Hunter Damron - Comp Sci 102 - 3/25/16

/\*\*

\*

\* HomeGrownList

\*

\*/

**import** java.util.NoSuchElementException;

**public** **class** **OurLinkedList** {

**private** **ListNode** first;

**private** **int** size; //added for problem 15.4

// private ListNode tail; added for Problem 15.6

**public** **OurLinkedList**()

{

first = **null**;

size = 0;

}

// get the first object in the list

// throw a new NoSuchElementException(), if no such element exists

**public** **Object** **getFirst**( )

{

**if** (first == **null**)

**throw** **new** NoSuchElementException();

**return** first.getValue();

}

// remove the first node in the list, return its data

**public** **Object** **removeFirst**()

{

**if**(first == **null**)

**throw** **new** NoSuchElementException();

**Object** **data** = first.getValue();

first = first.getNext();

size--;

**return** data;

}

// add the data to a new node before the beginning of the list

**public** **void** **addFirst**(**Object** data)

{

first = **new** ListNode(data, first);

size++;

}

// add a node with the data to the end of the list

**public** **void** **addLast**(**Object** data)

{

**ListNode** **newbie** = **new** ListNode(data, **null**);

**if** (first == **null**){

first = newbie;

}

**else** { // find the last node in the list

**ListNode** **current** = first;

**while** (current.getNext() != **null**){

current = current.getNext();

}

current.setNext(newbie);

}

size++;

}

// remove the last node from the list, return its data

**public** **Object** **removeLast**()

{

**if** (first == **null**){

**return** **null**;

} **else** { // find the last node in the list

**ListNode** **current** = first, **prev** = **null**;

**while** (current.getNext() != **null**){

prev = current;

current = current.getNext();

}

prev.setNext(**null**);

size--;

**return** current.getValue();

}

}

// return the number of nodes currently in the list

// do this using problem 15.4 method

**public** **int** **size**()

{

**return** size;

}

// create a list iterator for this list - pass necessary data

**public** **OurLinkedListIterator** **ourListIterator**()

{

**return** **new** OurLinkedListIterator();

}

// Create a String with the contents of the list within [ ]

**public** **String** **toString**()

{

**String** **message** = "";

**OurLinkedListIterator** **iterator** = ourListIterator();

message += "HomegrownLinkedList[";

**if**(iterator.hasNext())

message += iterator.next();

**while**(iterator.hasNext()) {

message += ", " + iterator.next();

}

message += "]";

**return** message;

}

// reverse the elements in the list, problem 15.3

**public** **void** **reverse**()

{

**if**(size <= 1)

**return**;

**ListNode** **last** = first, **mover** = last.getNext();

**while**(mover != **null**) {

last.setNext(mover.getNext());

mover.setNext(first);

first = mover;

mover = last.getNext();

}

}

**public** **Object** **remove**(**int** index) {

**if**(size == 0)

**return** **null**;

**ListNode** **prev** = getNode(index - 1);

**Object** **element** = prev.getNext().getValue();

prev.setNext(prev.getNext().getNext());

size--;

**return** element;

}

**public** **Object** **get**(**int** i) {

**return** getNode(i).getValue();

}

**private** **ListNode** **getNode**(**int** i) {

**if**(i >= size || i < 0)

**return** **null**;

**ListNode** **node** = first;

**for**(; i >= 1; i--) {

node = node.getNext();

}

**return** node;

}

// You might need this method to write the OurListIterator in 15.6

**public** **ListNode** **getHead**()

{

**return** first;

}

**private** **class** **OurLinkedListIterator** **implements** ListIterator {

**boolean** canRemove;

**ListNode** position, prev;

**public** **OurLinkedListIterator**() {

canRemove = **false**;

position = **null**;

prev = **null**;

}

***@Override***

**public** **Object** **next**() {

**if**(position == **null**) {

position = first;

**return** position.getValue();

}

prev = position;

position = position.getNext();

canRemove = **true**;

**return** position.getValue();

}

***@Override***

**public** **boolean** **hasNext**() {

**if**(position == **null**)

**return** first != **null**;

**return** position.getNext() != **null**;

}

//**TODO** Fix if position == null

***@Override***

**public** **void** **add**(**Object** element) {

**if**(position == **null**) {

addFirst(element);

}

position.setNext(**new** ListNode(element, position));

prev = position;

position = prev.getNext();

size++;

canRemove = **false**;

}

***@Override***

**public** **void** **remove**() {

**if**(canRemove) {

prev.setNext(position.getNext());

position.setNext(**null**);

position = prev.getNext();

canRemove = **false**;

size--;

}

}

***@Override***

**public** **void** **set**(**Object** element) {

**if**(canRemove) {

position.setValue(element);

canRemove = **false**;

}

}

}

}

**public** **class** **OurLinkedListTester** {

**public** **static** **void** **main**(**String**[] args) {

**OurLinkedList** **list** = **new** OurLinkedList();

list.addLast(13);

list.addLast(9);

list.addLast(82);

list.addLast(3);

list.addLast(34);

list.addLast(53);

list.addLast(399);

list.addLast(1);

**System**.***out***.println("(Size = " + list.size() + ") " + list.toString());

**Object** **thing** = list.get(2);

**System**.***out***.println("Get index 2 : " + thing);

**System**.***out***.println("Removing idem at index 2: " + list.remove(2));

**System**.***out***.println("(Removed item at index 2) " + list);

list.reverse();

**System**.***out***.println("(Reversed; Size = " + list.size() + ") " + list);

}

}

Output:

(Size = 8) HomegrownLinkedList[13, 9, 82, 3, 34, 53, 399, 1]

Get index 2 : 82

Removing idem at index 2: 82

(Removed item at index 2) HomegrownLinkedList[13, 9, 3, 34, 53, 399, 1]

(Reversed; Size = 7) HomegrownLinkedList[1, 399, 53, 34, 3, 9, 13]