PL Assignment #2 : Make Linked List

과제물 부과일 : 2019-03-13(수)
Program Upload 마감일 : 2019-03-19(화) 23:59:59

문제

임의의 개수의 문자열을 저장하고 있는 file("hw01.txt")에서 문자열을 읽어서 주어진 자료구조를 이용하여 List 를 Java 로 생성하시오.

예를 들어, 입력 data 가 다음과 같으면,

```
apw
```

생성되는 linked list 는 다음과 같다.

```
[a] -> [p] -> [w] -> null
```

(즉, 하나의 linked list 는 다수의 노드로 구성되며, 각 노드는 데이터를 의미하는 item 과 다음 노드를 가리키는 next 를 갖는다. 주어진 linked list 의 마지막 노드의 next 는 null 값을 갖는다. 자세한 자료구조는 뒤의 Java 코드를 참고하시오.)

과정

주어진 아래와 같은 메소드들의 구현을 완성하시오.

<필수로 완성할 메소드>

```
(1)private void linkLast(char 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>

```
(5)private void reverse(Node x, Node pred); // 추가점수 +10%
(6)private void addAll(Node x, Node y);     // 추가점수 + 5%
```

예) 필수 메소드만 작성했을 시 10 점, 추가 메소드까지 모두 구현시 11.5 점

주의

- **"Iteration 을 절대 사용하지 마시오. 즉, for, while, goto 등이 나타나면 0 점 처리함"**
- **주어진 Class 에서 새로운 메소드나 필드를 절대 추가하지 마시오.**
- **주어진 메소드를 변경하지 마시오.**
- 기타 과제 제출에 관한 구체적인 제반 사항은 각 TA 의 지침에 따른다.

테스트 예

```
public class SampleTest {
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		RecursionLinkedList list = new RecursionLinkedList();
		RecursionLinkedList list2 = new RecursionLinkedList();

		list.add('a');		list.add('b');
		list.add('c');		list.add('d');
		list.add('e');
		System.out.println(list);


		list2.add('f');list2.add('g');
		list2.add('h');
		System.out.println(list2);

		list.add(0, 'z');
		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);


		list.addAll(list2);
		System.out.println(list);

	}
}
```

<결과>

```
[ a b c d e]
[ f g h]
[ z a b c d e]
d
z
[ a b c d e]
[ e d c b a]
[ e d c b a f g h]
```

1. RecursionLinkedList.java : List 를 나타내는 클래스

```
public class RecursionLinkedList {
	private Node head;
	private static char UNDEF = Character.MIN_VALUE;
	/**
	 * 새롭게 생성된 노드를 리스트의 처음으로 연결
	 */
	private void linkFirst(char element) {
		head = new Node(element, head);
	}
	/**
	 * 과제 (1) 주어진 Node x 의 마지막으로 연결된 Node 의 다음으로 새롭게 생성된 노드를 연결
	 *
	 * @param element
	 *            데이터
	 * @param x
	 *            노드
	 */
	private void linkLast(char element, Node x) {
		// 채워서 사용,  recursion 사용
	}
	/**
	 * 이전 Node 의 다음 Node 로 새롭게 생성된 노드를 연결
	 *
	 * @param element
	 *            원소
	 * @param pred
	 *            이전노드
	 */
	private void linkNext(char element, Node pred) {
		Node next = pred.next;
		pred.next = new Node(element, next);
	}
	/**
	 * 리스트의 첫번째 원소 해제(삭제)
	 *
	 * @return 첫번째 원소의 데이터
	 */
	private char unlinkFirst() {
		Node x = head;
		char element = x.item;
		head = head.next;
		x.item = UNDEF;
		x.next = null;
		return element;
	}
	/**
	 * 이전 Node 의 다음 Node 연결 해제(삭제)
	 *
	 * @param pred
	 *            이전노드
	 * @return 다음노드의 데이터
	 */
	private char unlinkNext(Node pred) {
		Node x = pred.next;
		Node next = x.next;
		char element = x.item;
		x.item = UNDEF;
```

