ΑΝΑΦΟΡΑ ΠΑΡΑΔΟΣΗΣ

Όλες οι κλάσεις γράφτηκαν και δοκιμάστηκαν στο IDEA IntelliJ. Τα δεδομένα εισόδου τα δίνουμε χρησιμοποιώντας αντικείμενα τύπου Scanner τα οποία κάναμε import από το java.util.

a. Αρχικά, την εργασία την κάναμε με Generics.
   Θεωρήσαμε ως πρώτο στοιχείο της ουράς το tail και η υλοποίησή μας έγινε με βάση αυτή την υπόθεση. Ορίσαμε δύο αναφορές Node για το τέλος (head) και την αρχή (tail) της ουράς που δείχνουν σε null.
   Για την isEmpty σκεφτήκαμε ότι για να είναι κενή η ουρά πρέπει είτε το πρώτο είτε το τελευταίο στοιχείο της να είναι κενό.
   Για την addFirst δημιουργήσαμε ένα αντικείμενο τύπου Node που παίρνει ως όρισμα το στοιχείο που θέλουμε να προσθέσουμε στην πρώτη θέση της ουράς. Κάναμε έλεγχο σε περίπτωση που η ουρά είναι κενή (με τη χρήση της isEmpty) όπου και θέσαμε το πρώτο και το τελευταίο στοιχείο να είναι ίσα με το στοιχείο που θέλουμε να προσθέσουμε. Σε διαφορετική περίπτωση, ορίσαμε το επόμενο του στοιχείου που θα προστεθεί στην ουρά να είναι ίσο με το (αρχικό) πρώτο στοιχείο, βάλαμε το νέο στοιχείο μία θέση πριν το (αρχικό) πρώτο στοιχείο και ορίσαμε το νεό στοιχείο να είναι το tail.
   Για την removeFirst ελέγξαμε αν η ουρά είναι κενή και κάναμε throw το σχετικό exception αλλιώς ορίσαμε το στοιχείο που θα αφαιρεθεί να είναι το πρώτο στοιχείο της ουράς. Έπειτα, ελέγξαμε αν η ουρά έχει ένα μόνο στοιχείο ώστε να αφαιρεθεί το πρώτο στοιχείο και πλέον το τέλος και η αρχή να δείχνουν null. Σε διαφορετική περίπτωση, ορίσαμε ως tail το στοιχείο που βρισκόταν στη δεύτερη θέση της ουράς. Τέλος, επιστρέψαμε το στοιχείο που διαγράφτηκε.
   Για την addLast φτιάξαμε ένα αντικείμενο τύπου Node με όρισμα το στοιχείο που θα προστεθεί στο τέλος της ουράς. Ελέγξαμε αν η ουρά είναι κενή ώστε να θέσουμε την αρχή και το τέλος ίσα με το στοιχείο που θα προστεθεί. Διαφορετικά, ορίσαμε το στοιχείο που θα προστεθεί να είναι μετά το αρχικό head, άρα να παίρνει αυτό τον τίτλο του.
   Για την removeLast ελέγξαμε αν η ουρά είναι κενή, ώστε να κάνει throw το σχετικό exception, αλλιώς ορίσαμε το στοιχείο που θα αφαιρεθεί να είναι το τελευταίο της ουράς. Έπειτα, ελέγξαμε αν η ουρά έχει ένα στοιχείο ώστε να δείχνει η αρχή και το τέλος σε null, διαφορετικά διαγράψαμε το τελευταίο στοιχείο της ουράς.
   Για τις getFirst και getLast ελέγξαμε αν η ουρά είναι κενή ώστε να εμφανίζεται μήνυμα λάθους, αλλιώς επιστρέψαμε το πρώτο και τελευταίο στοιχείο αντίστοιχα.
   Για την printQueue ελέγξαμε αν η ουρά είναι κενή ώστε να κάνει throw το σχετικό exception, αλλιώς δημιουργήσαμε ένα βοηθητικό αντικείμενο τύπου Node για να αντιγράψουμε το tail εκεί και να μην χαθεί η πραγματική του τιμή. Έπειτα τρέξαμε την ουρά και τυπώναμε το κάθε στοιχείο της πηγαίνοντας το βοηθητικό Node μία θέση δεξιά κάθε φορά μέχρι να φτάσει στο head.

Για την size είχαμε έναν μετρητή που ανάλογα με τη μέθοδο που κάναμε πχ addFirst/addLast ή removeFirst/removeLast τον αυξάναμε ή τον μειώναμε κατά ένα αντίστοιχα.

