Project 1

Χρύσα Μαυράκη – Ελευθέριος Χατζηαράπης

Χρύσα Μαυράκη -> p3130128

Ελευθέριος Χατζηαράπης -> p3130225

Υλοποίηση της κλάσης ListNode.java για την αναπαράσταση μιας συνδεμένης λίστας που ακολουθείτε από ένα πεπερασμένο σειριακό αριθμό κόμβων.

❶Δημιουργούμε τα πεδία Object data,το οποίο ειναι η πληροφορία που περιέχει ο κόμβος, και ListNode next που ειναι μια αναφορά στο αντικείμενο της ίδιας κλάσης (αναφερόμενη δομή δεδομένων) και υποδεικνύει τον αμέσως επόμενο κόμβο που «κοιτάει» ο συγκεκριμένος κόμβος.

❷Υλοποιούμε τον κατασκευαστή ListNode με παραμέτρους ,ο πρώτος με ένα (int data) και ο δεύτερος με ένα (int data,ListNode next) για την δημιουργία των κόμβων.

❸Υλοποιούμε τις μεθόδους getData() και την getNext() που μας επιστρέφουν την αποθηκευμένη πληροφορία του κόμβου ή τον αμέσως επόμενο κόμβο στον οποίο συνδέεται ο συγκεκριμένος κόμβος αντίστοιχα.

Υλοποίηση της κλάσης IntQueueImpl.java χρησιμοποιώντας τη διασύνδεση της IntQueue.java

❶Εισάγουμε τη διασύνδεση(implements) στη κλάση οπότε υποχρεωτικά πρέπει να υλοποιήσουμε όλες τις μεθόδους που έχει η διασύνδεση.

❷Δημιουργούμε τα πεδία firstnode,lastnode ώστε να έχουν ισχύη στο υπάρχον αντικείμενο.

❸Υλοποίηση του κατασκευαστή IntQueueImpl() ώστε να δημιουργήτε ο κομβος της λιστας.Επίσης αρχικοποιεί τα πεδία firstnode,lastNode με null.

Υλοποίηση Μεθόδων

1. boolean isEmpty()

Ελέγχει αν το firstNode δείχει σε null.Αν ο πρώτος κόμβος δείχνει σε null τότε επιστρέφει την τιμή true δηλαδή ότι ο πίνακας ειναι κενός.Σε αντίθετη περίπτωση επιστρέφει false.

2. void put(int item)

Άν σε περίπτωση η λίστα ειναι κενή τότε δημιουργεί ένα κόμβο με όρισμα τα δεδομένα που θέλουμε να περάσουμε και κάνει το lastNode να γίνει firstNode.

Σε αντίθετη περίπτωση καταχωρεί στο lastNode.next ένα κομβο αντίστοιχο με το προηγούμενο και μετά κάνει το lastNode ισο με το lastNode.next(τον αμέσως επόμενο κόμβο).

3. int get() throws NoSuchElementException

Άν σε περίπτωση η λίστα ειναι κενή τότε «πετάει» μια εξαίρεση η οποία δείχνει ότι παρουσιάστηκε κάπιο σφάλμα κατά την εκτέλεση(δε βρέθηκαν κόμβοι).

Σε αντίθετη περίπτωση δημιουργείται ένας προσωρινός κόμβος που γίνεται firstNode.Μετά το firstNode(κεφαλή)

γίνεται ισο με τον επόμενο του προσωρινού κόμβου και τέλος επιστρέφει τα δεδομένα του προσωρινού κόμβου.

4. int peek() throws NoSuchElementException

Άν σε περίπτωση η λίστα ειναι κενή τότε «πετάει» μια εξαίρεση η οποία δείχνει ότι παρουσιάστηκε κάπιο σφάλμα κατά την εκτέλεση(δε βρέθηκαν κόμβοι).

Σε αντίθετη περίπτωση επιστρέφει τα δεδομένα της κεφαλής(firstNode.getData()).

5. void printQueue (PrintStream stream)

Άν σε περίπτωση η λίστα ειναι κενή τότε εκτυπώνει μέσα από το stream οτι η η λίστα είναι άδεια(“Empty Queue”) και μετά κλείνει την μέθοδο.

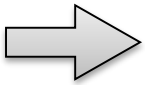
Άν όχι τότε δημιουργεί ένα προσωρινό κόμβο που είναι ίσο με το firstNode.Εκτελούμε την επανάληψη while μέχρι ο προσωρινός κόμβος να γίνει null(δηλαδή να φτάσει ο προσωρινός κόμβος στο τέλος της λίστας) και μέσω του stream εκτυπώνουμε τα δεδομένα του προσωρινού κόμβου και μετά θα δείχνει στον αμέσως επόμενο του.

