ΕΡΓΑΣΙΑ 1 – ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

➢ *ΜΕΡΟΣ Α – Υλοποίηση της διεπαφής `CharDoubleEndedQueue`*

- **Η διεπαφή αυτή περιέχει δηλώσεις μεθόδων για τον χειρισμό ουρών, στις οποίες μπορεί να γίνεται εισαγωγή και διαγραφή χαρακτήρων και από τα 2 άκρα της. Η κλάση `CharDoubleEndedQueueImpl` υλοποιεί αυτές τις μεθόδους.**
- **Εφόσον αυτή η διπλοουρά υλοποιείται με μια διπλά συνδεδεμένη λίστα χρειαζόμαστε αντικείμενα τύπου `Node` (εδώ `ListNode`) για τους κόμβους.**
- **Έτσι αρχικά υλοποιούμε μια κλάση `ListNode` η οποία παίρνει ως παράμετρο έναν χαρακτήρα `item` και αρχικά οι `previous` και `next` που είναι τύπου `ListNode` έχουν τιμή NULL. Επίσης αρχικοποιούμε την τιμή της `item` με την εντολή `this.item = item;`**
- **Τώρα στην κλάση `CharDoubleEndedQueueImpl` και συγκεκριμένα στον κατασκευαστή, οι `header` και `tail` που είναι τύπου `ListNode` έχουν την τιμή NULL εφόσον δεν έχει δημιουργηθεί ένας κόμβος. Επίσης η ακέραια μεταβλητή `size` που δείχνει το μέγεθος της διπλοουράς (ή αλλιώς τον αριθμό των στοιχείων της διπλοουράς) παίρνει την τιμή 0.**
- **Η μέθοδος `isEmpty()` ελέγχει αν η διπλοουρά είναι κενή. Ο έλεγχος γίνεται πολύ απλά με την `return size == 0;`**
- **Η μέθοδος `addFirst(item)` προσθέτει έναν χαρακτήρα στον μπροστινό άκρο της διπλοουράς. Φτιάχνουμε αρχικά ένα προσωρινό αντικείμενο τύπου `ListNode` ,το `tempNode` που αποθηκεύει την τιμή του χαρακτήρα. Αρχικά γίνεται έλεγχος του αν η κεφαλή είναι NULL. Αν είναι τότε ο επόμενος κόμβος του `tempNode` γίνεται ίσος με την κεφαλή και ο προηγούμενος κόμβος της κεφαλής γίνεται ίσος με τον `tempNode`. Αν δεν συμβαίνει αυτό τότε η ουρά (`tail`) γίνεται ίση με την `tempNode`. Μετά τον έλεγχο η κεφαλή γίνεται ίση με την `tempNode`. Αυξάνουμε κατά 1 την τιμή της `size` εφόσον προστέθηκε ένας χαρακτήρας στην ουρά. Έτσι δημιουργήθηκε ένας νέος κόμβος με την τιμή που θέλαμε να προσθέσουμε και θα βρίσκεται στην κεφαλή της διπλοουράς.**
- **Η μέθοδος `removeFirst()` εξάγει έναν χαρακτήρα από το μπροστινό άκρο της διπλοουράς. Πρέπει αρχικά να ελέγξουμε ότι η ουρά δεν είναι άδεια, ώστε να μπορέσουμε να εξάγουμε τουλάχιστον ένα στοιχείο. Αν είναι άδεια θα πετάξει εξαίρεση τύπου `NoSuchElementException` με το μήνυμα ότι η ουρά είναι άδεια. Εφόσον δεν είναι άδεια, δημιουργούμε έναν προσωρινό κόμβο που δείχνει στην κεφαλή και αποθηκεύουμε σε μια μεταβλητή τύπου `char` το περιεχόμενο της. Η κεφαλή θα δείχνει τώρα στον επόμενο κόμβο του προσωρινού (`tempNode.next`). Γίνεται στην συνέχεια έλεγχος του αν η κεφαλή είναι NULL. Αν είναι τότε ο κόμβος πριν από την κεφαλή γίνεται NULL αλλιώς η ουρά (`tail`) γίνεται NULL. Τελικά το στοιχείο που υπήρχε στην `tempNode` καθώς και ο ίδιος ο κόμβος διαγράφονται. Μειώνουμε κατά 1 την τιμή της μεταβλητής `size` εφόσον εξάγαμε έναν χαρακτήρα από την διπλοουρά και η μέθοδος επιστρέφει τον χαρακτήρα που διαγράφηκε.**
- **Η μέθοδος `addLast(item)` προσθέτει έναν χαρακτήρα πίσω στην διπλοουρά. Η λειτουργία της είναι ανάλογη της `addFirst` μόνο που στην θέση της κεφαλής μπαίνει η ουρά (`tail`). Ο προηγούμενος κόμβος της `tempNode` δείχνει στην ουρά (`tail`) και ο επόμενος κόμβος της ουράς (`tail`) δείχνει στον `tempNode`. Αυτό γίνεται αν η ουρά δεν είναι NULL. Αν είναι τότε η κεφαλή γίνεται ίση με την `tempNode`.**
- **Η μέθοδος `removeLast()` εξάγει έναν χαρακτήρα από το τέλος της ουράς. Η λειτουργία της είναι αντίστοιχη της `removeFirst()` μόνο που αντί για την κεφαλή τον ρόλο τον παίζει τώρα η ουρά (`tail`). Αν η ουρά (`tail`) δεν είναι NULL τότε ο επόμενος κόμβος αυτής θα γίνει NULL αλλιώς η κεφαλή θα γίνει NULL. Τελικά θα επιστρέψει τον χαρακτήρα που βγήκε από το τέλος της διπλοουράς**
- **Η μέθοδος `getFirst()` απλά επιστρέφει τον χαρακτήρα που βρίσκεται στην κεφαλή**
- **Η μέθοδος `getLast()` απλά επιστρέφει τον χαρακτήρα που βρίκσεται στην ουρά (`tail`)**
- **Η μέθοδος `printQueue(stream)` θα τυπώσει τα στοιχεία της διπλοουράς ξεκινώντας από την κεφαλή. Δημιουργούμε μια κεφαλή-μετρητή και όσο δεν φτάνουμε ακόμα στο τέλος της**

