ΑΝΑΦΟΡΑ ΓΙΑ ΤΗΝ ΠΡΩΤΗ ΕΡΓΑΣΙΑ ΣΤΙΣ ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

ΧΡΙΣΤΙΝΑ ΑΙΜΙΛΙΑ ΦΛΑΣΚΗ

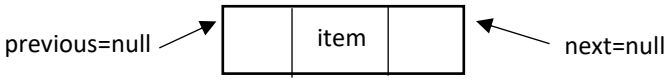
Όλη η εργασία γράφτηκε μέσω του IDE IntelliJ!

Α' ΜΕΡΟΣ:

Η υλοποίηση της διεπαφής στο Α μέρος της εργασίας:

isEmpty(): Ελέγχουμε αν η κεφαλή(head) της λίστας είναι άδεια, αν ναι τότε η λίστα είναι άδεια και η συνάρτηση επιστρέφει true.

addFirst(String item) : Δημιουργείται αντικείμενο της κλάσης Node*, τσεκάρουμε αν η κεφαλή είναι άδεια, αν ναι τότε βάζουμε το αντικείμενο που δημιουργήσαμε πριν μέσα στην λίστα ως κεφαλή αλλά και ως ουρά(tail) αφού είναι το μοναδικό αντικείμενο της λίστας.



Αν όχι τότε το προσθέτει στην αρχή της λίστας ως την κεφαλή, αδειάζοντας το previous και βάζοντας στο next την προηγούμενη κεφαλή. Επίσης με την είσοδο σε αυτή την συνάρτηση αυξάνουμε το μέγεθος του size(μεταβλητή) που αντιπροσωπεύει το μέγεθος της λίστας μας.

Αντίστοιχη υλοποίηση έχει η **addLast(String item)**.

removeFirst(): Τσεκάρει αν η λίστα είναι άδεια(αν είναι κάνει throw NoSuchElementException) και αν δεν είναι ελέγχει αν έχει μόνο ένα αντικείμενο. Σε αυτή την περίπτωση αδειάζει την λίστα, στην περίπτωση που έχει παραπάνω από ένα, χρησιμοποιώντας την συνάρτηση **getItem** της κλάσης Node κρατάει σε καινούριο αντικείμενο το περιεχόμενη της κεφαλής και αντικαθιστούμε την κεφαλή με αυτό που είχε η προηγούμενη στο next με την συνάρτηση **getNext**. Επίσης μειώνουμε το μέγεθος της size.

Αντίστοιχη υλοποίηση έχει η συνάρτηση **removeLast()**.

Οι μέθοδοι **getFirst(), getLast()** μας επιστρέφουν την κεφαλή και την ουρά της λίστας αντίστοιχα, σε περίπτωση άδειας λίστας έχουμε throw NoSuchElementException.

printQueue(PrintStream stream): Εκτυπώνει ένα ένα τα στοιχεία της λίστας μέσω μιας while loop, αν η λίστα είναι άδεια τότε έχουμε πάλι throw NoSuchElementException.

Η υλοποίηση και των λειτουργιών εξαγωγής/εισαγωγής αλλά και της μέθοδού size(που επιστρέφει το μέγεθος της λίστας) είναι σε χρόνο O(1). Αυτό συμβαίνει γιατί δεν χρησιμοποιήθηκε καμία επανάληψη για την υλοποίηση των παραπάνω μεθόδων και η λειτουργία δεν επηρεάζεται από το μέγεθος της εισόδου.

*Η κλάση Node περιέχει μεθόδους που μας επιστρέφουν αυτά που χρειαζόμαστε από τα nodes της λίστας.

Β' ΜΕΡΟΣ:

Οι μέθοδοι **isOperator(char c), isInt(char c)** ελέγχουν αν η παράσταση postfix που μας δόθηκε αποτελείται μόνο από τα σύμβολα +,-,*,/,^ και από μονοψήφιους δεκαδικούς αριθμούς.

Η μέθοδος **convert(String postfix)** είναι η μέθοδος μέσα στην οποία πραγματοποιείται η μετατροπή από Postfix σε infix.

