자료구조: 2022년 1학기 [강의]

Array Set, Linked Set

강지훈
jhkang@cnu.ac.kr
충남대학교 컴퓨터융합학부

집합 (Set)

Set 객체 사용법

- Set 객체 생성과 소멸
  - Set 객체 생성
- Set 상태 알아보기
  - Set 에 들어있는 원소의 개수를 알려주시오
  - Set 이 비어 있는지 알려주시오
  - Set 이 가득 찾는지 알려주시오
  - 주어진 원소가 Set 에 있는지 알려주시오
- Set 내용 알아보기
  - Set 에서 아무 원소 하나를 얻어내시오
- Set 내용 바꾸기
  - Set 에 주어진 원소를 넣으시오
  - Set 에서 아무 원소 하나를 제거하여 얻어내시오
  - Set 에서 지정된 원소를 찾아서 있으면 제거하시오
  - Set 을 비우시오

Class "Set<E>"

Class Set<E> 의 공개함수

- Set 객체 사용법을 Java 로 구체적으로 표현
  - public Set () { }
  - public int size () { }
  - public boolean isEmpty () { }
  - public boolean isFull () { }
  - public boolean doesContain (E anElement) { }
  - public E any() { }
  - public boolean add (E anElement) { }
  - public E removeAny () { }
  - public boolean remove (E anElement) { }
  - public void clear () { }

Class "ArraySet<E>"

"ArraySet" as a Set

- Java Array 를 이용하여 구현

- ArraySet
  - 집합에는 동일한 원소가 들어 있을 수 없다.
  - 동일한 원소란?
  - "equals()"
    - ArraySet 객체의 사용자가, 원소의 구현에 반영

ArraySet<E> 의 공개함수

- ArraySet 객체 사용법을 Java 로 구체적으로 표현
  - public ArraySet() { }
  - public ArraySet(int givenCapacity) { }
  - public int size () { }
  - public boolean isEmpty () { }
  - public boolean isFull () { }
  - public boolean doesContain (E anElement) { }
  - public E any() { }
  - public boolean add (E anElement) { }
  - public E removeAny( ) { }
  - public boolean remove (E anElement) { }
  - public void clear () { }

Class "ArraySet"의 초기 형태는 이렇게!

```
public class  ArraySet<E>
{
        public  ArraySet ( )
        {       // 수정해야 함
        }
        public  ArraySet (int givenCapacity)
        {       // 수정해야 함
        }

        public int     size ()
        {
                return 0 ; // 수정해야 함
        }
        public boolean      isEmpty ()
        {
                return true ; // 수정해야 함
        }
        public boolean      isFull ()
        {
                return true ; // 수정해야 함
        }
        public boolean      doesContain (E anElement)
        {
                return true ; // 수정해야 함
        }

        public E       any()
        {
                return  null ; // 수정해야 함
        }

        public boolean      add (E anElement) { return true ; /* 수정해야 함 */ }
        public E            removeAny () { return null ; /* 수정해야 함 */ }
        public boolean      remove (E anElement) { return true ; /* 수정해야 함 */ }
        public void         clear () { /* 수정해야 함 */ }

} // End of Class "ArrayBag"
```

이렇게만 정의해 두어도 사용하는 곳에서 프로그래밍 하는 데는 전혀 지장이 없다. 즉 컴파일 오류가 발생하지 않는다.

ArraySet 의 구현

```
public class  ArraySet<E>
{
    // 비공개 상수
    private static final int  DEFAULT_CAPACITY = 100 ;

    // 비공개 인스턴스 변수
    private int          _capacity ;
    private int          _size ;
    private E[]          _elements ; // ArraySet 의 원소들을 담을 java 배열
```

