PL Assignment #1 : Make Linked List

과제물 부과일 : 2014-03-14(금)
Program Upload 마감일 : 2014-03-24(월) 23:59:59

문제

임의의 개수의 정수를 저장하고 있는 file("hw01.txt")에서 정수를 하나씩 읽어서 주어진 자료구조를 이용하여 List를 Java로 생성하시오.

예를 들어, 입력 data가 다음과 같으면,

```
23  248  7
```

생성되는 linked list는 다음과 같다.

```
[23] -> [248] -> [7] -> null
```

(즉, 하나의 linked list는 다수의 노드로 구성되며, 각 노드는 데이터를 의미하는 item과 다음 노드를 가리키는 next를 갖는다. 주어진 linked list의 마지막 노드의 next는 null 값을 갖는다. 자세한 자료구조는 뒤의 Java 코드를 참고하시오.)

과정

주어진 아래와 같은 메소드들의 구현을 완성하시오.

<필수로 완성할 메소드>

```
(1)private void linkLast(int element, Node x);
(2)private Node node(int index, Node x);
(3)private int length(Node x);
(4)private String toString(Node x);
```

<완성하여 제출하면 추가점수가 있는 메소드>

```
private void reverse(Node x, Node pred);
```

주의

- **"Iteration을 절대 사용하지 마시오. 즉, for, while, goto 등이 나타나면 0점 처리함"**
- **주어진 Class에서 새로운 메소드나 필드를 절대 추가하지 마시오.**
- 기타 과제 제출에 관한 구체적인 제반 사항은 각 TA의 지침에 따른다.

테스트 예

```
public class SampleTest {
	public static void sample_test(){
		RecusionLinkedList list = new RecusionLinkedList();

		list.add(1); list.add(2);
		list.add(3); list.add(4);
		list.add(5);
		System.out.println(list);

		list.add(0, 10);
		System.out.println(list);

		System.out.println(list.get(4));
		System.out.println(list.remove(0));
		System.out.println(list);

		list.reverse();
		System.out.println(list);
	}

	public static void main(String[] args){
		sample_test();
	}
}
```

<결과>

```
<terminated> SampleTest [Java Application] C:₩Program Fil
[ 1 2 3 4 5 ]
[ 10 1 2 3 4 5 ]
4
10
[ 1 2 3 4 5 ]
[ 5 4 3 2 1 ]
```

1. RecusionLinkedList.java : List를 나타내는 클래스

```
public class RecusionLinkedList{
	private Node head;

	/**
	 * 새롭게 생성된 노드를 리스트의 처음으로 연결
	 */
	private void linkFirst(int element){
		Node h = head;
		head = new Node(element, h);
	}

	/**
	 * 과제 (1)
	 * 주어진 Node x의 마지막으로 연결된 Node의 다음으로 새롭게 생성된 노드를 연결
	 * @param element 데이터
	 * @param x 노드
	 */
	private void linkLast(int element, Node x){
		if(x.next == null)
			//다음 원소로 연결
		else
			//다음 노드 방문 recusion
	}

	/**
	 * 이전 Node의 다음 Node로 새롭게 생성된 노드를 연결
	 * @param element 원소
	 * @param pred 이전노드
	 */
	private void linkNext(int element, Node pred){
		Node next = pred.next;
		pred.next = new Node(element, next);
	}


	/**
	 * 리스트의 첫번째 원소 해제(삭제)
	 * @return 첫번째 원소의 데이터
	 */
	private int unlinkFirst(){
		Node x = head;
		int element = x.item;

		head = head.next;

		x.item = Integer.MIN_VALUE;
		x.next = null;

		return element;
	}

	/**
	 * 이전Node의 다음 Node연결 해제(삭제)
	 * @param pred 이전노드
	 * @return 다음노드의 데이터
	 */
	private int unlinkNext(Node pred){
		Node x = pred.next;
		Node next = x.next;
		int element = x.item;

		x.item = Integer.MIN_VALUE;
		x.next = null;

		pred.next = next;

		return element;
	}

	/**
	 * 과제 (2)
	 * x노드에서 index만큼 떨어진 Node 반환
	 */
	private Node node(int index, Node x){
		//채워서 사용, index를 줄여가면서 다음 노드 방문
	}
```

```java
/**
 * 과제 (3)
 * 노드 끝까지의 갯수 반환
 */
private int length(Node x){
	//채워서 사용, recusion 사용
}

/**
 * 과제 (4)
 * 노드 끝까지의 내용 반환
 */
private String toString(Node x){
	//채워서 사용, recusion 사용
}

/**
 * 리스트를 거꾸로 만듬
 * @param x 현재 노드
 * @param pred 현재노드의 이전 노드
 */
private void reverse(Node x, Node pred) {
	//채워서 사용, recuison 사용
}


/**
 * 원소를 리스트의 마지막에 추가
 */
public boolean add(int element) {
	if(head == null){
		linkFirst(element);
	}else{
		linkLast(element, head);
	}

	return true;
}

/**
 * 원소를 주어진 index 위치에 추가
 * @param index 리스트에서 추가될 위치
 * @param element 추가될 데이터
 */
public void add(int index, int element) {
	if (! (index >= 0 && index <= size()) )
		throw new IndexOutOfBoundsException("" + index);

	if(index == 0)
		linkFirst(element);
	else
		linkNext( element, node(index-1, head) );
}

/**
 * 리스트에서 index 위치의 원소 반환
 */
public int get(int index) {
	if (! ( index >= 0 && index < size()) )
		throw new IndexOutOfBoundsException("" + index);

	return node(index, head).item;
}

/**
 * 리스트에서 index 위치의 원소 삭제
 */
public int remove(int index) {
	if (! ( index >= 0 && index < size()) )
		throw new IndexOutOfBoundsException("" + index);

	if(index == 0){
		return unlinkFirst();
	}
	return unlinkNext( node(index - 1, head) );
}

/**
 * 리스트를 거꾸로 만듬
 */
```

```
	public void reverse(){
		reverse(head, null);
	}

	/**
	 * 리스트의 원소 갯수 반환
	 */
	public int size() {
		return length(head);
	}

	@Override
	public String toString() {
		if (head == null)
			return "[]";

		return "[ " + toString(head) + "]";
	}

	/**
	 * 리스트에 사용될 자료구조
	 */
	private static class Node {
		int item;
		Node next;

		Node(int element, Node next) {
			this.item = element;
			this.next = next;
		}
	}
}
```

2. Test.java : 파일입력 추가 메소드와 테스트 메소드

```
	public static void addAll(RecusionLinkedList list, Scanner s){
		if( !s.hasNextInt() )
			return;
		int data = s.nextInt();
		list.add(data);
		addAll(list, s);
	}
```

<사용방법>

```
	public static void main(String[] args){
		RecusionLinkedList list = new RecusionLinkedList();

		try {
			Scanner s = new Scanner(new File("./hw01.txt"));
			addAll(list, s);
		} catch (FileNotFoundException e) {
			e.printStackTrace();
		}

		System.out.println(list);
	}
```

참고 자료

● Recursion을 사용하여 toString()하기

예를 들어, linked list가 "[23]->[248]->[7]->null"인 경우, "23 248 7"을 반환하게 된다. 23에 접근한 시점에서, Node의 next는 "[248]->[7]->null"인 liked list를 가리킨다.

"[23]->[248]->[7]->null"을 toString()하는 과정과 "[248]->[7]->null"를 toString()하는 과정은 동일하다.

그러므로, recursion을 사용하여 구현할 수가 있는 것이다.

Base case는 Node가 null일 때이다.

수정: 2014-03-14 최초작성일:2013-03-07