Lab Exercise #7

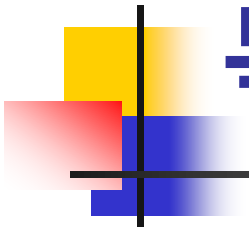
배열을 이용한 리스트 구현 실습

2018년도 2학기

컴퓨터프로그래밍2

김 영 국

충남대학교 컴퓨터공학과



목차

- 실습
  - List 인터페이스 만들기
  - ArrayList 구현 실습
- 과제
  - 가변길이 ArrayList 만들기
  - ArrayList 예외처리

실습7-1. List 인터페이스 만들기(1)

- ArrayList 구현에 앞서 제네릭 기반 List 인터페이스를 작성해보자.
- List 인터페이스는 다음과 같은 메소드를 가지고 있다.
  - public boolean add(int index, E element);
  - public boolean addFirst(E element);
  - public boolean addLast(E elemen);
  - public E remove(int index);
  - public E removeFirst();
  - public E removeLast();
  - public E get(int index);
  - public int indexOf(E element);
  - public int size();
  - public boolean isEmpty()
  - public E set()

실습7-1. List 인터페이스 만들기(2)

- 코드와 해당 메소드의 설명은 다음과 같다.

```
/**
 * 리스트 구현에 사용될 인터페이스
 * @param <E>
 */
public interface List<E> {

    /**
     * 해당 index에 element를 삽입
     * @param index
     * @param element
     * @return true
     */
    public boolean add(int index, E element);

    /**
     * 리스트 첫번째에 element를 삽입
     * @param element
     * @return true
     */
    public boolean addFirst(E element);

    /**
     * 리스트 마지막에 element를 삽입
     * @param element
     * @return true
     */
    public boolean addLast(E element);
```

```
    /**
     * 해당 index의 원소를 리턴 후에 제거
     * @param index
     * @return 해당 index의 원소
     */
    public E remove(int index);

    /**
     * 리스트 첫번째의 원소를 리턴 후 제거
     * @return 리스트의 첫번째 원소
     */
    public E removeFirst();

    /**
     * 리스트의 마지막 번째 원소를 리턴 후 제거
     * @return 리스트의 마지막 원소
     */
    public E removeLast();

    /**
     * 리스트의 index의 원소를 element로 변경.
     * @param index
     * @param element
     * @return 변경되기 전의 리스트의 원소
     */
    public E set(int index, E element);
```

```
    /**
     * 해당 index에 있는 값을 리턴
     * @param index
     * @return 리스트의 해당하는 index의 원소
     */
    public E get(int index);

    /**
     * element에 해당하는 값이 있으면
     * 그 값이 발견되는 첫번째 인덱스 값을 리턴
     * 없으면 -1을 리턴
     * @param element
     * @return element가 발견되는 첫번째 인덱스(없으면 -1을 리턴)
     */
    public int indexOf(E element);

    /**
     * 리스트의 크기를 리턴
     * @return 리스트의 크기
     */
    public int size();

    /**
     * 리스트가 비어있는지 검사
     * @return 배열에 원소가 없으면 true
     */
    public boolean isEmpty();
}
```

실습7-2. ArrayList 구현 실습(1)

- ArrayList를 구현해보자.
  - ArrayList<E>는 앞서 만들었던 List<E>를 구현한다.

```
public class ArrayList<E> implements List<E>
```

  - ArrayList<E>의 필드와 생성자는 다음과 같이 만들어보자.

```
private static final int DEFAULT_CAPACITY = 10;
private Object[] elementData;
private int size = 0;

public ArrayList() {
    this.elementData = new Object[DEFAULT_CAPACITY];
}
public ArrayList(int size) {
    this.elementData = new Object[size];
}
```

- List<E>의 메소드를 오버라이드만 해둔 상태에서의 테스트 코드

```
public class ArrayListTest {
    public static void main(String[] args) {
        List<Integer> list1 = new ArrayList<>();
        List<Integer> list2 = new ArrayList<>(50);
    }
}
```

실습7-2. ArrayList 구현 실습(2)