Οι λειτουργίες εισαγωγής/εξαγωγής είναι σε O(1), γιατί χρησιμοποιούμε πολύ απλές εντολές (αναθέσεις τιμών, έλεγχοι συνθηκών και μία επανάληψη με 2 εντολές) .
Η μέθοδος size είναι σε O(1), γιατί αντί να χρησιμοποιήσουμε ένα loop το οποίο θα κάνει προσπέλαση σε κάθε κόμβο της ουράς χρησιμοποιούμε έναν μετρητή που μεταβάλλεται μετά από κάθε εισαγωγή ή εξαγωγή από την ουρά.

Για να δούμε αν η υλοποίηση της διεπαφής τρέχει, φτιάξαμε μία main όπου δημιουργήσαμε ένα αντικείμενο τύπου StringDoubleEndedQueueImpl και δοκιμάσαμε κάποιες από τις μεθόδους που φτιάξαμε.

b. Στην PostfixToInfix αφού διαβάσαμε από τον χρήστη την αριθμητική παράσταση σε τύπο String ελέγξαμε αν έδωσε σωστή ή λάθος μορφή και εμφανίσαμε μήνυμα λάθους για τη δεύτερη περίπτωση.

ΧΡΗΣΗ Α' ΜΕΡΟΥΣ
Για την πρώτη περίπτωση όπου το input είναι σωστό, δημιουργήσαμε ένα αντικείμενο τύπου StringDoubleEndedQueueImpl και διανύσαμε το κάθε στοιχείο της παράστασης.
Μετατρέψαμε το κάθε στοιχείο σε χαρακτήρα και ελέγξαμε τον κάθε χαρακτήρα (στην ίδια επανάληψη). Στην περίπτωση που ο χαρακτήρας είναι αριθμός, τον προσθέτουμε στην τελευταία θέση της ουράς με τη χρήση της μεθόδου addLast του Α' μέρους (μετατρέποντας τον χαρακτήρα σε String).
Αν ο χαρακτήρας είναι τελεστής, σε μια προσωρινή μεταβλητή αποθηκεύσαμε τα δύο τελευταία στοιχεία της ουράς που έχουμε μέχρι στιγμής και το διαγράψαμε από αυτήν χρησιμοποιώντας τη μέθοδο removeLast του Α' μέρους δύο φορές αντίστοιχα.
Έπειτα, προσθέσαμε στην ουρά (με την addLast) ένα ενιαίο String το οποίο αποτελείται από μία αριστερή παρένθεση, τα στοιχεία που είχαμε αποθηκεύσει σε προσωρινές μεταβλητές με τη σειρά που ήταν στην ουρά και μία δεξιά παρένθεση.
Με αυτόν τον τρόπο, καταφέρνουμε να διαχωρίσουμε τα στοιχεία και να θεωρείται κάθε φορά η παρένθεση με τις πράξεις και τους αριθμούς που έχει ως ένα ενιαίο ξεχωριστό στοιχείο.
Στο τέλος, τυπώσαμε την ουρά με τη χρήση της printQueue του Α' μέρους.

c. Στην DNAPalindrome αφού διαβάσαμε από τον χρήστη την ακολουθία των χαρακτήρων σε μορφή String, τρέξαμε την ακολουθία αυτή σε μία δομή επανάληψης. Ελέγξαμε τον κάθε χαρακτήρα της παράστασης και σε περίπτωση που δεν είναι κάποιο από τα επιτρεπτά γράμματα τυπώσαμε μήνυμα λάθους.
Στην περίπτωση που η παράσταση είναι σωστή και έχει μόνο τα επιτρεπτά γράμματα, βάλαμε σε μία μεταβλητή τύπου string το κενό ώστε να ασχοληθούμε μαζί της στη συνέχεια του κώδικα. Ταυτόχρονα, δημιουργήσαμε και ένα αντικείμενο τύπου StringDoubleEndedQueueImpl.
Σε μία επανάληψη τρέξαμε την παράσταση μετατρέποντας το κάθε στοιχείο της σε χαρακτήρα. Έπειτα, ελέγξαμε τον κάθε χαρακτήρα και τον μετατρέψαμε στο συμπλήρωμά του. Μετά, προσθέσαμε στην αρχή της ουράς τον κάθε χαρακτήρα (τον οποίο στείλαμε ως String) ενώ στην αρχική μεταβλητή τύπου string προσθέταμε το πρώτο στοιχείο της ουράς κάθε φορά έτσι ώστε να έχει την ίδια σειρά με την αρχική παράσταση που διαβάσαμε.
Στο τέλος, ελέγξαμε αν η αρχική παράσταση ισούται σε τιμή με την νέα παράσταση που αλλάξαμε με χρήση της equals και τυπώσαμε ανάλογα το κατάλληλο μήνυμα.