자료구조 실습 보고서

[제 13 주] 리스트 반복자 - 기본기능

제출일	2017/06/06
학 번	201000287
소 속	일어일문학과
이 름	유다훈

201000287 일어일문학과 유다훈

1 프로그램 설명서

1 주요 알고리즘 및 자료구조

- 알고리즘
  - 입출력
    - 문자를 반복하여 입력받는다
      - i '!' 가 입력되면 프로그램을 종료한다.
    - '%' 를 입력하면 정수입력을 받는다. 받은 값은 리스트에 삽입한다.
      - i 미리 존재하는지 확인하여 존재하지 않으면 삽입한다.
      - ii 존재하면 존재한다는 오류 메세지를 내보낸다.
      - iii 삽입 전에 미리 검사하여 List 가 꽉 차면 삽입할 수 없다는 메세지를 출력한다.
      - iv 삽입 후에 정상적으로 삽입이 되었는지를 메세지로 출력한다.
    - '~' 를 입력하면 List 를 Empty 상태로 만든다.
    - '-' 를 입력하면 List 의 최소값을 삭제하고 그 값을 출력한다.
      - i 삭제 전 미리 검사하여 Empty 상태이면 삭제할 수 없다는 메세지를 출력한다.
    - '+' 를 입력하면 리스트에서 최대값을 삭제하고, 알려준다.
      - i 삭제 전에 검사하여 Empty 상태이면 삭제할 수 없다는 메세지를 출력한다.
    - '#'을 입력하면 List 의 길이를 출력한다.
    - '?'를 입력하면 정수를 입력받는다. 숫자에 해당하는 크기 순서의 위치에 있는 값을 List 상에서 찾아 삭제하고, 그 원소를 출력한다.
      - i 숫자가 0 보다 작거나, 또는 List 의 개수보다 크거나 같으면 찾을 수가 없으므로, 적절한 오류 메세지를 내보내고, 숫자를 다시 입력 받는다.
    - '/' 를 입력받으면 List 의 원소를 처음부터 차례대로 출력한다.

- 자료구조
  - 정렬된 Array List
  - 정렬된 Linked List
  - 자료형으로서의 Interface

1 함수 설명서

Class	AppController			
Method	메소드	파라미터 설명	리턴값	메소드 설명
	AppController()	없음	없음	_appView 변수와 _manager 변수를 초기화 하는 생성자 메소드
	run()	없음	없음	반복자 사용 프로그램을 실행하는 메소드
	void showSize()	없음	없음	해당 자료구조의 크기를 출력지시하는 메소드
	void reset()	없음	없음	리스트를 Empty 상태로 만들고 초기화 되었다는 메세지 출력
	void showAll()	없음	없음	반복자를 이용하여 리스트내부의 원소를 반복해서 출력 지시하는 메소드

	void add(int inputValue)	리스트에 추가할 값	없음	숫자를 Integer 형의 객체에 삽입 후 해당 객체를 리스트에 삽입 및 삽입 완료했다는 메세지를 출력하는 메소드
	void removeMin()	없음	없음	리스트의 최소값을 삭제 후 출력을 지시하는 메소드
	void removeMax()	없음	없음	리스트의 최대값을 삭제 후 출력을 지시하는 메소드
	void removeFrom(int aPosition)	위치값	없음	원하는 위치에 있는 리스틔 내의 값을 삭제 후 출력 지시하는 메소드
	void showMessage(MessageID aMessage)	Enum 값	없음	특정 상황마다 메세지 출력을 지시하는 메소드

Class	AppView			
Method	메소드	파라미터 설명	리턴값	메소드 설명
	AppView()	없음	없음	생성자 메소드 값을 입력받는 스캐너 생성
	String inputString()	없음	문자값	문자값을 입력받아 리턴하는 메소드
	int inputInt()	없음	숫자값	정수의 숫자값을 입력받아 리턴하는 메소드
	char inputCharacter()	없음	문자값	문자를 입력받아 그 문자의 맨 앞 글자만 리턴하는 메소드
	int inputNumber()	없음	숫자값	숫자를 입력받아 문자열의 숫자를 정수값으로 바꾸어 리턴하는 메소드
	void outputSize(int size)	숫자값	없음	리스트의 최대 사이즈를 입력받아 출력하는 메소드
	void outputAdd(int anElement)	삽입된 원소	없음	삽입된 원소값을 입력받아 출력하는 메소드
	void outputRemove(int anElement)	삭제된 원소	없음	삭제된 원소값을 입력받아 출력하는 메소드
	void outputElement(int anElement)	리스트 내부 원소	없음	리스트 내부의 원소를 차례대로 출력하는 메소드
	void outputMessage(String aMessage)	문자값	없음	문자값을 입력받아 메세지를 출력하는 메소드

Class	Interface Iterator<E>			
Method	메소드	파라미터 설명	리턴값	메소드 설명
	boolean hasNext()	없음	boolean	다음에 반복할 것이 있으면 참, 없으면 거짓을 리턴하는 메소드
	E next()	없음	원소값	리스트 내부의 존재하는 원소들을 하나씩 리턴하며 반복한다.