- ArrayList에 원소를 추가하는 메소드인 addLast(), add(), addFrist()와 ArrayList의 원소들을 출력하는 toString() 메소드를 작성해보자.
  - toString() 메소드

```
@Override
public String toString() {
    StringBuilder str = new StringBuilder("[");
    for (int i = 0; i < size; i++) {
        str.append(elementData[i]);
        if (i < size - 1)
            str.append(", ");
    }
    str.append("]");
    return new String(str);
}
```

  - addLast() 메소드

```
@Override
public boolean addLast(E element) {
    this.elementData[size++] = element;
    return true;
}
```

실습7-2. ArrayList 구현 실습(3)

- addLast() 메소드의 테스트 코드와 결과

```
public class ArrayListTest {
    public static void main(String[] args) {
        List<Integer> list = new ArrayList<>();
        list.addLast(10);
        list.addLast(20);
        list.addLast(30);
        System.out.println(list);
    }
}

[10, 20, 30]
```

- 추가적으로 원소를 원하는 인덱스에 추가하는 add()와 리스트 맨 앞에 추가하는 addFirst()를 구현해보자. addFirst()는 내부적으로 add() 메소드 호출을 통해 구현한다.
  - addFirst() 메소드

```
@Override
public boolean addFirst(E element) {
    return add(0, element);
}
```

실습7-2. ArrayList 구현 실습(4)

- add() 메소드

```
@Override
public boolean add(int index, E element) {
    for(int i = size - 1; i >= index; i--) {
        elementData[i+ 1] = elementData[i];
    }

    elementData[index] = element;
    size++;
    return true;
}
```

- add(), addFirst()의 테스트 코드와 결과

```
public class ArrayListTest {
    public static void main(String[] args) {
        List<Integer> list = new ArrayList<>();
        list.addLast(10);
        list.addLast(20);
        list.addLast(30);
        list.add(1, 15);
        list.addFirst(-10);
        System.out.println(list);
    }
}

[-10, 10, 15, 20, 30]
```

실습7-2. ArrayList 구현 실습(5)

- 인덱스에 해당하는 값을 반환하고 제거하는 remove() 메소드와 리스트의 가장 첫번째 원소를 반환하고 제거하는 removeFirst() 메소드, 가장 마지막 원소를 반환하고 제거하는 removeLast() 메소드를 작성해보자. 그후 원소를 제거하지 않고 반환하는 get() 메소드를 작성해보자.
  -
```
@Override
public E remove(int index) {
    Object obj = this.elementData[index];

    for(int i = index + 1; i <= size - 1; i++) {
        this.elementData[i - 1] = this.elementData[i];
    }
    size--;
    elementData[size] = null;
    return (E)obj;
}
```
  - removeFirst() 메소드
```
@Override
public E removeFirst() {
    return this.remove(0);
}
```

실습7-2. ArrayList 구현 실습(6)

- removeLast() 메소드

```
@Override
public E removeLast() {
    return this.remove(size - 1);
}
```

- get() 메소드

```
@Override
public E get(int index) {
    return (E)this.elementData[index];
}
```

- remove(), removeFirst(), removeLast(), get() 테스트 코드와 결과

```
public class ArrayListTest {
    public static void main(String[] args) {
        List<Integer> list = new ArrayList<>();
        list.addLast(10);
        list.addLast(20);
        list.addLast(30);
        list.add(1, 15);
        list.addFirst(-10);
        System.out.println(list.remove(2));
        System.out.println(list.removeLast());
        System.out.println(list.removeFirst());
        System.out.println(list.get(0));
        System.out.println(list);
    }
}
```

```
15
30
-10
10
[10, 20]
```

실습7-2. ArrayList 구현 실습(7)

- ArrayList에 저장된 원소들의 수를 반환하는 size() 메소드와 인자로 전달된 값의 가장 첫번째 인덱스를 반환하는 indexOf() 메소드를 작성해 보자.
- indexOf() 메소드는 해당하는 값을 찾지 못했을 경우 -1를 반환한다.
  - size() 메소드

```
@Override
public int size() {
    return this.size;
}
```

