GitHub Repo: https://github.com/GiwrgosRoupas/DoubleEndedQueue

Implementation

Η εφαρμογή είναι μια υλοποίηση Double Ended Queue. Για την υλοποίηση χρησιμοποιείται ένα Array, για το οποίο κρατάμε δυο δείκτες, front και back. Η είσοδος δεδομένων στον πινάκα γίνεται
αυξάνοντας ή μειώνοντας τους δείκτες, οπότε τα νέα δεδομένα
μπορούν να προστεθούν “πίσω” ή μπροστά από τα υπάρχοντα. Μια κενή
θέση στον πινάκα έχει την τιμή null.

Concurrent Changes

Για την προστασία από **παράλληλες αλλαγές** κατά την χρήση ενός iterator, η προτροπή ήταν να υπάρχει ένας counter, τον οποίο αυξάνουμε σε κάθε αλλαγή, και ενώ κάτι τέτοιο θα είχε σχεδόν μηδενικό perfomance και memory impact, δεν θα “έπιανε” μία περίπτωση. Η περίπτωση αυτή είναι να υπάρξουν αλλαγές στην λίστα οι οποίες όμως να μην είναι ουσιαστικές ή να γινουν revert μέχρι να προσπαθήσουμε να πάρουμε το επόμενο item της λίστας. Έτσι, ο iterator κρατάει ένα snapshot του array κατά την δημιουργία του, και ελέγχει εάν το υπάρχον array είναι δομικά ίδιο. Η περίπτωση αυτή γίνεται πιο κατανοητή στα τεστ

- exceptionOnValuesAltered
- exceptionOnValuesAltered_SkipIfNotChanged

Logging

Στα log messages μπορουμε να δουμε τα εξής στοιχεία

Count: το πλήθος των δεδομένων στην λίστα

First: το πρώτο/ “μπροστά” index

Rear: το τελευταίο/ “πίσω” index

InternalView: Το array σε μορφη string κρατόντας τις κενές τιμές και την σειρά των εγγραφών

ActualView: Είναι η “user friendly” μορφή του πίνακα, φαίνονται μόνο τα κελιά που έχουν τιμή και λαμβάνονται υπόψιν οι δείκτες

Runs

1. Pop ενώ η λίστα είναι άδεια

- Input

```
var queue = new DoubleEndedQueue<Integer>();
System.out.println(queue);
queue.popLast();
```

- Output

```
Exception in thread "main" java.lang.NullPointerException Create breakpoint : Queue is empty
    at groupas.queue.DoubleEndedQueue.popLast(DoubleEndedQueue.java:71)
    at groupas.queue.Main.main(Main.java:31)
Count: 0
```

2. Προσθήκη string σε διαφορετικά σημεία

- Input

```
var stringQ = new DoubleEndedQueue<String>();
stringQ.pushFirst( elem: "first");
stringQ.pushFirst( elem: "second");
stringQ.pushLast( elem: "third");
stringQ.pushLast( elem: "fourth");
System.out.println(stringQ);
```

- Output

```
Count: 4
Front: 7
Rear: 2
InternalView: first, third, fourth, empty, empty, empty, empty, second
ActualView: second, first, third, fourth
```

3. Προσθήκη δεδομένων, με τον πίσω δείκτη να κάνει “κύκλο”

- Input

```
queue.pushFirst( elem: 1);
queue.pushFirst( elem: 2);
queue.pushFirst( elem: 3);
queue.pushFirst( elem: 4);
queue.pushFirst( elem: 5);
queue.pushFirst( elem: 6);
queue.popFirst();
queue.pushLast( elem: -2);

System.out.println(queue);
queue.popLast();
System.out.println(queue);
```

- Output

```
Count: 6
Front: 4
Rear: 1
InternalView: 1, -2, empty, empty, 5, 4, 3, 2
ActualView: 5, 4, 3, 2, 1, -2

Count: 5
Front: 4
Rear: 0
InternalView: 1, empty, empty, empty, 5, 4, 3, 2
ActualView: 5, 4, 3, 2, 1
```

4. Προσθήκη αρκετών στοιχείων ώστε η λίστα να αυξηθεί σε μέγεθος

- Input

```
queue.pushFirst( elem: 1);
queue.pushFirst( elem: 2);
queue.pushFirst( elem: 3);
queue.pushFirst( elem: 4);
queue.pushFirst( elem: 5);
queue.pushFirst( elem: 6);
queue.pushFirst( elem: 7);
queue.pushFirst( elem: 8);
System.out.println(queue);
queue.pushFirst( elem: 9);
System.out.println(queue);
```

- Output

```
Count: 8
Front: 1
Rear: 0
InternalView: 1, 8, 7, 6, 5, 4, 3, 2
ActualView: 8, 7, 6, 5, 4, 3, 2, 1

Count: 9
Front: 3
Desktop
: 11
InternalView: empty, empty, empty, 9, 8, 7, 6, 5, 4, 3, 2, 1, empty, empty, empty, empty
ActualView: 9, 8, 7, 6, 5, 4, 3, 2, 1
```