Αυτό συμβαίνει με την βοήθεια της υλοποίησης από το μέρος α.

Ένα παράδειγμα:

Μας εμφανίζεται το μήνυμα Postfix: οπού γράφουμε την παράσταση που θέλουμε να μετατρέψουμε, έστω ότι δίνουμε την παράσταση 12+

1. Ξεκινάει να τρέχει την convert.
2. Δημιουργείται ένα καινούριο αντικείμενο της κλάσης **StringDoubleEndedQueueImpl**.
3. Μπαίνει σε ένα for loop που θα κάνει αριθμό επαναλήψεων ίσο με το μέγεθος της παραστάσεις που δόθηκε από τον χρήστη.
4. Παίρνει το character όταν βρίσκεται στην αντίστοιχη θέση i.
5. Το μετατρέπει σε String.
6. Καλεί την **isOperator** και ελέγχει αν ο χαρακτήρας που δόθηκε ανήκει στα σύμβολα που ζητήθηκαν, αν όχι (στο παράδειγμά μας στην πρώτη επανάληψη έχουμε το 1) ελέγχει αν είναι αριθμός με την **isInt()** αλλιώς εμφανίζει το μήνυμα error! στην οθόνη και σταματάει την επανάληψη. Στην περίπτωση μας μπαίνει στο δεύτερη περίπτωση του if επειδή είναι αριθμός τον προσθέτει στο τέλος της λίστας με την μέθοδο **addLast(s)** από το πρώτο μέρος της εργασίας.

null	1	null

7. Το ίδιο συμβαίνει και με την δεύτερη επανάληψη στο παράδειγμά μας.

null	1	2
1	2	null

8. Στην Τρίτη επανάληψη μπαίνει μέσα στο πρώτη περίπτωση του if τότε δημιουργεί δυο νέες String μεταβλητές στις οποίες βάζει τους δυο πρώτους χαρακτήρες, χρησιμοποιώντας την μέθοδο **getFirst()** από το μέρος α και μετά από την κάθε καταχώρηση σβήνει τα πρώτα nodes της λίστας με την **removeFirst()**

9. Βάζει στην κεφαλή της λίστας το a και b αντικείμενα ανάμεσα από () και με ανάμεσα τους το String s που αντιπροσωπεύει τον operator με την **addFirst()**

null	(1+2)	null

10. Μετατρέπει σε String το περιεχόμενο της κεφαλής της λίστας και το επιστρέφει (δηλαδή το (1+2)).
11. Εκτυπώνει μήνυμα στην οθόνη με το Infix: και το τελικό αποτέλεσμα.

```
Postfix: 12+
Infix: (1+2)

Process finished with exit code 0
```

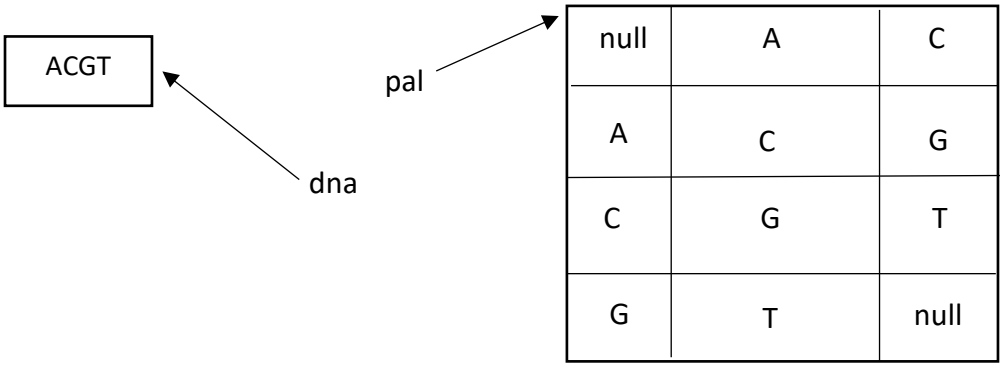
Γ' ΜΕΡΟΣ:

Η μέθοδος **isDNA(char c)** ελέγχει αν ο χαρακτήρας που δίνεται ανήκει στα 4 κεφαλαία γράμματα από τα οποία αποτελείται ένα DNA.

