Advanced Object-Oriented Programming

CPT204 – Lab 5

Erick Purwanto

Xi'an Jiaotong-Liverpool University

西交利物浦大学

CPT204 Advanced Object-Oriented Programming

Lab 5

Linked List 2, Deque 1

Welcome !

- Welcome to Lab 5 !
  - We are going to improve the SLList in Lecture 5 into LLDeque

- You will find in this lab
  1. Lab Exercise 5.1 - 5.4, and their hints
  2. Exercise 5.1 - 5.4

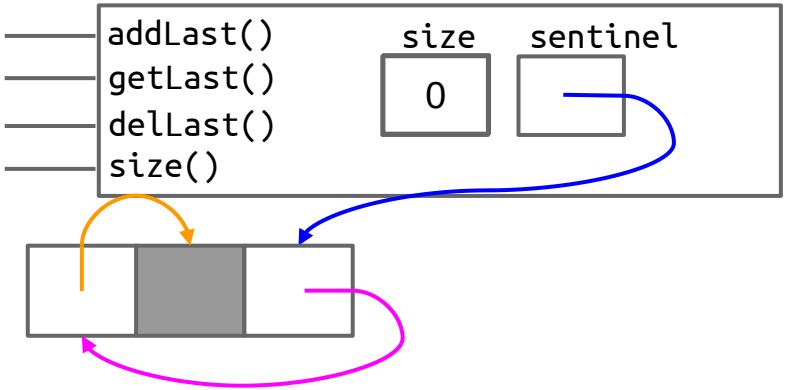
- Download **lab5** zip files from ICE
- Don't forget to import the **lab5** files and the library into an IntelliJ project
  - Read **lab1** again for reference

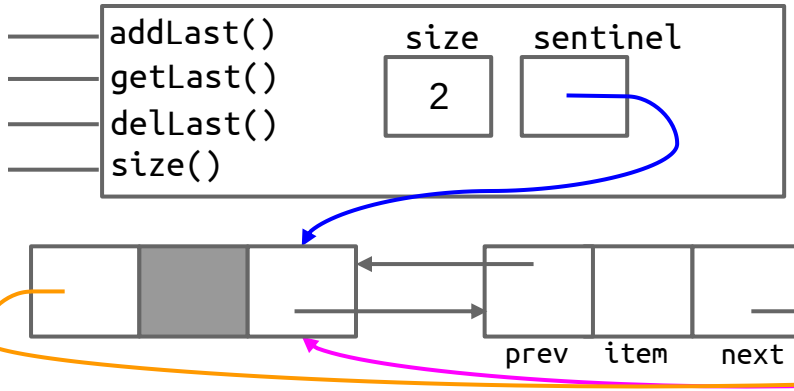
Deque

- You must have heard about **queue**, a data structure where you can add a new item at the front, and delete an item at the back, so that the first item in is the first item out (FIFO).
- In this lab, we are going to create a more general data structure called **deque**, an acronym of a double-ended queue.
- In a deque, you can add and delete item at both ends (either its front or its back).
    - In this lab, you will complete a number of methods including adding and deleting.
- In addition, you must use **generic types**, so that the deque can be instantiated to store *any type* of objects.

Lab 5 : Deque using Linked List

- We will implement a deque using Linked List, in particular, **Circular Linked List**






```
private class Node {
    Node prev;
    T item;
    Node next;

    Node(Node p, T i, Node n) {
        prev = p;
        item = i;
        next = n;
    }
}

private Node sentinel;
```

node with an item and two pointers

it has **one sentinel** that's both the front and the back

Lab Exercise 5.1 - 5.4

- Lab Exercise 5.1 LLDeque EMPTY CONSTRUCTOR
- Lab Exercise 5.2 LLDeque ADD TO FRONT
- Lab Exercise 5.3 LLDeque PRINT ITEMS
- Lab Exercise 5.4 LLDeque ITERATIVE GET ITEM

- Hint: Draw and design your code in paper, test using JUnit, and debug using Java Visualizer

Test Case for Lab Exercise 5.1 - 5.4

- Test case 1:

```
LLDeque<String> deque = new LLDeque<>();
deque.isEmpty();          →     true
deque.size();             →     0
deque.addFirst("b");
deque.addFirst("a");
deque.iterGet(0);         →     "a"
deque.iterGet(1);         →     "b"
deque.printDeque();       →     "a b↵"
deque.delFirst();
deque.iterGet(0);         →     "b"
```

Lab Exercise 5.1 LLDeque EMPTY CONSTRUCTOR

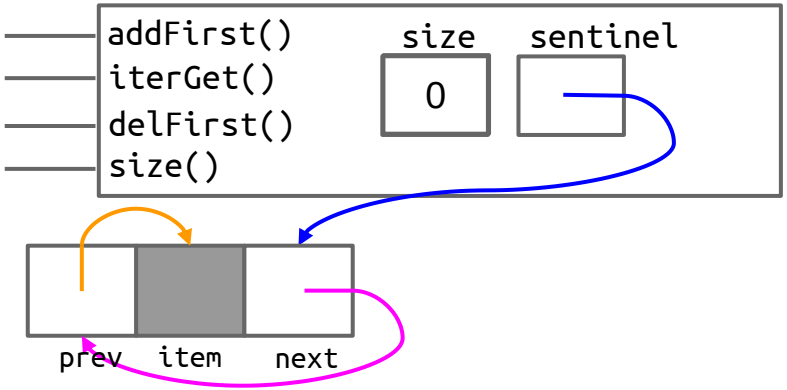
- Complete the empty deque constructor public LLDeque().
- It creates an empty deque.