  - indexOf() 메소드

```
@Override
public int indexOf(E element) {
    for(int i = 0; i < size; i ++) {
        if(element.equals(elementData[i]))
            return i;
    }
    return -1;
}
```

실습7-2. ArrayList 구현 실습(8)

- size(), indexOf() 테스트 코드와 결과

```
public class ArrayListTest {
    public static void main(String[] args) {
        List<Integer> list = new ArrayList<>();
        list.addLast(10);
        list.addLast(20);
        list.addLast(30);
        list.add(1, 15);
        list.addFirst(-10);
        System.out.println(list);
        System.out.println("size: " + list.size());
        System.out.println("20의 index: " + list.indexOf(20));
        System.out.println("100의 index: " + list.indexOf(100));
    }
}
```

```
[-10, 10, 15, 20, 30]
size: 5
20의 index: 3
100의 index: -1
```

과제7-1. 가변 길이 ArrayList 만들기(1)

- 실습에서 만든 ArrayList는 내부에서 사용하는 배열의 길이보다 많은 원소를 추가하면 런타임 에러가 발생한다.

```
public class ArrayListTest {
    public static void main(String[] args) {
        List<Integer> list = new ArrayList<>(3);
        list.addLast(10);
        list.addLast(20);
        list.addLast(30);
        list.add(1, 15);
        list.addFirst(-10);
        System.out.println(list);
    }
}
```

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 3
        at ArrayList.ArrayList.add(ArrayList.java:26)
        at ArrayList.ArrayListTest.main(ArrayListTest.java:10)
```

과제7-1. 가변 길이 ArrayList 만들기(2)

- 이를 해결하기 위해 내부 배열의 길이를 두배 늘려주는 resize() 메소드를 작성하고, 배열이 다 차있을 때 새로운 원소가 추가되면 resize() 메소드를 호출하여 실습으로 작성된 ArrayList를 가변 길이를 가지게 만들어보자.

```
private void resize() {
    // 작성하시오.
}
```

- 테스트 코드와 결과

```
public class ArrayListTest {
    public static void main(String[] args) {
        List<Integer> list = new ArrayList<>(3);
        for(int i = 0; i < 20; i++)
            list.addLast(i);
        System.out.println(list);
    }
}
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```

과제7-2. ArrayList 예외처리(1)

- 실습에서 만든 ArrayList에는 다음과 같은 문제점이 있다.

```java
public class ArrayListTest {
    public static void main(String[] args) {
        List<Integer> list = new ArrayList<>(3);
        list.add(2, 30); // 잘못된 위치에 삽입
        System.out.println("List: " + list+ "\n" + "size: " + list.size());
    }
}
```

```
List: [null]
size: 1
```

- 위 경우 잘못된 위치에 원소를 삽입하였기 때문에 size는 늘었지만 내부 배열에 공백이 생겨서 null 값이 출력된다.
- 이 문제를 add 메소드를 수정하여서 해결하라. (Exception을 사용하지 않아도 된다.)

```
List: []
size: 0
```

과제7-2. ArrayList 예외처리(2)

- remove() 메소드의 문제점은 다음과 같다.

```
public class ArrayListTest {
    public static void main(String[] args) {
        List<Integer> list = new ArrayList<>(3);
        System.out.println(list.remove(0)); // 잘못된 호출
    }
}
```

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: -1
        at ArrayList.ArrayList.remove(ArrayList.java:75)
        at ArrayList.ArrayListTest.main(ArrayListTest.java:7)
```

- 이 경우 비어 있는 리스트이기 때문에 remove() 메소드는 null 값을 반환해야 하나 내부 구현 문제로 ArrayIndexOutOfBoundsException이 발생하였다.
- isEmpty() 메소드를 구현해서 이를 통해 리스트가 비어 있을 경우 remove() 메소드가 null 값을 반환하도록 수정해보자.
- 추가로 잘못된 인자가 전달되면 IndexOutOfBoundsException이 발생하도록 remove() 메소드를 수정해보자.

과제 제출 및 기한

- 제출 방법
  - 사이버캠퍼스를 통하여 제출
  - 소스코드를 제출
- 제출 기한
  - 이번 주 토요일(11/3) 자정