Θα εκτελέσουμε πάλι ένα παράδειγμα βήμα βήμα για να δούμε πως λειτουργεί το πρόγραμμά μας.

Εμφανίζεται το μήνυμα Give DNA: και έστω ότι δίνουμε την συμβολοσειρά ACGT

1. Δημιουργείται ένα καινούριο αντικείμενο της κλάσης **StringDoubleEndedQueueImpl** με το όνομα **pal.**
2. Αρχικοποιείται η Boolean b1 μεταβλητή με false περιεχόμενο, η συγκεκριμένη μεταβλητή μας βοηθάει όταν θα θέλουμε να εμφανίσουμε το τελικό μήνυμα στον χρήστη, σε περίπτωση που έχει δώσει έστω και ένα λανθασμένο γράμμα να μην μπει στην διαδικασία να τσεκάρει αν είναι παλινδρομικό ή όχι.
3. Μπαίνει σε ένα for loop που θα κάνει αριθμό επαναλήψεων ίσο με το μέγεθος του DNA που δόθηκε από τον χρήστη.
4. Παίρνει το character όταν βρίσκεται στην αντίστοιχη θέση i.
5. Το μετατρέπει σε String.
6. Μπαίνει στην πρώτη περίπτωση του if που ο χαρακτήρας ανήκει σε αυτούς του DNA, κάθε φορά που μπαίνει σε αυτή την περίπτωση το b1 αλλάζει σε true.
7. Αν ο χαρακτήρα είναι το A τότε βάζει στην κεφαλή της λίστας T με την **addFIrst** από το πρώτο μέρος της εργασίας. Το ίδιο θα κάνει αντίστοιχα για το T βάζει το A, για G βάζει C και για C βάζει το G. Στην δικιά μας περίπτωση θα βάζει στην κεφαλή της λίστας το T.
8. Σε περίπτωση που ο χαρακτήρας δεν ανήκε σε αυτούς του DNA θα εμφανιστεί το μήνυμα error! στην οθόνη και θα σταματήσει η επανάληψη.
9. Στην περίπτωση μας το βήμα 7 θα πραγματοποιθείς άλλες 3 φορές και θα αντικαταστήσει αντίστοιχα τους χαρακτήρες που δώσαμε. Η λίστα pal θα περιέχει μετρά την εισαγωγή των 4 αντικειμένων θα περιέχει A,C,G,T. Αυτό προκύπτει επειδή προσθέτουμε τα αντικείμενα στην αρχή της λίστας με αποτέλεσμα κάθε φορά που εισχωρούμε ένα καινούριο αντικείμενο το προηγούμενο να βρίσκεται στη δεύτερη θέση με το πρώτο αντικείμενο που εισχωρήσαμε να βρίσκεται στην τελευταία και το τελευταίο στην πρώτη.



10. Ορίζουμε ένα String(s0) με περιεχόμενο το κενό και ένα αντικείμενο της Node το current που περιέχει την κεφαλή της pal.
11. Ξεκινάει ένα while loop το οποίο θα τερματίσει όταν το current δείχνει σε άδειο node δηλαδή στο τέλος της λίστας που το τελευταίο node έχει για next = null.
12. Σε κάθε επανάληψη με την **getItem** της κλάσης Node παίρνουμε το περιεχόμενο της μεταβλητής current και το προσθέτουμε στο s0 και δίνουμε για περιεχόμενο του current το περιεχόμενο του επόμενου node.

13. Ελέγχουμε αν το b1 είναι ίσο με true αν ναι συγκρίνουμε το s0 με την συμβολοσειρά dna(η εκχώρηση του χρήστη) αν είναι παλινδρομικό εμφανίζεται το μήνυμα Is palindrome! αν όχι εμφανίζεται το μήνυμα Is not palindrome!

```
Give DNA:
ACGT
Is palindrome!

Process finished with exit code 0
```