WARNING: Hints to the exercise on the next slide

Please try to solve the exercise by yourself first...

Lab Exercise 5.1 LLDeque EMPTY CONSTRUCTOR Hints

- An empty deque is just a single circular sentinel node, so let us code to create that!



- Create a new node, all null parameters, and set the sentinel to point to it
- Set both its prev and next to point to itself
- Set size to 0

Lab Exercise 5.2 LLDeque ADD TO FRONT

- Complete the method void addFirst(T item).
- It adds an item of type T to the front of the deque.
- It must **not** use any loops or recursion.
- Each operation must take **constant time**, that is, it does not depend on the deque's size.

WARNING: Hints to the exercise on the next slide

Please try to solve the exercise by yourself first...

Lab Exercise 5.2 LLDeque ADD TO FRONT Hints

- We need to create a new node and place it immediately next to the sentinel

- Create a new node with the input item, with its prev points to the sentinel, and with its next point to the old node after the sentinel
- Set the prev of the old node after the sentinel to point to the new node
- Set the next of sentinel to point to the new node
- Increment the size

Lab Exercise 5.3 LLDeque PRINT ITEMS

- Complete the method void printDeque().
- It prints the items in the deque from first to last, separated by a space, ended with a new line.

WARNING: Hints to the exercise on the next slide

Please try to solve the exercise by yourself first...

Lab Exercise 5.3 LLDeque PRINT ITEMS Hints

- We need to go through every item and print it

- The item starts at next of sentinel, so set a pointer p to it
- While p is not back to sentinel
  - Print item inside node pointed by p using print (not println)
  - Add a spacebar
  - Move p to point to the next node
- Add new line with println

Lab Exercise 5.4 LLDeque ITERATIVE GET ITEM

- Complete the method T iterGet(int index) iteratively.
- It returns the item at the given index, where index 0 is the front.
  If no such item exists, it returns null.
- It must **use** loops, and **not** recursion.
- It must **not** mutate the deque.

WARNING: Hints to the exercise on the next slide

Please try to solve the exercise by yourself first...

Lab Exercise 5.4 LLDeque ITERATIVE GET ITEM Hints

- If the deque is empty, or if index is invalid (negative, greater or equal size) then return null
- Create a node pointer p that starts from next of sentinel
- Use for/while to move the pointer to the index-th node
- Return the item inside node pointed by p

Exercise 5.1 - 5.4

- Exercise 5.1 LLDeque ADD TO BACK
- Exercise 5.2 LLDeque DELETE FRONT
- Exercise 5.3 LLDeque DELETE BACK
- Exercise 5.4 LLDeque RECURSIVE GET ITEM

- Hint: Draw and design your code in paper, test using JUnit, and debug using Java Visualizer

Test Case for Exercise 5.1 - 5.4

- Test case 1:

```
LLDeque<String> deque = new LLDeque<>();
deque.addLast("a");
deque.addLast("b");
deque.addLast("c");
deque.recGet(2);          →     "c"
deque.delFirst();         →     "a"
deque.delLast();          →     "c"
deque.recGet(0);          →     "b"
```

Exercise 5.1 LLDeque ADD TO BACK

- Complete the method void addLast(T item).
- It adds an item of type T to the back of the deque.
- It must **not** use any loops or recursion.
- Each operation must take **constant time**, that is, it does not depend on the deque's size.

Exercise 5.2 LLDeque DELETE FRONT

- Complete the method T delFirst().
- It deletes and returns the item at the front of the deque.
  If no such item exists, returns null.
- It must **not** use any loops or recursion.
- Each operation must take **constant time**, that is,
  it does not depend on the deque's size.

Exercise 5.3 LLDeque DELETE BACK

- Complete the method T delLast().
- It deletes and returns the item at the back of the deque.
  If no such item exists, returns null.
- It must **not** use any loops or recursion.
- Each operation must take **constant time**, that is,
  it does not depend on the deque's size.

Exercise 5.4 LLDeque RECURSIVE GET ITEM

- Complete the method T recGet(int index) recursively.
- It returns the item at the given index, where index 0 is the front.
  If no such item exists, it returns null.
- It must **not** use loops.
- It must **not** mutate the deque.

Thank you for your attention !

- In this lab, you have learned:
    - To create a data structure called deque using circular linked list, complete with its methods that is either run in *constant-time*, *iteratively* or *recursively*