Class	interface List<E>			
Method	메소드	파라미터 설명	리턴값	메소드 설명
	boolean add(E anElement)	원소	boolean	입력받은 원소를 추가하여 제대로 추가하였으면 참, 추가하지 못했으면 거짓을 리턴하는 메소드
	boolean contains(E anElement)	원소	boolean	입력받은 원소가 리스트 내부에 이미 존재하는지 검사하여, 존재하면 참, 존재하지 않으면 거짓을 리턴하는 메소드
	boolean isFull()	없음	boolean	리스트가 꽉 차있다면 참, 한 칸이라도 비어있다면 거짓을 리턴하는 메소드
	boolean isEmpty()	없음	boolean	리스트가 비어있다면 참, 원소가 하나라도 들어 있다면 거짓을 리턴하는 메소드
	void clear()	없음	없음	리스트 내부를 Empty 상태로 만드는 메소드
	E removeMin()	없음	원소	리스트 내부에서 제일 작은 최소값을 리턴하는 메소드
	E removeMax()	없음	원소	리스트 내부에서 제일 큰 최대값을 리턴하는 메소드
	int size()	없음	정수형 숫자값	리스트의 최대 크기를 리턴하는 메소드
	E removeFrom(int aPosition)	위치값	원소값	리스트 내부에서 입력받은 숫자의 위치에 있는 원소를 삭제하는 메소드
	E elementAt(int anOrder)	위치값	원소값	리스트 내부에서 입력받은 숫자의 위치에 있는 원소를 리턴하는 메소드
	Iterator<E> listIterator()	없음	반복자형 Iterator	반복자 Iterator 를 리턴하는 메소드

Class	SortedArrayList <E extends Comparable<E>> implements List<E>			
Method	메소드	파라미터 설명	리턴값	메소드 설명
	SortedArrayList()	없음	없음	디폴트값으로 리스트를 생성하는 생성자 메소드
	SortedArrayList(int givenMaxSize)	정수형	없음	입력받은 정수값으로 최대값과 배열의 크기를 결정하는 생성자 메소드

Class	SortedLinkedList <E extends Comparable<E>> implements List<E>			
Method	메소드	파라미터 설명	리턴값	메소드 설명
	SortedLinkedList()	없음	없음	리스트의 현재 갯수와 head 값을 초기화시키는

				생성자 메소드

Class	ListIterator<E> implements Iterator<E>			
Method	메소드 이름	파라미터 설명	리턴값	메소드 설명
	ListIterator()	없음	없음	반복 시에 다음 값을 참조하기 위하여 사용되는 클래스 내 전역변수를 초기화하는 생성자 메소드

2 종합 설명서

- 프로그램을 실행하면 입력된 특수 기호에 따라 정해진 동작을 실행한다.
- 자료구조를 선언할 때에는 Interface List 로 선언하지만, 실제 run()내부에서 사용할 때에 그것의 인스턴스화를 사용자가 필요한 Array 형태나 Linked Chain 형태로 선언해준다.
  - ◆ 자료형은 같은 List 지만 인스턴스화시킬 때 제각각 다른 자료구조를 하여도 작동되는 것은, 이미 Linked Chain 과 Array 형태의 자료구조는 같은 Interface List 를 구현하고 있기 때문에, 사용자는 List 에서 선언한 메소드의 이름만 알고 있으면 된다.
- '%'를 입력받으면 add()가 실행되면서 삽입을 하면서 오름차순 순으로 정렬하며 삽입한다.
- '/'를 입력받으면 AppCotroller 의 showAll()이 실행되는데, 이때 각 자료구조 내부에 선언해준 반복자 클래스의 메소드를 이용하여 자료구조 내부의 원소들을 차례대로 출력한다.
- '+' 를 입력 받으면 removeMax()를 이용하여 최대값을 삭제한다.
- '?' 를 입력 받으면 다시 숫자를 입력하게 되고, 입력받은 숫자의 위치에 해당하는 원소를 removeFrom()을 이용하여 삭제한다.
- '#' 을 입력 받으면 현재 자료구조 내부에 존재하는 원소의 총 개수를 size()를 통해서 출력한다.
- '-' 를 입력 받으면 removeMin()을 이용하여 최소값을 삭제한다.
- '~' 를 입력받으면 clear()를 이용하여 자료구조 내부 원소를 모두 없애버린다.

