자료구조 실습 보고서

[제 14-2 주] Call Back 의 구현

제출일	2017/06/19
학 번	201000287
소 속	일어일문학과
이 름	유다훈

201000287 일어일문학과 유다훈

1 프로그램 설명서

1 주요 알고리즘 및 자료구조

- 알고리즘
  - 입출력
    - ◆ 입력은 없음.
      - i 필요한 데이터는 프로그램에서 생성
    - ◆ 출력
      - i 이진트리에 원소를 삽입해나가는 모습을 출력한다.
      - ii 이진트리를 중위탐색한 순서대로 출력한다.
      - iii 이진트리에 원소를 삭제해나가는 모습을 출력한다.
- 자료구조
  - Binary Search Tree

1 함수 설명서

Class	AppController implements VisitEventForDictionaryByBinarySearchTree<Integer, Integer>			
Method	메소드	파라미터 설명	리턴값	메소드 설명
	AppController()	없음	없음	_appView 변수를 초기화 하는 생성자 메소드
	void run()	없음	없음	이진탐색트리를 Traversal 하는 프로그램을 실행하는 메소드
	void addToBinarySearchTreeAndShowShape()	없음	없음	이진탐색트리에 원소를 추가하고 추가하면서의 이진탐색트리의 모습을 출력지시하는 메소드
	void showInorderOfBinarySearchTree()	없음	없음	이진탐색트리를 중위탐색한 모습을 차례대로 출력 지시하는 메소드
	void removeFromBinarySearchTreeAndShowShape()	없음	없음	이진탐색트리에서 원소를 삭제하고, 삭제하면서의 이진탐색트리의 모습을 출력지시하는 메소드

Class	AppView			
Method	메소드	파라미터 설명	리턴값	메소드 설명
	AppView()	없음	없음	생성자 메소드 값을 입력받는 스캐너

				생성
	void output(String aString)	문자열	없음	줄바꿈을 하지 않는 메세지를 출력
	void outputLine(Stirng aString)	문자열	없음	줄바꿈을 하는 메세지를 출력.

Class	DataGenerator			
Method	메소드	파라미터 설명	리턴값	메소드 설명
	static int[] ascendingOrderList(int aSize)	크기	정수형 배열	전달받은 크기만큼의 오름차순 데이터를 생성
	static int[] descendingOrderList(int aSize)	크기	정수형 배열	전달받은 크기만큼의 내림차순 데이터를 생성
	static int[] randomOrderList(int aSize)	크기	정수형 배열	전달받은 크기만큼의 무작위순 데이터를 생성

Class	Interface VisitEventForDictionaryByBinarySearchTree<K extends Comaparable <K>, O>			
Method	메소드	파라미터 설명	리턴값	메소드 설명
	void visitForInorder (DictionaryElement<K, O> anElement, int aLevel)	DictionaryElement, 각 노드의 레벨	없음	중위탐색으로 노드에 방문했을 시 해야할 작업들을 하는 메소드
	void visitForReverserOfInorder(DictionaryElement<K, O> anElement, int aLevel)	DictionaryElement, 각 노드의 레벨	없음	왼쪽, 가운데, 오른쪽 순으로 방문하는 중위탐색의 역순으로 오른쪽, 가운데, 왼쪽 순으로 방문하며 방문했을 시 해야할 작업을 하는 메소드

Class	BinaryNode			
Method	메소드 이름	파라미터 설명	리턴값	메소드 설명
	BinaryNode()	없음	없음	BinaryNode 를 모두 null 값으로 초기화시키는 생성자 메소드
	BinaryNode(T givenElement, BinaryNode<T> givenLeft, BinaryNode<T> givneRight)	원소, 왼쪽노드, 오른쪽 노드	없음	전달받은 값으로 BinaryNdoe 의 원소와 왼쪽, 오른쪽 자식을 초기화하는 메소드
	T element()	없음	원소	원소를 리턴하는 메소드
	void setElement(T newElement)	원소	없음	현재 노드의 원소를 새롭게 지정하는 메소드
	BinaryNode<T> left()	없음	BinaryNode	현재노드의 왼쪽 노드를 리턴하는 메소드
	void setLeft(BinaryNode<T> newLeft)	BinaryNode	없음	현재 노드의 왼쪽 노드를 새롭게 지정하는 메소드
	BinaryNode<T> right()	없음	BinaryNode	현재 노드의 오른쪽 노드를 리턴하는 메소드
	void setRight(BinaryNode<T> newRight)	BinaryNode	없음	현재 노드의 오른쪽 노드를 새롭게 지정하는 메소드

