객체지향개발론및실습

Laboratory 8. 반복자 패턴

1. 목적

- 복합데이터를 유지하는 객체의 구현 방법을 노출시키지 않으면서도 그 객체에 포함되어 있는 모든 데이터에 차례대로 접근할 수 있도록 해주는 **반복자**(iterator) 패턴을 실습한다.

2. 실습. 배열 기반 리스트와 이중 연결구조 기반 리스트에 대한 반복자

- el 사이트를 통해 배열 기반 비정렬 리스트와 이중 연결구조 기반 비정렬 리스트가 포함된 프로젝트가 제공된다.
- 자바
  - 보통 자바에서 반복자는 **`java.util.Iterator<T>`**를 상속받아 정의한다.
  - 이 실습에서는 더 강건한 **`java.util.ListIterator<T>`**를 상속받아 정의한다. 여기서 강건하다는 것은 반복을 하면서 데이터를 추가, 삭제, 변경할 수 있다는 것을 말한다.
  - 두 자료구조의 포함된 반복자를 완성하시오.
- C++와 파이썬: C++와 파이썬에서 반복자를 구현하는 방법은 자바와 다르지만 이 실습에서 자바와 동일한 형태의 반복자를 구현해보고자 한다.
  - C++와 파이썬의 경우에도 자바의 **`java.util.ListIterator<T>`**와 동일하게 동작하는 것을 구현해 본다.
  - C++는 eclipse 프로젝트에 googletest를 이용한 테스트 프로그램이 포함되어 있다.

2.1 구현해야 하는 메소드에 대한 설명

- 두 반복자 모두 커서 개념을 이용한다. 배열 기반 리스트에서는 커서는 색인값이며, 연결구조에서는 노드를 가리키는 참조가 커서가 된다. 배열 기반 리스트에서 커서는 0으로 초기화되며, 연결구조에서는 첫 번째 노드 **`head`**로 초기화된다,
- **`next`**: 다음 요소를 반환하고, 커서를 다음 위치로 이동함. 예) [2,3,5]에서 **`next`**를 연속적으로 호출하면 2, 3, 5를 차례로 반환한다. 더 이상 제공할 요소가 없으면 **`NoSuchElementException`**을 발생한다.
- **`prev`**: 이전 요소를 반환하고, 커서를 이전 위치로 이동함. [2,3,5]에서 **`next`**가 5를 반환한 후에 연속하여 **`prev`**를 부르면 5, 3, 2를 반환하여야 한다. **`next`**, **`prev`**를 번걸아 호출하면 계속 동일 값을 주어야 한다. 더 이상 제공할 요소가 없으면 **`NoSuchElementException`**을 발생한다.
- **`nextIndex`**: **`next`** 메소드가 호출하면 반환할 요소의 색인을 반환함. 커서가 맨 뒤에 있으면 리스트의 크기를 반환함
- **`previousIndex`**: **`prev`** 메소드가 호출하면 반환할 요소의 색인을 반환함. 커서가 맨 앞에 있으면 -1을 반환함

- **set**: 이전 **next()**나 **previous()** 호출에서 반환된 요소를 교체한다. **next()**나 **previous()** 호출 이후 **add()**나 **remove()**가 호출되었으면 실행하지 않고 예외(**IllegalStateException**)를 발생해야 한다.
  - 예) [2, 3, 5]에서 **next()** 호출로 첫 번째 2를 받은 후 **set(4)**를 호출하면 [4, 3, 5]가 되어야 한다.
- **add**: **next()**를 호출하였을 때 반환할 요소 앞에 추가한다. **add** 이후 **next()**의 호출은 **add** 하지 않은 경우와 차이가 없어야 한다.
  - 예) [2, 3, 5]에서 **next()** 호출로 첫 번째 2를 받은 후 **add(4)**를 호출하면 [2, 4, 3, 5]가 되어야 하며, 그다음 **next()**가 호출되면 3을 반환하여야 한다.
- **remove**: 이전 **next()**나 **previous()** 호출에서 반환된 요소를 제거한다. **next()**나 **previous()** 호출 이후 한 번만 실행 가능하다. **next()**나 **previous()** 호출 이후 **add()**나 **remove()**가 호출되면 실행하지 않고 예외(**IllegalStateException**)를 발생해야 한다.
  - 예) [2, 3, 5]에서 **next()** 호출로 첫 번째 2를 받은 후 **remove()**를 호출하면 [3, 5]가 되어야 하며, 그다음 **next()**가 호출되면 3을 반환하여야 한다.
- 연결구조에서는 커서를 색인으로 사용하지 않지만 **nextIndex**, **previousIndex**의 구현을 위해서는 배열 기반 리스트와 유사하게 색인을 유지해야 한다.