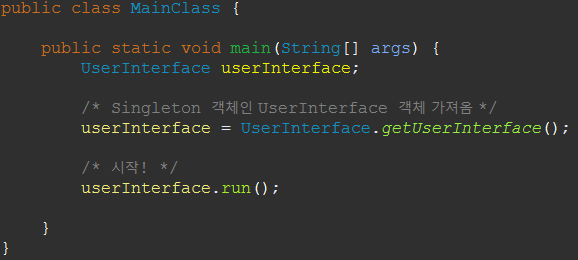
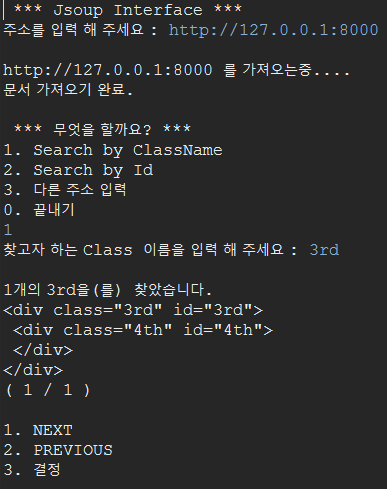
확장 기능 설명

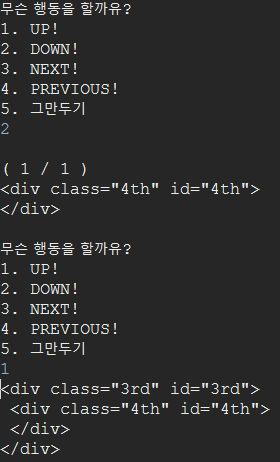
* 개요
  + 우리 팀은 Jsoup을 모르는 사람도 손 쉽게 그때그때 문서를 가져와 사용 할 수 있는 Interface를 만들어 보았다. 기존에 Jsoup을 이용해 문서를 가져와 Parsing 하려면 Jsoup.Connect( address ).get 을 이용해 문서를 가져와야 했고 얻어진 Document에 대해 필요한 함수를 찾아 입력해 주어야 했다. 우리는 이런 방법뿐만 아니라 프로그램이 한번 실행되면 그때그때 필요한 주소를 입력하고 문서를 가져 올 수 있는 UI 프로그램을 작성하였다.
  + 자세하게는 Document를 가져와서 Class명이나 ID 로 Select를 할 수 있고 얻어진 Element에 대해 부모, 자식, 형제 Element를 간단한 입력 명령어로 손쉽게 얻을 수 있는 기능을 가진 인터페이스 이다.
* 확장기능 소개
* 우리의 확장 기능은 Singleton객체 이자 Façade 객체인 UserInterface 객체를 통해 이용 할 수 있다.

\

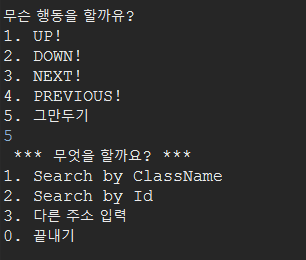
* Singleton 객체인 UserInterface를 불러와 run 함수를 통해 모든 일들을 수행 하게 된다.
* UserInterface를 실행하면 Document를 가져오고 싶은 주소, 하고자 하는 행동 을 선택 할 수 있다.



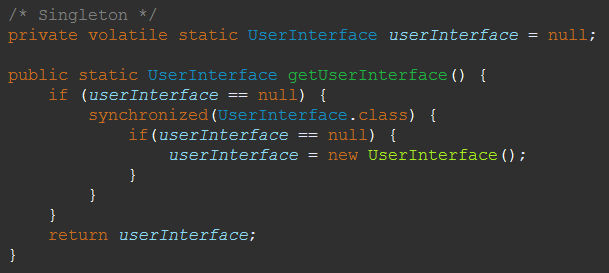
* 맨 처음에 Element를 찾을 때는 같은 속성 이름을 가진 클래스가 여러 개 존재 할 수 있으니 여러 개중에 하나를 선택 할 수 있게 하였다. 만약 유저가 하나의 Element를 선택 한다면 그 Element로부터 부모, 자식, 형제 Element들을 찾을 수 있게 된다.
  + UP(부모 찾기), DOWN(자식 찾기), NEXT(형제 찾기), PREVIOUS(형제 찾기)

 (3rd 클래스에서 - > 자식 - > 부모를 탐색 한 결과)

* 5. 그만두기를 누른다면 주소를 다시 입력하거나 끝내는 등의 행동을 할 수 있다.