6. int size()

Άν σε περίπτωση η λίστα ειναι κενή τότε κλείνει η μέθοδος επιστρέφοντας το 0 δηλαδή ότι περιέχει μηδέν στοιχεία.

Άν όχι τότε αρχικοποιούμε μια μεταβλητή int i=1,μετά δημιουργούμε ένα προσωρινό κόμβο που είναι ίσο με τον επόμενο της κεφαλής(Αυτό γίνεται γιατί θεωρούμε ότι έχουμε ήδη μετρήσει τον πρωτο κόμβο,γι ‘αυτό γράψαμε και i=1).Τώρα τρέχουμε την while μέχρι ο επόμενος το προσωρινου να δείχνει σε null(δηλαδη να μην υπάρχει επόμενος).Αυξανει το i κατά 1 και μετά κάνει το προσωρινό

κόμβο ισο με τον επόμενο του.Μετά τελιωνει μέθοδος επιστρέφοντας το μέγεθος της λίστας.

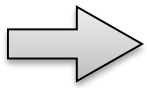


Υλοποίηση της κλάσης DListNode.java για την αναπαράσταση μιας συνδεμένης λίστας που ο κάθε κόμβος συνδέεται με τον επόμενο του αλλά και το προηγούμενο του.

❶Εδώ η υλοποίση είναι περίπου η ίδια με την ListNode μόνο που προσθέτουμε και έναν επιπλέον κόμβο , τον ListNode back που υποδεικνύει τον αμέσως προηγούμενο κόμβο του συγκεκριμένου.

❷ Υλοποιούμε τον κατασκευαστή DListNode με παραμέτρους (int data) και ο δεύτερος ένα (int data,DListNode next,DListNode back) για την δημιουργία των κόμβων.

❸ Προστιθεται μια ακόμα μέθοδος σε σχέση με την ListNode ,η getBack() που επιστρέφει τον προηγούμενο κόμβο.



Υλοποίηση της κλάσης IntDoubleEndedQueueImpl.java χρησιμοποιώντας τη διασύνδεση της IntDoubleEndedQueue.java

❶Εισάγουμε τη διασύνδεση(implements) στη κλάση οπότε υποχρεωτικά πρέπει να υλοποιήσουμε όλες τις μεθόδους που έχει η διασύνδεση.

❷Δημιουργούμε τα πεδία firstnode,lastnode ώστε να έχουν ισχύη στο υπάρχον αντικείμενο.

❸Υλοποίηση του κατασκευαστή IntDoubleEndedQueueImpl() ώστε να δημιουργήτε ο κομβος της λιστας.Επίσης αρχικοποιεί τα πεδία firstnode,lastNode με null.

Υλοποίηση Μεθόδων

1. boolean isEmpty()

Ίδια λογική με αυτή της IntQueueImpl.java

2. void addFirst(int item)

Άν σε περίπτωση η λίστα ειναι κενή τότε δημιουργεί ένα κόμβο με όρισμα τα δεδομένα που θέλουμε να περάσουμε και να είναι ίσο με το firstNode και το lastNode.

Σε αντίθετη περίπτωση καταχωρεί στο firstNode ένα κόμβο τύπου DListNode με ορίσματα τα δεδομένα,την κεφαλίδα(firstNode) και null που είναι ο προηγουμενος του (back).

3. int removeFirst(int item) throws NoSuchElementException

Άν σε περίπτωση η λίστα ειναι κενή τότε «πετάει» μια εξαίρεση η οποία δείχνει ότι παρουσιάστηκε κάπιο σφάλμα κατά την εκτέλεση(δε βρέθηκαν κόμβοι).

Σε αντίθετη περίπτωση αποθηκέυει σε μια μεταβλητή τα δεδομένα της κεφαλίδας μετά ορίζει την κεφαλίδα ως ο ακριβώς επόμενος του και μετά επιστρέφει την τιμή(Αυτό γίνεται γιατί αν ορίσουμε την κεφαλίδα με τον επόμενο του δε θα μπορούμε να έχουμε πρόσβαση στο πρωτο δεδομένο που θέλουμε να πάρουμε).

4. void addLast(int item)

Άν σε περίπτωση η λίστα ειναι κενή τότε δημιουργεί ένα κόμβο με όρισμα τα δεδομένα που θέλουμε να περάσουμε και να είναι ίσο με το firstNode και το lastNode.

Σε αντίθετη περίπτωση καταχωρεί στο firstNode ένα κόμβο τύπου DListNode με ορίσματα τα δεδομένα,null για την κεφαλίδα(firstNode) και το προηγούμενο του τελευταίου κόμβου.

5. **int removeFirst(int item) throws NoSuchElementException**

Άν σε περίπτωση η λίστα ειναι κενή τότε «πετάει» μια εξαίρεση η οποία δείχνει ότι παρουσιάστηκε κάπιο σφάλμα κατά την εκτέλεση(δε βρέθηκαν κόμβοι).