2 프로그램 장단점 분석

- 장점
  - 데이터를 삽입할 때에 자동으로 오름차순 순으로 정렬해가며 자료구조에 삽입한다.
  - 최소값과 최대값을 바로 삭제할 수 있다
  - 자료구조 내부의 원소의 총 갯수와 어떤 원소가 들어있는지 알 수 있다.
  - Interface 와 Inner class 를 이용하여, 서로 다른 자료구조라도 사용자는 똑같은 방법으로 이용할 수 있다.
- 단점
  - 구현하는 관점에서, Inner class 를 이용한 반복자 개념을 이해하기 어렵다.

3 실행 결과 분석

1 입력과 출력

프로그램 실행

```
< 리스트를 시작합니다 >
```

입출력 1-1 - '%' 입력 후 숫자 입력 및 데이터 삽입 안내 메세지 출력

```
> 문자를 입력하시오 : %
> 숫자를 입력하시오 : 24
[Insert] 삽입된 원소는 24입니다.
```

입출력 1-2 - '/' 입력 후 자료구조 내 데이터 출력

```
> 문자를 입력하시오 : /
[LIST] 2 5 24 51 78 99 132
```

입출력 1-3 - '+' 입력 후 자료구조 내 최대값 삭제 및 안내 메세지 출력

```
> 문자를 입력하시오 : +
[Delete] 삭제된 원소는 132입니다.
```

입출력 1-4 – '?' 입력 후 숫자 입력 및 자료구조 내 해당 숫자 위치의 원소 삭제 및 안내 메세지 출력

```
> 문자를 입력하시오 : ?
> 숫자를 입력하시오 : 2
[Delete] 삭제된 원소는 24입니다.
```

입출력 1-5 - '#' 입력 후 자료구조 내 현재 존재하는 원소의 갯수 출력

```
> 문자를 입력하시오 : #
[Length] 리스트에는 현재 5개가 있습니다.
```

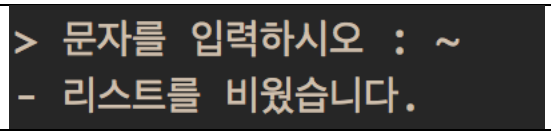
입출력 1-6 - '-' 입력 후 자료구조 내 최소값 원소 삭제 및 안내 메세지 출력

```
> 문자를 입력하시오 : -
[Delete] 삭제된 원소는 2입니다.
```

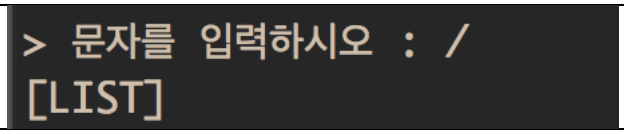
입출력 1-7 삭제 처리 후 남아 있는 원소들을 통해 삭제가 잘 되고 있는 지 확인

```
> 문자를 입력하시오 : /
[LIST] 5 51 78 99
```