διπλοουράς, θα τυπώνει τον χαρακτήρα που βρίσκεται στην κεφαλή. Φυσικά μετακινούμε και την κεφαλή στην επόμενη (μετακινούμαστε δηλαδή από κόμβο σε κόμβο). Η παράμετρος της `printStream()` δείχνει το stream στον οποίο θα τυπωθούν τα στοιχεία. Εδώ θέλουμε την `System.out`.

- **Η μέθοδος `size()` επιστρέφει το μέγεθος της διπλοουράς ή αλλιώς τον αριθμό τωνστοιχείων που βρίσκονται στην διπλοουρά.**

➢ *ΜΕΡΟΣ Β – Υλοποίηση του προγράμματος-πελάτη `DNAPalindrome`*

- **Αυτό το πρόγραμμα-πελάτης ζητάει από τον χρήστη να εισάγει μια ακολουθία DNA και να ελέγξει αν είναι συμπληρωματικά παλινδρομική κατά Watson-Crick. Αυτό σημαίνει ότι όταν αντικαταστήσουμε κάθε νουκλεοτίδιο με το συμπλήρωμά του και μετά αντιστρέψουμε την σειρά, τότε παίρνουμε την ίδια ακολουθία με την αρχική.**
- **Στην μέθοδο `main()` αρχικά δημιουργούμε μια διπλοουρά (αντικείμενο `CharDoubleEndedQueueImpl`)**
- **Στην συνέχεια φτιάχνουμε ένα αντικείμενο τύπου `Scanner` που θα μας βοηθήσει να αποθηκεύσουμε την ακολουθία που θα δώσει ο χρήστης στην μεταβλητή `DNA_Sequence` που είναι τύπου `String`.**
- **Η μέθοδος `validDNASequence(sequence)` που καλείται , ελέγχει αν η ακολουθία που δώσαμε είναι έγκυρη ή όχι. Για να ελέγξουμε την εγκυρότητα ελέγχουμε αν η ακολουθία έχει MONO τους χαρακτήρες A,T,C,G ή αν ισούται με το κενό string. Αν είναι έγκυρη επιστρέφει true αν όχι false και πιο κάτω στο πρόγραμμα εμφανίζεται το κατάλληλο μήνυμα λάθους.**
- **Εφόσον δώσαμε έγκυρη ακολουθία χρησιμοποιούμε μια for-loop για να βάλουμε κάθε χαρακτήρα της ακολουθίας (με την χρήση της `charAt(letter)`) στο τέλος της διπλοουράς (Επιλέξαμε αυτό τον τρόπο ώστε να μπει η ακολουθία έτσι όπως είναι).**
- **Στην συνέχεια καλούμε την μέθοδο `complementDNA(deque)` η οποία παίρνει ως όρισμα την διπλοουρά και σκοπός της είναι να φτιάξει το συμπληρωματικό της ακολουθίας που δόθηκε από την χρήστη. Αρχικά στην μέθοδο αρχικοποιούμε έναν μετρητή με την τιμή 0 και όσο αυτός ο μετρήτης είναι μικρότερος του μεγέθους της διπλοουράς (δηλάδη μικρότερος από τα στοιχεία της διπλοουράς) θα κάνει την εξής λειτουργία: Αν το πρώτο στοιχείο της διπλοουράς είναι το A (T,C,G ή το κενό), η λειτουργία της οποίας γίνεται εφικτή με την χρήση της μεθόδου `getFirst()`, τότε θα εξάγει το στοιχείο αυτό με την χρήση της `removeFirst()` και θα προσθέσει το συμπληρωματικό του, δηλαδή το T(A,G,C και το κενό) στο τέλος της ουράς. Ειδικά για το κενό, το συμπληρωματικό του θα είναι φυσικά το κενό, αλλά MONO το κενό της αρχής θεωρείται ότι είναι συμπληρωματικά παλιδρομική ακολουθία κατά Watson-Crick. Αυξάνουμε τον μετρητή κατά 1 και συνεχίζει η επανάληψη, Μόλις τελειώσει η διπλοουρά θα περιέχει το συμπληρωματικό της ακολουθίας που έδωσε ο χρήστης και αυτό γραμμένο στην σειρά.**
- **Όταν τελειώσει η παραπάνω λειτουργία, ελέγχουμε αν τελικά η ακολουθία DNA είναι συμπληρωματικά παλινδρομική κατά Watson-Crick με την χρήση της μεθόδου `isDNAPalindrome(deque, initial_DNASequence)`. Το όρισμα `deque` περιέχει το συμπληρωματικό της αρχικής ακολουθίας DNA που έδωσε ο χρήστης (`initial_DNASequence`). Αρχικοποιούμε την τιμή 0 σε ένα ακόμα μετρητή. Με την χρήση μιας for-loop, ελέγχουμε αν κάθε χαρακτήρας της αρχικής ακολουθίας είναι ίσος με τον χαρακτήρα που βρίσκεται στο τέλος της ουράς. Δηλαδή αν ο πρώτος χαρακτήρας της `initial_DNASequence` είναι ίσος με τον τελευταίο χαρακτήρα της διπλοουράς, στην συνέχεια ο δεύτερος με τον προτελευταίο και πάει λέγοντας. Κάθε φορά που είναι ισχύει η ισότητα εξάγουμε από το τέλος της διπλοουράς το στοιχείο και αυξάνουμε τον μετρητή κατά 1. Με το πέρας της επανάληψης ελέγχουμε αν ο μετρητής είναι ίσος με το μήκος της αρχικής ακολουθίας. Αν ναι, τότε επιστρέφει true , δηλαδή είναι συμπληρωματικά παλινδρομική κατά Watson-Crick και στο κυρίως πρόγραμμα τυπώνεται το αντίστοιχο μήνυμα. Αν όχι, τότε επιστρέφει false, δηλαδή δεν είναι συμπληρωματικά παλινδρομική κατά Watson-Crick και στο κυρίως πρόγραμμα τυπώνεται το αντίστοιχο μήνυμα.**