ArraySet: Getter / Setter

```
public class  ArraySet<E>
{
      ......
      // 비공개 인스턴스 변수
      private int          _capacity ;
      private int          _size ;
      private E[]          _elements ; // 원소들을 저장할 배열

      // Getters / Setters
      private int capacity() { // Class 내부에서만 사용
            return this._capacity ;
      }
      private void setCapacity (int newCapacity) { // Class 내부에서만 사용
            this._capacity = newCapacity ;
      }

      pubic int size() { // 공개 함수
            return this._size ;
      }
      private void setSize (int newSize) { // Class 내부에서만 사용
            this._size = newSize ;
      }

      private E[] elements () { // Class 내부에서만 사용
            return this._elements ;
      }
      private void setElements (E[] newElements) { // Class 내부에서만 사용
            this._elements = newElements ;
      }
```

ArraySet 의 구현: 객체의 생성

```
public class  ArraySet<E>
{
    ……

    // 비공개 인스턴스 변수
    ……

    // 생성자
    @SuppressWarnings ("unchecked")
    public ArraySet  () {
        this._capacity = ArraySet.DEFAULT_CAPACITY ;
        this._elements = (E[]) new Object[this._capacity] ;
        this._size = 0 ;
    }

    @SuppressWarnings ("unchecked")
    public ArraySet  (int givenCapacity) {
        this._capacity = givenCapacity ;
        this._elements = (E[]) new Object [this._capacity] ;
        this._size = 0 ;
    }
```

Getter/Setter 를 사용하자

```
public class  ArraySet<E>
{
    ……

    // 비공개 인스턴스 변수
    ……

    // 생성자
    @SuppressWarnings (“unchecked”)
    public ArraySet  () {
        this.setCapacity (ArraySet.DEFAULT_CAPACITY) ;
        this.setElements ( (E[]) new Object[this.capacity()] ) ;
        this.setSize (0) ;
    }

    @SuppressWarnings ("unchecked")
    public ArraySet  (int givenCapacity) {
        this.setCapacity (givenCapacity) ;
        this.setElements ( (E[]) new Object [this.capacity()] ) ;
        this.setSize (0) ;
    }
```

```
@SuppressWarnings ("unchecked")
public ArraySet  () {
        this._capacity = ArraySet.DEFAULT_CAPACITY ;
        this._elements = (E[]) new Object[this._capacity] ;
        this._size = 0 ;
}

@SuppressWarnings ("unchecked")
public ArraySet  (int givenCapacity) {
        this._capacity = givenCapacity ;
        this._elements = (E[]) new Object [this._capacity] ;
        this._size = 0 ;
}
```

ArraySet 의 구현 : 상태 알아보기

```
public class  ArraySet<E>
{
    ……

    // 생성자
    ……

    // 상태 알아보기

    // public int  size () { }

    public boolean  isEmpty ()
    {
        return  (this.size() == 0) ;
    }

    public boolean  isFull ()
    {
        return  (this.size() == this.capacity()) ;
    }
```

ArraySet: 상태 알아보기

```
// 상태 알아보기
.....
public boolean  doesContain (E anElement)
{
    boolean  found = false ;
    int  i ;
    for ( i = 0 ;  i < this.size()  && ! found ; i++ ) {
        if ( this.elements()[i].equals(anElement) ) {
            found = true ;
        }
    }
    return  found ;
}
```

ArraySet: 내용 알아보기

```
// 내용 알아보기
public E   any ()
{
    if ( this.isEmpty() ) {
        return   null ;
    }
    else {
        return   this.elements()[0] ;
    }
}
```

ArraySet: add()

```
// 내용 바꾸기
public boolean  add (E anElement)
{
    if ( this.isFull() ) {
        return  false ;
    }
    else {
        if ( ! this.doesContain(anElement) ) {
            this.elements()[this.size()] = anElement ;
            this.setSize(this.size()+1) ;
            return  true ;
        }
        else {
            return  false ;
        }
    }
}
```

ArraySet : removeAny()

```
// 내용 바꾸기
.....
public E  removeAny ()
{
    if ( this.isEmpty() ) {
        return  null ;
    }
    else {
        this.setSize(this.size()-1) ;
        E  removedElement = this.elements()[this.size()] ;
        this.elements()[this.size()] = null ;
        return  removedElement ;
    }
}
```

