Sửa Lab01

* Exercise 2

public E removeCur(Node<E> curr) throws NoSuchElementException {

if(head == null) {

throw new NoSuchElementException();

}

else {

if (curr == null) {

....

} else {

if (head == curr) {

return removeFirst();

}

Node<E> tmp = head;

while(tmp.getNext() != curr) { // không sử dụng equals, sử dụng != để so sánh địa chỉ vì curr là Node

tmp = tmp.getNext();

}

tmp.setNext(curr.getNext());

num\_nodes--;

return tmp.getData();

}

}

}

* Exercise 3a

// Composition

public class IntLinkedList {

private MyLinkedList<Integer> list;

}

// inheritance

public class IntLinkedList extends MyLinkedList<Integer> {

public int countEven() {

int count = 0;

Node<Integer> tmp = getHead();

while(tmp != null) {

if (tmp.getData() % 2 == 0) {

count++;

}

tmp = tmp.getNext();

}

return count;

}

}

Thói quen để tăng hiệu suất khi code, tránh lỗi:

* Khi khai báo biến nên gán giá trị
* Hàm nào Return thì viết câu lệnh return ngay để chương trình không báo lỗi

Sorted Linked List

public class Node {

Integer data;

…

}

data = 1

data = 10

data = 18

Case one item: 2 → null

Case two items: 2 → 6 → null

Case more than 2 items: 2 → 6 → 11 → 17→ null

Idea:

- Trường hợp 1: head == null => addFirst

- Trường hợp 2: head != null

+ Trường hợp 2.1: head.getNext() == null (list chỉ có duy nhất 1 phần tử => Có 2 trường hợp để thêm: thêm đầu, thêm cuối)

Trường hợp 2.1.1: head.getData().compareTo(data) < 0 => addFirst

Trường hợp 2.1.2: else => addLast

+ Trường hợp 2.2 else (list có 2 phần tử hoặc nhiều hơn 2 phần tử giống nhau đều có 3 trường hợp để thêm: thêm đầu, thêm giữa, thêm cuối)

Trường hợp 2.2.1: if (head.getData().compareTo(data) > 0) => addFirst

Trường hợp 2.2.2:

else {

Node tmp = head;

while (tmp.getNext().getData().compareTo(data) < 0 && tmp.getNext() != null) { tmp = tmp.getNext();

}

Node newNode = new Node(data, tmp.getNext());

tmp.setNext(newNode);

}

Exercise 1: define Stack, Queue from scratch with Node

Exercise 2: khử đệ quy (không làm bằng đệ quy) => Tính kết quả đẩy vào Stack

Exercise 5: stack là ngược => 2 lần ngược là xuôi = Queue

Queue: a, b, c

poll();

poll();

poll();

Tạo 2 Stack

Stack<> stk1 = new …

Stack<> stk2 = new …

stk1.push(‘a’)

…

stk2.push(stk1.pop());

stk1 (top → bottom): c b a

stk2 (top → bottom): a b c

while (!stk2.isEmpty()) {

stk2.pop();

}