➢ *ΜΕΡΟΣ Γ – Υλοποίηση της διεπαφής CharQueue*

- **Η διεπαφή αυτή περιέχει δηλώσεις μεθόδων για τον χειρισμό μιας ουράς FIFO με την χρήση μιας λίστας μονής σύνδεσης.**
- **Η υλοποίηση της διεπαφής αυτής γίνεται με την κλάση `CharQueueImpl`.**
- **Θα κάνουμε ξανά χρήση της κλάσης `ListNode` μόνο που θα την προσαρμόσουμε για ουρά που υλοποιείται από απλή συνδεδεμένη λίστα.**
- **Αρχικά βέβαια οι 3 μεταβλητές (`header`,`tail`,`size`) και ο κατασκευαστής είναι παρόμοιοι με αυτοί της κλάσης `CharDoubleEndedQueueImpl`. Οι υπόλοιπες μέθοδοι θα καθορίσουν την λειτουργία της απλής συνδεδεμένης λίστας.**
- **Οι μέθοδοι `isEmpty()`, `size()`, `printQueue(stream)` έχουν επίσης παρόμοιες λειτουργίες με αυτή της προαναφερθείσας κλάσης. (Στην `printQueue(stream)` χρησιμοποιήσαμε μια for-loop αλλά δεν έχει διαφορά από την while-loop)**
- **Η μέθοδος `put(item)` προσθέτει έναν χαρακτήρα μέσα στην ουρά FIFO. Αρχικά δημιουργείται ένα αντικείμενο `ListNode` που δείχνει στην ουρά (tail) της ουράς FIFO και περιέχει το στοιχείο που θέλουμε να εισάγουμε. Γίνεται έλεγχος του αν η ουρά FIFO είναι άδεια. Αν είναι τότε η κεφαλή είναι ίση με την ουρά. Δηλαδή η κεφαλή είναι ταυτόχρονα και ουρά. Αν όχι τότε ο επόμενος κόμβος μετά την `tempNode` που ορίσαμε γίνεται ίσος με την ουρά (δείχνει στο `tail`). Αυξάνουμε τον μετρητή `size` κατά 1 εφόσον εισάγαμε νέο χαρακτήρα.**
- **Η μέθοδος `get()` εξάγει τον παλαιότερο χαρακτήρα που μπήκε στην ουρά FIFO. Πρέπει να ελέγξουμε αρχικά αν η ουρά FIFO είναι άδεια, γιατί η μέθοδος προϋποθέτει ότι υπάρχει ένα τουλάχιστον στοιχείο προς εξαγωγή. Αν είναι άδεια τότε θα πετάξει εξαίρεση τύπου `NoSuchElementException` με το αντίστοιχο μήνυμα. Εφόσον δεν είναι, αποθηκεύουμε σε μια μεταβλητή τύπου char το αντικείμενο που βρίσκεται στην κεφαλή (γιατί εκεί βρίσκεται το παλαιότερο στοιχείο). Στη συνέχεια, αν η κεφαλή ταυτίζεται με την ουρά τότε και τα 2 παίρνουν την τιμή NULL αλλιώς η κεφαλή θα δείχνει στο επόμενο κόμβο της. Μειώνουμε την `size` κατά 1 εφόσον εξάγαμε στοιχείο και τέλος η μέθοδος επιστρέφει τον χαρακτήρα που διαγράφηκε.**
- **Η μέθοδος `peek()` επιστρέφει απλά τον παλαιότερο χαρακτήρα της ουράς FIFO. Και εδώ πρέπει να ελέγξουμε αν η ουρά FIFO είναι άδεια. Αν είναι τότε πετάει εξαίρεση τύπου `NoSuchElementException`. Αν όχι τότε επιστρέφει το περιεχόμενο της κεφαλής.**