Class	abstact Dictinary<K, O>			
Method	메소드 이름	파라미터 설명	리턴값	메소드 설명
	int size()	없음	크기	사전의 크기값을 리턴하는 메소드
	void setSize(int newSize)	크기	없음	사전의 크기 값을 새롭게 설정하는 메소드
	Dictionary()	없음	없음	사전을 생성하는 생성자 메소드
	boolean isEmpty()	없음	boolean	사전이 비어있는지 없는지를 리턴하는 메소드
	abstract boolean isFull()	없음	boolean	사전이 꽉 찼는지 아닌지를 리턴하는 메소드
	abstract boolean keyDoesExist(K aKey)	Key	boolean	사전에 해당 키가 존재하는지 여부를 리턴하는 메소드
	abstract O objectForKey(K aKey)	key	Object	전달받은 키 값의 오브젝트를 리턴하는 메소드
	abstract boolean addKeyAndObject(K aKey, O anObject)	key, object	boolean	키와 오브젝트를 사전에 추가하고 추가 여부를 리턴하는 메소드
	abstract O removeObjectForKey(K aKey)	key	object	입력받은 키값과 해당 키값의 오브젝트를 삭제하는 메소드
	abstract void clear()	없음	없음	사전을 초기화시키는 메소드

Class	DictinaryElement <K extends Comparable<K>, O>			
Method	메소드 이름	파라미터 설명	리턴값	메소드 설명
	DictionaryElement(K givenKey, O givenObject)	key, Object	없음	전달받은 키와 오브젝트로 사전의 원소를 생성하는 메소드
	K key()	없음	key	현재 원소의 키를 리턴하는 메소드
	void setKey(K newKey)	key	없음	현재 원소의 키를 새롭게 설정하는 메소드
	O object()	없음	Object	현재 원소의 오브젝트를 리턴하는 메소드
	void setObject(O newObject)	Object	없음	현재 원소의 오브젝트를 설정하는 메소드

Class	DictionaryByBinarySearchTree<K extends Comparable<K>, O> extends Dictionary<K, O>			
Method	메소드 이름	파라미터 설명	리턴값	메소드 설명
	DictionaryByBinarySearchTree()	없음	없음	BinarySearch Tree 를 초기화시키는 메소드
	BinaryNode<DictinaryElement<K,O>> root()	없음	BinaryNode	트리의 루트값을 리턴하는 메소드

	void setRoot(BinaryNode<DictinaryElement<K, O>> newRoot)	BinaryNode	없음	트리의 루트값을 새롭게 설정하는 메소드
	DictionaryElement elementForKey(K aKey)	Key	DictionaryElement	키가 들어있는 노드를 찾아 해당 키와 맞는 노드의 원소를 리턴한다.
	DictionaryElement<K, O> removeRightMostElementOfLeftSubTree (BinaryNode<DictinaryElement<K,O>> root)	root	DictionaryElement	왼쪽 서브트리의 최대값을 찾아 삭제하는 메소드
	VisitEventForDictionaryBiBinarySearchTree visitEvent()	없음	VisitEventForDictionaryBiBinarySearchTree	인터페이스 객체값을 리턴해주는 메소드
	void setVisitEvent(VisitEventForDictionaryByBinarySearchTree newVisitEvent)	VisitEventForDictionaryBiBinarySearchTree	없음	인터페이스 객체값을 받아서 설정하는 메소드
	void inorder()	없음	없음	중위탐색을 시작하는 메소드
	void inorderRecursively(BinaryNode<DictionaryElement<K, O>> aRootOfSubtree, int aLevel)	DictionaryElement, 각 노드의 레벨	없음	중위탐색을 재귀적으로 실행해가며 인터페이스에 규약된 메소드를 실행하는 메소드
	void reverseInorder()	없음	없음	중위탐색을 역방향으로 실행하는 메소드
	void reverseOfInorderRecursively(BinaryNode<DictionaryElement<K, O>> aRootOfSubtree, int aLevel)	DictionaryElement, 각 노드의 레벨	없음	중위 탐색을 역방향으로 재귀적으로 실행하는 메소드

2 종합 설명서

- 프로그램을 실행하면 정해진 디폴트 값만큼의 원소가 생성된다. 본 프로그램에서는 무작위순으로 생성된 데이터 리스트를 사용한다.
- 무작위 데이터가 담긴 배열을 이용하여 DictionaryByBinarySearchTree 에 데이터를 추가해준다. 배열의 값이 키값으로 , 배열의 인덱스값이 오브젝트값으로 하여 삽입을 한다.

- 삽입을 해나가는 과정에서의 이진탐색트리가 어떻게 변화하는지를 출력한다.
- 삽입 완료 후 해당 이진탐색트리를 중위탐색한 결과를 출력한다.
- 중위탐색 종료 후, 원소를 한개씩 삭제해나가며 트리가 어떻게 변화하는지 출력한다.