ArraySet : remove()

```
// 내용 바꾸기
.....
public boolean  remove (E anElement)
{
    // 첫 번째 단계: 삭제할 원소의 위치 찾기
    int  foundIndex = -1 ;
    for ( int  i = 0 ;  i < this.size() && foundIndex < 0 ; i++ ) {
        if ( this.elements()[i].equals(anElement) ) {
            foundIndex = i ;
        }
    }

    // 단계 2: 삭제된 원소 이후의 모든 원소를 앞쪽으로 한 칸씩 이동시킨다.
    if ( foundIndex < 0 ) {
        return  false ;
    }
    else {
        for ( int i = foundIndex ; i < this.size()-1 ; i++ ) {
            this.elements()[i] = this.elements()[i+1] ;
        }
        this.elements()[this.size()-1] = null ; // 더 이상 의미가 없는 소유권은 null 로!
        this.setSize (this.size()-1) ;
        return  true ;
    }
}
```

ArraySet: clear()

```
// 내용 바꾸기
.....

public void  clear ()    // Set 을 비운다
{
    for ( int  i = 0 ; i < this.size() ; i++ ) {
        this.elements()[i] = null ;
    }
    this.setSize (0) ;
}
```

Class "LinkedSet"

LinkedSet 의 공개함수

- LinkedSet 객체 사용법을 Java 로 구체적으로 표현
  - public LinkedSet () { }
  - public LinkedSet (int givenCapacity) { }

  - public int size () { }
  - public boolean isEmpty () { }
  - public boolean isFull () { }
  - public boolean doesContain (E anElement) { }

  - public Element any () { }

  - public boolean add (E anElement) { }
  - public E removeAny () { }
  - public boolean remove (E anElement) { }
  - public void clear () { }

LinkedSet 의 구현

```
public class  LinkedSet<E>
{
    // 비공개 인스턴스 변수
    private int                 _size ;
    private LinkedNode<E>       _head ;
```

LinkedSet: Getter/Setter

```
public class  LinkedSet<E>
{
    // 비공개 인스턴스 변수
    private int                 _size ;
    private LinkedNode<E>       _head ;

    // Getters/Setters
    public int  size() {
        return  this._size ;
    }
    private void  setSize (int newSize) {
        this._size = newSize ;
    }

    private LinkedNode<E>  head() {
        return  this._head ;
    }
    private void  setSize (LinkedNode<E> newHead) {
        this._head = newHead ;
    }
```

LinkedSet 의 구현: 객체의 생성

```
public class  LinkedSet<E>
{
    // 비공개 인스턴스 변수
    ……

    // 생성자
    public  LinkedSet ()
    {
        this.setSize (0) ;
        this.setHead (null) ;
    }
```

LinkedSet 의 구현: 상태 알아보기

```
public class  LinkedSet<E>
{
    // 비공개 인스턴스 변수
    ……

    // 생성자
    ……

    // 상태 알아보기
    // public int  size ()
    // {
    //      return  this._size ;
    // }

    public boolean  isEmpty ()
    {
        return  (this.size() == 0) ; // 또는 return (this.head() == null) ;
    }

    public boolean  isFull ()
    {
        return  false ;
    }
```

LinkedSet: 상태 알아보기

```
// 상태 알아보기
.....
// 순차 검색
public boolean  doesContain (E anElement)
{
    boolean  found = false ;
    LinkedNode<E>  currentNode = this.head() ;
    while ( currentNode != null && ! found ) {
        if ( currentNode.element().equals(anElement) ) {
            found = true ;
        }
        currentNode = currentNode.next() ;
    }
    return  found ;
}
```

LinkedSet: 내용 알아보기

```
// 내용 알아보기
public   E   any ()
{
    if ( this.isEmpty() ) {
        return   null ;
    }
    else {
        return   this.head().element() ;
    }
}
```