- **ΠΡΟΑΙΡΕΤΙΚΑ: Στις μεθόδους `put(item)` και `get()` προσθέσαμε μια εντολή `System.out.println()` όπου τυπώνεται ο χαρακτήρας που εισήχθη και εξήχθη αντίστοιχα. Έγινε απλά για λόγους καλύτερης διαδραστικότητας με τον χρήστη-πελάτη (user-friendly).**

➢ ***ΜΕΡΟΣ Δ – Υλοποίηση της διεπαφής `CharQueueWithMin`***

- **Η διεπαφή αυτή υλοποιεί μια μέθοδο την `min()`, η οποία επιστρέφει τον λεξικογραφικά μικρότερο χαρακτήρα στην ουρά FIFO και κληρονομεί την διεπαφή `CharQueue`.**
- **Η κλάση `CharQueueWithMinImpl` κληρονομεί την κλαση `CharQueueImpl` εφόσον έχουν παρόμοιες λειτουργίες καθώς και υλοποιεί την παραπάνω διεπαφή. Άρα αντί να δημιουργήσουμε νέο αντικείμενο `CharQueueImpl` θα χρησιμοποιούμε την `super()` και την μέθοδο που περιέχει. (π.χ. `super.put(item)`).**
- **Ωστόσο, οι μέθοδοι `put(item)`, `get()` και `printQueue(stream)` έχουν μια διαφορετική υλοποίηση οπότε θα εστιάσουμε σε αυτές (η 3η είναι προαιρετική).**
- **Η ιδέα μας είναι ότι η ουρά FIFO θα είναι η βασική μας δομή στην οποία θα εισαχθούν και εξαχθούν οι χαρακτήρες, ενώ η διπλοουρά θα κρατάει ΜΟΝΟ το λεξικογραφικά μικρότερο χαρακτήρα της ουράς FIFO!**
- **Αρχικά δημιουργούμε ένα αντικείμενο τύπου `CharDoubleEndedQueueImpl`, για να έχουμε στην διάθεση μας την διπλοουρά. Επίσης , εκμεταλλευόμενοι την κληρονομικότητα, βάζουμε απλά ένα `super()` στον κατασκευαστή της κλάσης `ChaeQueueWithMinImpl`!**
- **Η μέθοδος `put(item)` έχει σκοπό να εισάγει έναν χαρακτήρα και στις δύο ουρές. Αρχικά ελέγχουμε αν η διπλοουρά είναι κενή και αν είναι τότε εισάγουμε τον χαρακτήρα και στις δύο ουρές. Στην ουρά FIFO χρησιμοποιούμε την `super.put(item)` και στην διπλοουρά την `deque.addFirst(item)` / `deque.addLast(item)`. Αν η διπλοουρά δεν είναι κενή τότε γίνεται το εξής:**
    - **Ο χαρακτήρας τοποθετείται κανονικά στην ουρά FIFO με την `super.put(item)`.**
    - **Ωστόσο στην διπλοουρά ενδέχεται να χρειαστεί διαγραφή στοιχείων (λόγω της λειτουργίας που θέλουμε να έχει η διπλοουρά). Ελέγχουμε αν ο χαρακτήρας που εισάγουμε είναι λεξικογραφικά μικρότερος από αυτόν που βρίσκεται στο τέλος της διπλοουράς και αν ισχύει τότε εξάγουμε αυτό το στοιχείο από το τέλος της ουράς με την χρήσης της `deque.getLast()` και προσθέτουμε αυτό το στοιχείο με την χρήση της `deque.addFirst(item)`. (Μπορούμε να χρησιμοποιήσουμε και τις `getFirst()` `/removeFirst()/` `addLast(item)` εφόσον έχουμε (και θέλουμε) 1 μόνο στοιχείο στην διπλοουρά!).**