2 프로그램 장단점 분석

- 장점
  - 주어진 데이터를 이용하여 실제로 이진탐색트리가 어떻게 생성되는지 그 모습을 확인할 수 있다.
  - 삽입과 삭제 과정에서 코드가 어떻게 동작하는지 확인할 수 있다.
  - 트리의 중위 탐색 결과를 확인할 수 있다.
  - Interface 를 이용한 Call Back 개념을 구현할 수 있다.
  - Call Back 개념을 구현하여 모델로직을 구현할 때에는 오직 사용자가 원하는 행위만 하게끔 구현하여, AppController 에서 해당 모델을 사용하는 사용자가 자신의 목적과 의도에 알맞게 코드를 구현하여 모델을 수정하지않고도 사용할 수 있다.
- 단점
  - 프로그램을 구현하면서, 이해하기가 어렵다.

3 실행 결과 분석

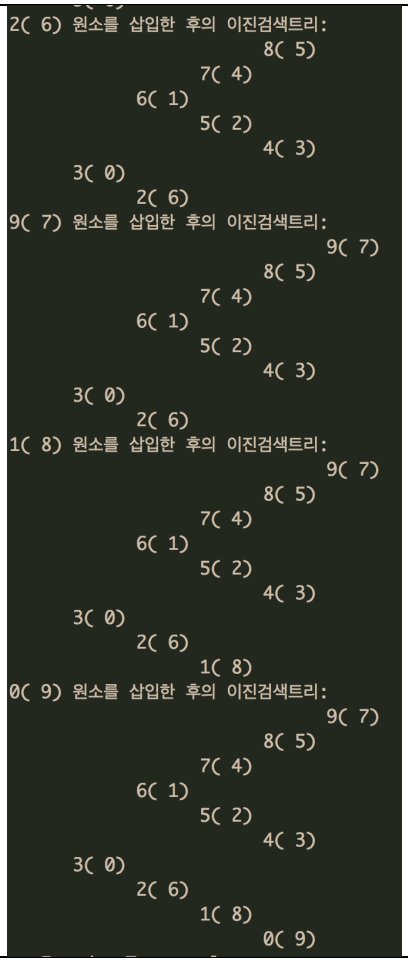
1 입력과 출력

프로그램 실행

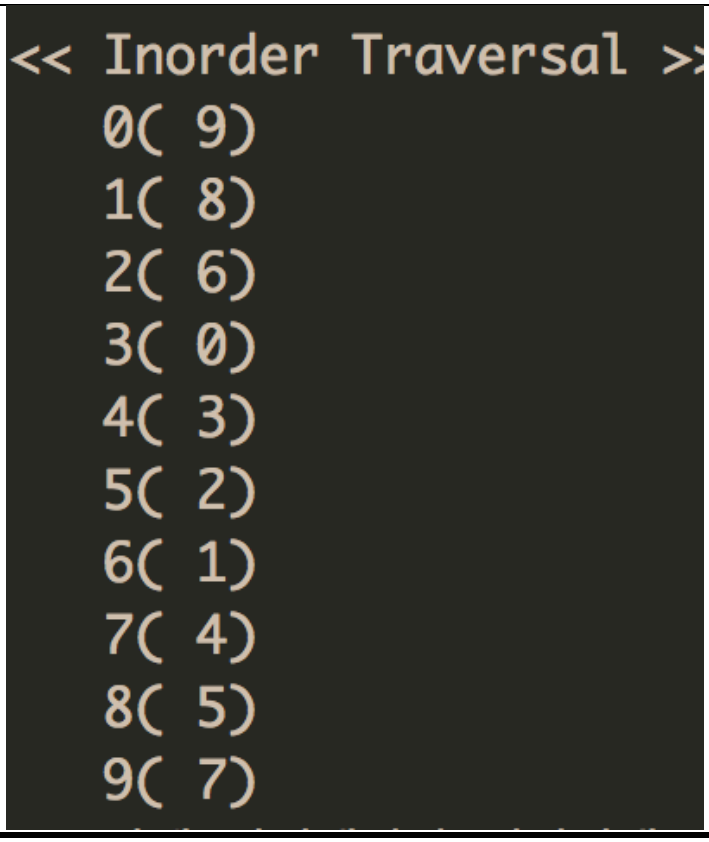
<< 삽입 과정에서의 이진검색트리의 변화 >>

출력 1 - 삽입을 전개해나가며 이진탐색트리가 변화하는 과정

```
3( 0) 원소를 삽입한 후의 이진검색트리:
      3( 0)
6( 1) 원소를 삽입한 후의 이진검색트리:
            6( 1)
      3( 0)
5( 2) 원소를 삽입한 후의 이진검색트리:
            6( 1)
                  5( 2)
      3( 0)
4( 3) 원소를 삽입한 후의 이진검색트리:
            6( 1)
                  5( 2)
                        4( 3)
      3( 0)
7( 4) 원소를 삽입한 후의 이진검색트리:
                  7( 4)
            6( 1)
                  5( 2)
                        4( 3)
      3( 0)
8( 5) 원소를 삽입한 후의 이진검색트리:
                        8( 5)
                  7( 4)
            6( 1)
                  5( 2)
                        4( 3)
      3( 0)
```



출력 2 - 이진탐색트리를 중위탐색한 결과.