6. **int getFirst(int item) throws NoSuchElementException**

Άν σε περίπτωση η λίστα ειναι κενή τότε «πετάει» μια εξαίρεση,αλλιώς επιστρέφει τα δεδομένα του πρώτου κόμβου.

7. **int getLast(int item) throws NoSuchElementException**

Άν σε περίπτωση η λίστα ειναι κενή τότε «πετάει» μια εξαίρεση,αλλιώς επιστρέφει τα δεδομένα του τελευταίου κόμβου.

8. **void printQueue (PrintStream stream)**

Ίδια λογική με τη μέθοδο της intQueueImpl.

9. **int size ()**

Ίδια λογική με τη μέθοδο της intQueueImpl.

Υλοποίηση της Άσκησης Β

1. Main:

Δημιουργούμε έναν πίνακα απο IntQueImpl[] με όνομα fifo. Κάθε στοιχείο του πίνακα αντιπροσωπεύει μια ουρά.

Ζητάμε απο τον χρήστη να εισάγει το input και το αποθηκεύουμε σε string. Ύστερα ελέγχουμε αν ο χρήστης έδωσε σωστό input, αν όχι τότε εμφανίζεται μήνυμα λάθους.

Ύστερα κάνουμε split στο string του Input και αποθηκεύουμε των πίνακα απο strings που προκύπτει με όνομα InputValues. Ύστερα παιρνοντάς ένα ένα τα στοιχεία του, τα μετατρέπουμε σε integers. Με τη χρήση μιας for ελέγχουμε πόσα ίδια στοιχεία υπάρχουν (πχ πόσα 0) και ύστερα εκτελούμε την removefromQueue ώστε να φύγουν απο τις ουρές όσοι πρέπει να φύγουν. Και ελέγχουμε αν ο χρόνος στον οποίο θα φύγουν αυτοί που προσθέτουμε στις ουρές είναι μεγαλύτερος απο το προηγούμενο Max και αν είναι τον θέτουμε ως max (για να βρούμε πότε φεύγει ο τελευταίος πελάτης). Με τη χρήση μιας int Μεταβλητής counter υπολογίζουμε πόσα άτομα υπάρχουν συνολικά.

2. AddtoQueue

Δέχεται ως ορίσματα 2 integers. Το πόσα άτομα έρχονται και σε ποια χρονική στιγμή. Αρχικοποιούμε μια μεταβλητή time=0

Με μια for εκτελούμε για κάθε άτομο που έρχεται εκείνη τη χρονική στιγμή: Αρχκοποιούμε ενα string s=”0”

Θέτουμε ως Min το μέγεθος της πρώτης ουράς και μέσα σε μια for εξακριβώνουμε το πόσες και ποιές ουρές έχουν το

Minimum αριθμό ατόμων. Αυτές δηλαδή που μας ενδιαφέρουμε για να προσθέσουμε άτομα. Και έτσι το string s διαμορφώνεται κρατώντας τις Min ουρες.(πχ s=”2 3”)

Τότε με την ίδια λογική με προηγουμένως, κάνουμε split και αποθηκεύουμε πίνακα st[] με τις τιμές των ουρών. Αν ο πίνακας έχει μόνο ένα στοιχείο προφανώς μόνο μια ουρά έχει το Min αλλιώς επιλέγουμε τυχαία απ αυτές τις ουρές.

Ύστερα αρχικοποιούμε μια μεταβλητή int LastPersonLeaves

Αν η ουρά είναι άδεια τότε LastPersonLeaves=when δηλαδή στην χρονική στιγμή που είμαστε.

Αλλιώς υπολογίζουμε το LastPersonLeaves ως(το μέγεθος της ουράς -1)*4 (γιατι χρειάζεται χρόνο 4 για να εξυπηρετηθεί ο κάθε πελάτης) +fifo[i].peek()

Ύστερα θέτουμε το time= LastPersonleaves +4

Το άθροισμά μου (ωστέ να βρώ ύστερα το average waiting time) = time-when

Κάνουμε fifo[i].put(time) Και κάνουμε return το time

3. RemoveFromQueue

Για κάθε μια απο τις ουρές ελέγχουμε αν δεν είναι άδεια και fifo[i].peek() <=time Δηλαδή αν η χρονική στιγμή που πρέπει να φύγει ο πρώτος εχεί ήδη περάσει ή είναι τώρα τον βγάζουμε απο την ουρά (fifo[i].get());

4. printInfo

Τελικά φτιάχνουμε μια μέθοδο η οποιά τυπώνει το Max δηλαδή πότε έφυγε και ο τελευτάιος πελάτης, υπολογίζουμε το average time Με μια απλή διαίρεση και το τυπώνουμε!