- **Η μέθοδος `get()` έχει σκοπό να εξάγει έναν χαρακτήρα και από τις δύο ουρές. Προφανώς πρέπει να χρησιμοποιήσουμε την μέθοδο `isEmpty()` για να ελέγξουμε ότι οι ουρές δεν είναι άδειες. Αν είναι τότε πετάει εξαίρεση τύπου `NoSuchElementException`. Μετά τον έλεγχο χρησιμοποιούμε την `super.get()` για να να αποθηκεύσουμε τον χαρακτήρα που θα εξαχθεί από την ουρά FIFO σε μια μεταβλητή τύπου `char`. Με την μεταβλητή αυτή ελέγχουμε αν αυτός ισούται με τον χαρακτήρα που βρίσκεται στην διπλοουρά με την χρήση της `deque.getFirst()` / `deque.getLast()`.**

Αν ισχύει τότε τον εξάγουμε από την διπλοουρά με την χρήση της `deque.removeFirst() / deque.removeLast()`. Έπειτα δημιουργούμε ένα αντικείμενο `ListNode` με όνομα node που δείχνει στην κεφαλή της ουράς FIFO και προσθέτουμε σε αυτή τον επιθυμητό χαρακτήρα. Όσο δεν φτάνουμε στο τέλος της ουράς FIFO αν ο χαρακτήρας που κρατιέται από την node είναι λεξικογραφικά μικρότερος από τον χαρακτήρα που βρίσκεται στην διπλοουρά (`deque.getFirst() / deque.getLast()`) τότε το εξάγουμε από την διπλοουρά με χρήση της `deque.removeFirst() / deque.removeLast()` και εισάγουμε τον επιθυμητό χαρακτήρα στην διπλοουρά με χρήση της `deque.addFirst(node.item) / deque.addLast(node.item)`. Μετακινούμε και τον κόμβο με χρήση της `node.next`. Τέλος, επιστρέφεται ο χαρακτήρας που εξήχθη από τις ουρές.

- **Η υλοποίηση της μεθόδου `printQueue(stream)` είναι ΠΡΟΑΙΡΕΤΙΚΗ. Ουσιαστικά καλεί την μέθοδο `printQueue(stream)` των δύο ουρών αλλά τα αποτέλεσματα θα βγουν πιο ομοιόμορφα.**
- **Η μέθοδος `min()` τελικά επιστρέφει το πρώτο στοιχείο (το μοναδικό) που είναι στην διπλοουρά με χρήση της `deque.getFirst() / deque.getLast()`.**
- **Ακολουθεί ένα παράδειγμα του πως θα παρουσιάζει τα στοιχεία των ουρών η `main()`:**

Putting the character: K
QUEUE: K
DEQUE: K

Putting the character: E
QUEUE: KE
DEQUE: E

Putting the character: R
QUEUE: KER
DEQUE: E

Removing the character: K
QUEUE: ER
DEQUE: E

Putting the character: E
QUEUE: R
DEQUE: R

…
…
…

Minimum character: R