객체지향개발론및실습

Laboratory 7. Iterator 패턴과 Composite 패턴

1 목적

- 복합데이터를 유지하는 객체의 구현 방법을 노출시키지 않으면서도 그 객체에 포함되어 있는 모든 데이터에 차례대로 접근할 수 있도록 해주는 반복자 패턴을 실습함
- 객체를 트리구조로 구성할 수 있도록 해주며, 클라이언트가 단말이나 중간 노드를 동일하게 취급할 수 있도록 해주는 Composite 패턴을 실습함

2 실습 1. 연결구조에 대한 반복자

- el 사이트에서 2017-Pattern-Iterator-LinkedList.zip을 다운받아 활용하십시오.
- 보통 자바에서 반복자는 `java.util.Iterator<T>`를 상속받아 정의한다.
- 이 실습에서는 보다 강건한 `java.util.ListIterator<T>`를 상속받아 정의한다.
- hasNext, next, remove, set, add 5개의 메소드를 완성하시오. 나머지 메소드는 UnsupportedOperationException을 발생함
- set: 이전 next() 호출에서 반환된 요소를 교체함. next() 이후 add()나 remove()가 호출되면 실행되지 않고 예외(illegalStateException)가 발생되어야 함
  - 예) [2, 3, 5]에서 next() 호출로 첫 번째 2를 받은 후 set(4)를 호출하면 [4, 3, 5]가 되어야 함
- add: next()를 호출하였을 때 반환할 요소 앞에 추가함. add 이후 next()의 호출은 add 하지 않은 경우와 차이가 없어야 함
  - 예) [2, 3, 5]에서 next() 호출로 첫 번째 2를 받은 후 add(4)를 호출하면 [2, 4, 3, 5]가 되어야 함
- remove: 이전 next() 호출에서 반환된 요소를 제거함 한 번 실행가능하며, next() 이후 add()가 호출되면 실행되지 않고 예외(illegalStateException)가 발생되어야 함
  - 예) [2, 3, 5]에서 next() 호출로 첫 번째 2를 받은 후 remove()를 호출하면 [3, 5]가 되어야 함

3 실습 2. 파일, 폴더

- 이론 수업에서 실습한 Composite 패턴 소스에서 단말노드를 Leaf, 중간노드를 NonLeaf, 이들을 일반화한 노드를 TreeNode라고 이름을 바꾼 후에 이들을 이용하여 파일시스템의 특정 폴더를 접근하였을 때, 그 폴더 아래에 있는 모든 폴더와 디렉토리를 트리 형태로 구성하고, 이들을 JavaFX의 TreeView를 이용하여 화면에 표시함
- 기본적인 소스는 2017-Pattern-Composite-Folder.zip을 다운받아 활용함
- JavaFX에서는 javafx.stage.DirectoryChooser를 이용하여 폴더를 선택할 수 있도록 해줌. 참고로 파일을 선택할 경우에는 FileChooser를 사용함
- DirectoryChooser를 이용하여 폴더를 선택하면 File의 listFiles() 메소드를 이용하여 해당 폴더에 있는 모든 파일(폴더 포함)을 얻을 수 있음. 이를 이용하여 반복적으로 NonLeaf와 Leaf 노드를 이용하여 선택한 폴더와 그 아래의 모든 파일을 나타내는 트리 구조를 구성함. 이 때 NonLeaf와 Leaf 노드들은 해당 폴더와 파일 이름을 가지도록 함. 이를 위한 다음 메소드를 완성하시오.
  - `NonLeaf constructFolderTree(File selectedDirectory)`

  - 이 메소드는 트리를 구성한 다음 선택한 폴더를 나타내는 NonLeaf 노드를 반환하여 줌. 즉, 반환된 NonLeaf 노드의 getName() 메소드의 호출 결과와 selectedDirectory의 getName()의 호출 결과 값은 같아야 하며, NonLeaf 노드는 해당 폴더에 있는 모든 폴더와 파일을 자식 노드로 가져야 함

- 이와 같이 구성한 트리를 화면에 보여주기 위해 JavaFX의 javafx.scene.control.TreeItem을 구성함. 이를 위한 다음 메소드를 완성하시오.
  - `TreeItem<String> constructTreeItem(NonLeaf currentFolder)`
  - 이 메소드는 주어진 NonLeaf 노드가 루트인 부분 트리를 나타내는 TreeItem을 만들어 줌. 즉, 반환되는 TreeItem은 NonLeaf의 getName()을 값을 가지고 있어야 하며, NonLeaf의 자식 노드들을 자식(이 역시 TreeItem 타입임)으로 가지고 있어야 함

- 이 경우 보여주는 순서는 우리가 원하는 순서와 다를 수 있음. 이에 LeafNode에 다음 메소드를 만들어 폴더와 파일 순으로 보여주고, 폴더 간에는 또는 파일 간에는 이름 순으로 보여주도록 함. 이 때 실제 자식 목록을 수정하지 않고 동일된 노드이지만 ArrayList에 저장된 자식 노드들의 순서가 바뀐 노드를 반환하여야 함
  - `public NonLeaf getRearranged()`

- constructTreeItem으로 구성된 뷰는 선택한 폴더 아래에 있는 모든 파일과 폴더를 포함하고 있음. 특정 확장자를 가진 파일들만 표시되도록 필터링 기능을 다음과 같이 추가하시오.
  - `TreeItem<String> constructFilteredTreeItem(NonLeaf currentFolder, FileFilter filter)`
  - 이 메소드는 constructTreeItem가 필터링 기능이 추가된다는 것을 제외하고는 동일하다. 주어진 filter에 참인 파일들만 추가되며, 하위 디렉토리에 해당 파일이 전혀 없는 경우에는 폴더 자체도 포함되지 않아야 함
  - 여기서 FileFilter는 FunctionalInterface로 함수형 프로그래밍 요소임.

```
@FunctionalInterface
interface FileFilter{
    boolean filter(String fileName);
}
```

  - 만약 확장자가 html인 파일만 필터링하고 싶으면 어떻게?

```
TreeItem<String> root = constructFilteredTreeItem(currentFolder,
   name ->{
       // 완성하시오.
   });
```