LinkedSet: add()

```
// 내용 바꾸기
public boolean  add (E anElement)
{
    if ( this.isFull() ) {
        return  false ;
    }
    else {
        if ( ! this.doesContain(anElement) ) {
            LinkedNode<E> nodeForAdd = new LinkedNode<E>() ;
            nodeForAdd.setElement (anElement) ;
            nodeForAdd.setNext (this.head()) ;
            /* 또는 위의 3 줄을, 이렇게 한 줄로:
                LinkedNode<E> nodeForAdd =
                    new LinkedNode<E>(anElement, this.head()) ;
            */
            this.setHead (nodeForAdd) ;
            this.setSize (this.size()+1) ;
            return  true ;
        }
        else {
            return  false ;
        }
    }
}
```

LinkedSet: removeAny()

```
// 내용 바꾸기
.....
public E  removeAny()
{
    if ( this.isEmpty() ) {
        return  null ;
    }
    else {
        E  removedElement = this.head().element()  ;
        this.setHead (this.head().next()) ;
        this.setSize (this.size()-1) ;
        return  removedElement ;
    }
}
```

LinkedSet: remove() [1]

```
// 내용 바꾸기
.....
public boolean  remove (E anElement)
{
        if ( this.isEmpty() ) {
                return  false ;
        }
        else {
                // 첫 번째 단계: 삭제할 위치 찾기
                LinkedNode<E>  previousNode = null ;
                LinkedNode<E>  currentNode = this.head() ;
                boolean  found = false ;
                while ( currentNode != null && !found ) {
                        if ( currentNode.element().equals(anElement) ) {
                                found = true ;
                        }
                        else {
                                previousNode = currentNode ;
                                currentNode = currentNode.next() ;
                        }
                }
                // 두 번째 단계: 삭제하기
                if ( ! found ) {
                        return  false ;
                }
                else {
                        if ( currentNode == this.head() ) {
                                this.setHead (this.head().next()) ;
                        }
                        else {
                                previousNode.setNext (currentNode.next()) ;
                        }
                        this.setSize (this.size()-1) ;
                        return  true ;
                }
        }
}
```

LinkedSet: remove() – Step 1

```
// 내용 바꾸기
.....
public boolean  remove (E anElement)
{
        if ( this.isEmpty() ) {
                return  false ;
        }
        else {
                // 첫 번째 단계: 삭제할 위치 찾기
                LinkedNode<E>  previousNode = null ;
                LinkedNode<E>  currentNode = this.head() ;
                boolean  found = false ;
                while ( currentNode != null && !found ) {
                        if ( currentNode.element().equals(anElement) ) {
                                found = true ;
                        }
                        else {
                                previousNode = currentNode ;
                                currentNode = currentNode.next() ;
                        }
                }
                // 두 번째 단계: 삭제하기
                if ( ! found ) {
                        return  false ;
                }
                else {
                        if ( currentNode == this.head() ) {
                                this.setHead (this.head().next()) ;
                        }
                        else {
                                previousNode.setNext (currentNode.next()) ;
                        }
                        this.setSize (this.size()-1) ;
                        return  true ;
                }
        }
}
```

LinkedSet: remove() – Step 2

```
// 내용 바꾸기
.....
public boolean  remove (E anElement)
{
        if ( this.isEmpty() ) {
                return  false ;
        }
        else {
                // 첫 번째 단계: 삭제할 위치 찾기
                LinkedNode<E>  previousNode = null ;
                LinkedNode<E>  currentNode = this.head() ;
                boolean  found = false ;
                while ( currentNode != null && !found ) {
                        if ( currentNode.element().equals(anElement) ) {
                                found = true ;
                        }
                        else {
                                previousNode = currentNode ;
                                currentNode = currentNode.next() ;
                        }
                }
                // 두 번째 단계: 삭제하기
                if ( ! found ) {
                        return  false ;
                }
                else {
                        if ( currentNode == this.head() ) {
                                this.setHead (this.head().next()) ;
                        }
                        else {
                                previousNode.setNext (currentNode.next()) ;
                        }
                        this.setSize (this.size()-1) ;
                        return  true ;
                }
        }
}
```

LinkedSet: clear()

```
// 내용 바꾸기
.....

public void  clear()
{
    this.setSize (0) ;
    this.setHead (null) ;
}
```

End of

"Array Set, Linked Set"