```
            x.next = null;
            pred.next = next;
            return element;
}
/**
 * 과제 (2) x 노드에서 index 만큼 떨어진 Node 반환
 */
private Node node(int index, Node x) {
            // 채워서 사용,  recursion 사용
}
/**
 * 과제 (3) 노드로부터  끝까지의 리스트의 노드 갯수 반환
 */
private int length(Node x) {
            // 채워서 사용, recursion 사용
}
/**
 * 과제 (4) 노드로부터 시작하는 리스트의 내용 반환
 */
private String toString(Node x) {
            // 채워서 사용, recursion 사용
}
/**
 * 추가 과제 (5) 현재 노드의 이전 노드부터 리스트의 끝까지를 거꾸로 만듬
 * ex)노드가 [s]->[t]->[r]일 때, reverse 실행 후 [r]->[t]->[s]
 * @param x
 *             현재 노드
 * @param pred
 *             현재노드의 이전 노드
 */
private void reverse(Node x, Node pred) {
            // 채워서 사용, recursion 사용
}
/**
 * 리스트를 거꾸로 만듬
 */
public void reverse() {
            reverse(head, null);
}


/**
 * 추가 과제 (6) 두 리스트를 합침 ( A + B )
 * ex ) list1 =[l]->[o]->[v]->[e] , list2=[p]->[l] 일 때,
 *         list1.addAll(list2) 실행 후 [l]->[o]->[v]->[e]-> [p]->[l]
 * @param x
 *                   list1의 노드
 * @param y
 *                   list2 의 head
 */
private void addAll(Node x, Node y) {
            // 채워서 사용, recursion 사용
}
```

```
/**
 * 두 리스트를 합침 ( this + B )
 */
public void addAll(RecursionLinkedList list) {
	addAll(this.head, list.head);
}
/**
 * 원소를 리스트의 마지막에 추가
 */
public boolean add(char 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, char 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 char get(int index) {
	if (!(index >= 0 && index < size()))
		throw new IndexOutOfBoundsException("" + index);

	return node(index, head).item;
}
/**
 * 리스트에서 index 위치의 원소 삭제
 */
public char remove(int index) {
	if (!(index >= 0 && index < size()))
		throw new IndexOutOfBoundsException("" + index);

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

```
	/**
	 * 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 {
		char item;
		Node next;

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

2. Test.java : 파일입력 추가 메소드

```
	public static void main(String[] args) throws FileNotFoundException {
		RecursionLinkedList list = new RecursionLinkedList();
		FileReader fr;
		try {
			fr = new FileReader("hw01.txt");
			BufferedReader br = new BufferedReader(fr);
			String inputString = br.readLine();
			for(int i = 0; i < inputString.length(); i++)
			list.add(inputString.charAt(i));
		} catch (IOException e) {
			e.printStackTrace();
		}
		System.out.println(list.toString());
		list.add(3, 'b');	System.out.println(list.toString());
		list.reverse();	System.out.println(list.toString());
		// 등등 구현한 기능 추가해서 사용
	}
```

파일 입출력을 이용하되, 반드시 위와 같은 형태를 따를 필요는 없다. 입력 파일은 하나의 문자열이다. 한 라인만 있는 텍스트 파일이면 된다.
입력 파일 예) `Iloveprogramminglanguege`

참고 자료

● Recursion 을 사용하여 toString()하기
예를 들어, linked list 가 "`[a]->[b]->[c]->null`"인 경우, "`a b c`"을 반환하게 된다.
"`[a]->[b]->[c]->null`"을 toString()하는 과정과 "`[a]->[b]->null`"를 toString()하는 과정은 동일하다.

그러므로, recursion 을 사용하여 구현할 수가 있는 것이다.
Base case 는 `Node` 가 `null` 일 때이다.

수정: 2019-03-12 최초작성일:2019-03-12