

size()

Αμέσως μετά, η μέθοδος size η οποία απλά μας επιστρέφει την int τύπου μεταβλητή current_size (αριθμος στοιχείων στο queue).

push(E elem)

Η μέθοδος push έχει σειρά, η μέθοδος αυτή παίρνει ως argument μια E type μεταβλητή ονόματι elem, ιδίου τύπου με της queue. Πρώτα εντός ενός if χρησιμοποιείται η μέθοδος isFull για να δει εάν η queue είναι γεμάτη και σε αυτή τη περίπτωση να τη διπλασιάσει σε μέγεθος με τη χρήση μιας άλλης μεθόδου που θα δούμε στη συνέχεια, ονόματι doubleSize. Στη συνέχεια αλλάζει τη rear σε (rear + 1) mod Queue Length, θέτει στη queue με index rear το elem και αυξάνει κατα ένα την current_size. Τέλος με τη χρήση μιας ακόμη if ελέγχει εάν η front είναι -1 και την θέτει ως 0 σε αυτή την περίπτωση.

pop()

΄Υστερα, η μέθοδος pop, τύπου E, ελέγχει αρχικά, μέσω μιας if και με τη χρήση της μεθόδου isEmpty αν η queue είναι κενή, σε αυτή τη περίπτωση εμφανίζει το exception "Queue is empty; no elements to dequeue.". Στη συνέχεια, η μεταβλητή τύπου E ονόματι dequeuedElement παίρνει το value που έχει η queue στη θέση με index front και ύστερα το ίδιο αυτό value γίνεται null. Αμέσως μετά η front αλλάζει σε (front + 1) mod Queue Length και η current_size μειώνεται κατα 1. Αμέσως μετά, μέσω μιας ακόμη if και

τη χρήση μιας μεθόδο ονόματι isQuarterFilled που θα δούμε στη συνέχεια, ελέγχουμε αν η queue είναι κατα το ¼ γεμάτη (ή λιγότερο) και σε αυτή την περίπτωση την υποδιπλασιαζουμε σε μέγεθος με τη χρήση μιας μεθόδου ονόματι halfSize που επίσης θα δούμε στη συνέχεια. Τέλος επιστρέφουμε τη dequeuedElement.

first()

Η μέθοδος που ακολουθεί ονομάζετε first, είναι τύπου E και επιστρέφει το value της queue με index front.

clear()

Αμέσως μετά βρήσκεται η clear μέθοδος η οποία απλός αρχικοποιεί τις τιμές των μεταβλητών front, rear και current_size με τον ίδιο τρόπο όπως ο constructor νωρίτερα. Δεν επηρεάζει λοιπόν το μέγεθος της queue, απλώς την “αδειάζει”.

doubleSize()

Σειρά έχει η μέθοδος doubleSize, στην οποία αρχικά δημιουργούμε μια local μεταβλητή τύπου int ονόματι new_length και αρχικοποιείται ως Queue Length * 2. Στη συνεχεια δημιουργούμε μια νέα local queue (array) τύπου E με μέγεθος new_length ονόματι Temp_Circular_Q και ύστερα τοποθετούμε σε εκείνη τα στοιχεία της παλαιάς ουράς φροντίζοντας να κρατήσουν τη σειρά τους. Τέλος θέτουμε την Circular_Queue ως Temp_Circular_Q.

isQuarterFilled()

΄Υστερα βρίσκουμε την μέθοδο isQuarterFilled, είναι τύπου boolean και επιστρέφει το αποτέλεσμα της με βάση το αν η current_size*4 είναι μικρότερη ή ίση του Queue Length.

halfSize()

Στη συνέχεια βρίσκεται η μέθοδος halfSize στην οποία αρχικά δημιουργούμε μια local μεταβλητή τύπου int ονόματι new_length και την αρχικοποιούμε ως Queue Length / 2. Στη συνεχεια δημιουργούμε μια νέα local queue (array) τύπου E με μέγεθος new_length ονόματι Temp_Circular_Q και ύστερα τοποθετούμε σε εκείνη τα στοιχεία της παλαιάς ουράς φροντίζοντας να κρατήσουν τη σειρά τους. Τέλος θέτουμε την Circular_Queue ως Temp_Circular_Q, τη front ως 0 και τη rear ως current_size - 1.

getFront()

Στη συνέχεια υπάρχει και η μέθοδος getFront, η οποία επιστρέφει τη τιμή της front εκείνη την στιγμή. Την χρησιμοποιούμε κυρίως στα Tests.

getRear()

Ομοίως η μέθοδος getRear, η οποία επιστρέφει τη τιμή της rear εκείνη την στιγμή. Την χρησιμοποιούμε κυρίως στα Tests.

getCircular_QueueLength()

Πριν το τέλος υπάρχει και η μέθοδος getCircular_QueueLength, η οποία επιστρέφει το μήκος του Queue εκείνη την στιγμή. Την χρησιμοποιούμε κυρίως στα Tests.

toString()

Η τελευταία μας μέθοδος είναι μια overwritten toString η οποία είναι τύπου String και επιστρέφει σε απλοποιημένη μορφή το Queue.

Comments

Τα comments εντός του κώδικα αυτού του αρχείου είναι αποκλειστικά για troubleshooting και δοκιμαστικούς λόγους ή για την περίπτωση που θέλουμε να αλλάξουμε κάποια παράμετρο της εργασίας. Για παράδειγμα, να δούμε που είναι το front και το rear κάθε στιγμή μετά και πριν από ένα push ή pop ή να ελέγξουμε αν μετά τη σμίκρυνση της queue το μέγεθος της είναι ακόμα πολλαπλάσιο του 2 (στην περίπτωση που δεν αρχικοποιηθεί ως δύναμη του 2 από εμάς τους ίδιους μπορεί να προκύψει μια τέτοια κατάσταση) έτσι ώστε στην επόμενη σμίκρυνση να μην έχουμε προβλήματα.

Tests

Το αρχείο CircularQueueTest.java περιέχει και τα 3 Test μας και χρησιμοποιείται για να κάνουμε test τη λειτουργικότητα του implementation μας.

Test 1

Το πρώτο δημιουργήθηκε κατα τη διάρκεια του εργαστηριακού μαθήματος, άρα δεν θα το αναλύσουμε.

Test 2

Το δεύτερο υπάρχει για να ελέγξει εάν η queue πληρεί τις προϋποθέσεις του ερωτήματος 3 “Επεκτάσιμη Ουρά”.

Test 3
Το τρίτο υπάρχει για να επιβεβαιωθεί πως η queue που δημιουργήται με τη χρήση του κώδικα μας, είναι όντως κυκλικής μορφής. Το επιτυγχάνει αυτό ελέγχοντας ότι, χωρίς να γεμίσει πλήρως η queue (και να διπλασιαστεί σε μέγεθος λόγω αυτού), η rear θα μπορέσει να κάνει τον “κύκλο” της queue και να επιστρέψει στη θέση 0 (μετά από κάποια push/pop).

Συνάδελφοι της εργασίας

=={ it22003, Δημήτριος Αναγνωστόπουλος }==
=={ it22007, Αλέξανδρος Ανδρούτσος }==
=={ it22148, Γεώργιος Δημητρόπουλος }==

GitHub Repository

! Ευχαριστούμε για τον χρόνο σας !