출력 3 - 이진탐색트리에서 원소를 하나씩 제거해나가며 이진탐색트리의 변화과정 출력 및 프로그램 종료

```
Key 값이 3인 원소를 삭제한 후의 이진검색트리:
                                9( 7)
                            8( 5)
                        7( 4)
                6( 1)
                        5( 2)
                            4( 3)
        2( 6)
                1( 8)
                        0( 9)
Key 값이 6인 원소를 삭제한 후의 이진검색트리:
                                9( 7)
                            8( 5)
                        7( 4)
                5( 2)
                        4( 3)
        2( 6)
                1( 8)
                        0( 9)
Key 값이 5인 원소를 삭제한 후의 이진검색트리:
                                9( 7)
                            8( 5)
                        7( 4)
                4( 3)
        2( 6)
                1( 8)
                        0( 9)
Key 값이 4인 원소를 삭제한 후의 이진검색트리:
                            9( 7)
                        8( 5)
                    7( 4)
            2( 6)
                    1( 8)
                        0( 9)
Key 값이 7인 원소를 삭제한 후의 이진검색트리:
                        9( 7)
                    8( 5)
            2( 6)
                    1( 8)
                        0( 9)
Key 값이 8인 원소를 삭제한 후의 이진검색트리:
                    9( 7)
            2( 6)
                    1( 8)
                        0( 9)
Key 값이 2인 원소를 삭제한 후의 이진검색트리:
                    9( 7)
            1( 8)
                    0( 9)
Key 값이 9인 원소를 삭제한 후의 이진검색트리:
            1( 8)
                    0( 9)
Key 값이 1인 원소를 삭제한 후의 이진검색트리:
            0( 9)
Key 값이 0인 원소를 삭제한 후의 이진검색트리:
```

2 결과 분석

- 이진탐색트리의 모습
  - 스크린샷에서, 레벨이 가장 낮은 원소(띄어쓰기가 가장 적게 된, 왼쪽에 있는 원소) 가 트리의 루트 원소이다.
  - 루트를 기준으로 아래는 트리의 왼쪽, 위는 트리의 오른쪽이다.
  - 삽입 시에 루트와 비교 하여 값이 크면 오른쪽, 작으면 왼쪽으로 배치를 해나가며 트리를 그리고 있다.
  - 삭제 시는 루트 노드를 삭제할 때, 루트의 왼쪽서브트리의 가장 큰 오른쪽 값을 루트 노드로 대체하고 있다.
- 중위 탐색
  - 이진탐색트리의 특성상 루트보다 작은 값은 왼쪽으로, 큰 값은 오른쪽으로 나가며 서브트리를 구성하게 된다.
  - 중위 탐색 순서는 왼쪽-중간-오른쪽 순이다.
  - 이진탐색트리의 특성과 중위 탐색의 순서로 인하여 중위탐색결과는 Key 값들의 오름차순으로 나열된다.
- Call Back
  - AppController 에서 Interface 를 구현하였다.
  - AppController 에서는 Interface 를 구현한 자신을 DictionaryByBinarySearchTree 로 전달해주는데, DictionaryByBinarySearchTree 에서는 이것으로 인터페이스의 객체를 생성할 수 있다.
  - 같은 인터페이스를 사용하는 객체와 클래스로 인하여, 해당 객체는 해당 인터페이스 내에 선언되어 있는 메소드들을 사용할 수 있다.
  - AppController 에서는 인터페이스들의 메소드들을 무조건 구현해주어야 하는데, 이것으로 인하여 모델 내부에서는 인터페이스의 메소드만 사용할 줄 알면, 구현방법에 대해서까지는 신경을 쓰지 않아도 되는 상태가 된다.
  - 반면 AppController 에서는 노드를 방문했을 때 자신의 마음대로 하고싶은 행동을 정의해서 사용할 수 있다.
  - 이렇게 한다면 모델내부에서 출력 등의 Model-Controller-View 의 설계 관점을 저해하지 않고, 또한 사용할 때 마다 사용자에 맞는 의도로 수정할 필요없이 코드를 사용할 수 있어서 코드의 재활용에도 매우 도움이 된다.
- 결론
  - 실제 이진탐색트리가 어떻게 작동되는지 눈으로 확인할 수 있다.
  - Call back 의 효율성을 알 수 있다.