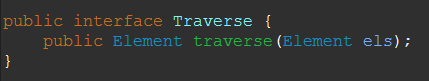
* 적용된 디자인 패턴
  + Singleton Pattern – 확장 기능을 위한 프로그램의 생명 주기를 담당하는 UserInterface 객체를 Singleton 으로 작성하였다.



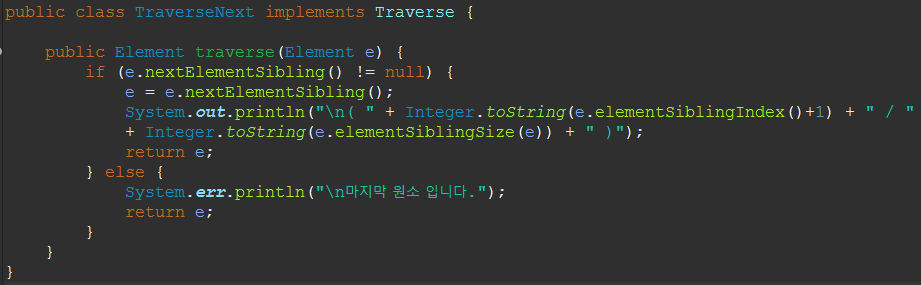
* + Façade Pattern – Main 함수 내에 userInterface의 ‘run’ method를 한번 호출 함으로써모든 확장기능을 이용 할 수 있도록 Façade Pattern을 적용하여 작성 하였다.

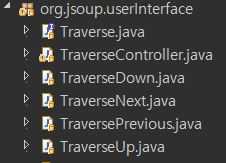


* + Strategy Pattern – 우리 프로그램은 runtime에 사용자의 입력을 받아 Element를 탐색 하는 행위를 한다. Element를 기준으로 runtime에 behavior(algorithm)가 바뀌는 상황이므로 그때 그때 필요한 알고리즘을 삽입하기 위하여 Strategy Pattern을 이용하였다.
    - Traverse 라는 Interface를 만들었다.

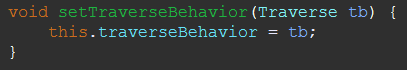


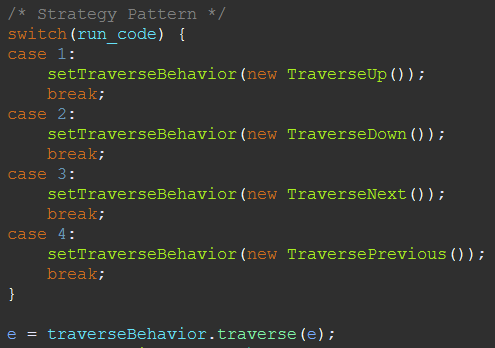
* + - Traverse 인터페이스를 구현하는 여러 알고리즘 들이 있다.





* + - TraverseController에서 SetTraverseBehavior를 통해 runtime에 알고리즘을 바꿀 수 있다.





* + - Traverse는 결국 Element에 대한 연산 과정을 지원해주는 알고리즘들의 인터페이스로써 현재는 Up, Down Next, Previous의 네 가지 동작이 있지만 향 후 Element에 관한 추가적인 연산을 지원하는 확장의 가능성을 열어 두어 알고리즘들을 Encapsulate하여 runtime에 바꿀 수 있는 Strategy Pattern을 적용하기로 하였다.