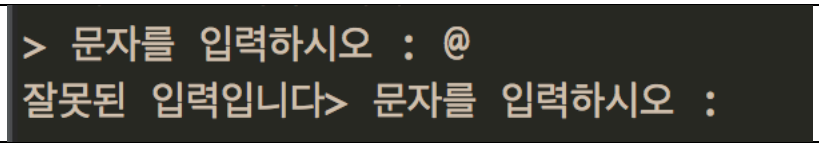
입출력 1-8 - '~' 입력 후 자료구조 초기화 및 안내 메세지 출력



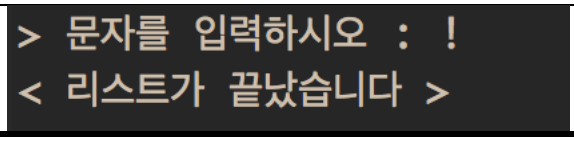
입출력 1-9 - 자료구조 초기화 후 내부 상태 확인



입출력 1-10 - 잘못된 문자 입력 후 안내 메세지 출력



프로그램 종료



2 결과 분석

- List
  - ArrayList 와 LinkedList 에서 공통적으로 사용되는 방법들을 List 로 Interface 화 시켜서, 두 개의 서로 다른 자료구조에서 무조건 구현시키도록 해놓았음.
  - ArrayList 와 LinkedList 는 서로 다른 자료구조이지만, List 를 구현함으로 인하여 동일한 이름을 가진 메소드를 가지게 됨.
  - 사용자로 하여금, 자료구조 내부의 정보를 알지 못하여도, List 에서 제공하는 공개함수만으로도 서로 다른 두개의 자료구조를 손쉽게 사용할 수 있게 됨.
    - i 내부 자료구조를 몰라도 충분히 사용할 수 있는, 구현의 독립성이 보장된다.
- Iterator
  - 서로 다른 자료구조이나 어떠한 행위를 반복적으로 하여 동일한 결과를 출력한다고 할 때 사용.
  - Interface 로 구현을 해주고, 이 Interface 를 해당 자료구조 내의 Inner class 에 구현을 한다.
  - Inner class 는 Inner class 를 품고 있는 class 의 private 변수에도 접근이 가능하다.
  - 사용자는 Interface 로 선언된 Iterator 를 이용하는 것만으로도 서로 다른 자료구조 내부의 private 변수를 간접적으로 참조하여 같은 반복 행위를 할 수 있게 된다.
- 두 자료구조 구현의 장단점

- ArrayList
  - i 본 프로그램은 삽입 시에 정렬을 하며 오름차순 순으로 삽입하는 프로그램이다. 따라서 삽입 시에 정렬을 실행하는데, 배열을 이용하는 ArrayList 는 중간 부분에 삽입 시, 그 뒷부분을 모두 한 칸씩 미뤄야하기에 삽입할 때 번거로움이 있다.
  - ii 정렬이 이루어진 상태에서, 최소값과 최대값은 각각 배열의 앞, 뒷부분이므로 쉽게 참조할 수 있다 ▶ 시간이 빠르다. 그러나 최소값을 삭제할 경우, 뒤의 데이터들을 전부 한 칸씩 당겨주어야 하므로 시간이 걸린다.
  - iii 원하는 위치의 데이터값을 삭제할 때도 해당 위치를 인덱스로 사용한다면 쉽게 삭제할 수 있다. 그러나 그 이후의 데이터들을 한 칸씩 당겨주어야 하므로 시간이 걸린다.
  - iv 자료구조를 초기화 시키는 것도 모든 배열의 원소값을 하나씩 null 처리를 해줘야 하므로 시간이 걸린다.
- LinkedList
  - i 삽입 시 정렬을 실행할 때, 중간 위치에 삽입하게 된다면 그냥 앞 노드와 뒤 노드 사이에 넣기만 하면 되므로 삽입 시간이 빠르다.
  - ii 최소값과 최대값은 각각 앞 부분과 뒷부분에 있으므로, 삭제가 간편하다. 그러나 최대값을 삭제할 경우, Linked Chain 의 특성상 인덱스를 이용할 방법이 없으므로, Chain 의 맨 뒷부분까지 반복문을 이용하여 참조해야할 필요가 있다.
  - iii 원하는 위치의 데이터를 삭제할 때도 해당 위치까지 일일이 이동하는 시간이 걸리나, 해당 위치의 뒷부분에 있는 원소들을 당겨줘야할 필요는 없다.
  - iv 자료구조를 초기화 시키는 것도 _head 값만 null 처리 해주면 되는 부분이므로 매우 간편하다.

- Iterator 를 사용해보고 느낀 점
  - 매우 복잡하고 처음 코딩할때에는 작동이 안되어서 어떻게 해야하는 건지 이해하기가 어려웠다.
  - 프로그램 상에서, ListIterator<E>를 반환하는 listIterator()메소드를 사용하였는데, 정작 List 에는 Iterator<E>를 반환하는 형태로 사용되었다. 이 부분이 이해하기가 어려웠지만, ListIterator 는 결국 Iterator 를 구현하기 때문에 가능한 것이라고 이해했다.
  - 본 프로그램에서 사용된 반복 수준은 굳이 반복자를 사용하지 않아도 충분히 구현할 수 있지 않았을까 라고 생각한다.

- 결론
  - SortedArrayList
    i 삽입시간 : O(n)
    ii 최대값 삭제 : O(1)
    iii 최소값 삭제 : O(n)
    iv 원하는 위치 삭제 : O(n)
    v 초기화 : O(n)
  - SortedLinkedList
    i 삽입시간 : O(n)
    ii 최대값 삭제 : O(n)
    iii 최소값 삭제 : O(1)
    iv 원하는 위치 삭제 : O(n)
    v 초기화 : O(1)
  - Interface 를 이용한 자료구조의 구현 및 사용법은 사용자에게 있어서 매우 간편하다.
  - Iterator 또한 사용하기 간편하나, 이해하기 어려웠다.