**설계패턴 텀 프로젝트**

1. Jsoup 개요

jsoup은 DOM을 사용하여 HTML, XML 문서에 저장된 데이터를 구문 분석, 추출 및 조작하도록 설계된 오픈 소스 Java 라이브러리

2. Jsoup 설계 및 구현 조사

\*설계 Overview

Jsoup 기능 접근을 위해 public Jsoup 클래스를 core public access point로 둔다.

 Jsoup은 클라이언트가 Jsoup.Connect 함수를 통해 HTTP Request와 Response를 만들어 내며 그 과정에서 해당하는 사이트의 Html 혹은 Xml을 가져와 파싱하는 과정이 있다. 유저는 Connect 함수를 통해 url을 넘겨주기만 하면 통신에 대한 과정을 알 필요 없이 해당 url에 대한 결과인 document를 얻어 낼 수 있다. 유저가 Connect라는 함수만을 불러 document를 얻어낸다는 점에서 Facade 패턴이 적용되었다고 판단했다.

 Facade 패턴은 내부의 복잡한 과정을 알 필요 없고, 제공된 간단한 인터페이스를 통해 내부 작업을 수행 할 수 있게 만들어진 패턴을 의미하며 이 다음부터는 Facade 안에 숨겨진 세부적인 코드에 대해 적용된 설계 패턴을 소개하려 한다.

\*적용된 설계 패턴 소개 - 이름, 클다, 판단 근거, 소스코드와 매핑

i. ConstrainableInputStream.java – wrap method

- Decorator Pattern (BufferedInputStream을 extends)

최초의 Input Stream 객체 만들고 여러가지 Input Stream으로 Decorate 하면서 최종 결과물을 만들어 냄

       ii. NodeVisitor.java, NodeTraversor.java

- Visitor Pattern, Adapter Pattern

NodeVisitor Interface가 Visitor에 매칭 되고 Accumulator class, CleanVisitor class등이 Concrete visitor에 매칭.

Node class가 패턴의 Element, Traverse method가 패턴의 accept method에 대응, Element class와 LeafNode class는 Concrete Element이다.

Node자체가 추가될 염려가 없으므로 visitor에서 Elements가 증가함에 따라, 수정하게 될 걱정을 하지 않아도 된다.

NodeVisitor에서 visit이 선언되어 있지 않고, 각각의 비지터는 visit을 구현하지 않고 Node Traversor에서 visit에 해당하는 traverse를 구현해 놓았다.

traverse하는 동작이 런타임에 바뀔 수 있으므로, (cleaning, 등)

Adapter 패턴으로는 Client: NodeTraversor, Target: NodeVisitor, Adapter: Accumulator(CleaningVisitor 등도 포함), Adaptee: Element이다.

\* 특이사항

- Concrete visitor를 호출 시에 생성하여 사용 nodes>Element>wholeText(), text()

- Visitor와 Adapter패턴이 함께 적용되어 Accumulator 등 Concrete Visitor들이 Concrete Element인 Element를 직접 알고있다.

텍스트, 지도이(가) 표시된 사진

자동 생성된 설명

iii. TreeBuilder.java

- Builder Pattern

TreeBuilder에서 만드는 Part들( ex) Element, Token 등 )을 만들어준다. Client는 Part들에 대해선 모르지만 TreeBuilder를 통해 완성된 Tree를 볼 수 있다.

Director: Parser

abstract Builder 클래스: TreeBuilder

Concrete Builder: HtmlTreeBuilder, XmlTreeBuilder

Client: Document

스크린샷이(가) 표시된 사진

자동 생성된 설명

iV. Evaluator.java

- Decorator Pattern

텍스트, 지도이(가) 표시된 사진

자동 생성된 설명

V. ChangeNotifyingArrayList.java

- Observer Pattern

abstract Observer 클래스: ChangeNotifyingArrayList

Concrete Observer: NodeList

Subject: Element

텍스트이(가) 표시된 사진

자동 생성된 설명

Vi. TokeniserState.java, HtmlTreeBuilderState.java

- State Pattern

State interface를 따로 두지 않고, State 클래스 안에 handleReq메서드들과 enum으로 State들을 같이 정의해놨다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

예상되는 문제점: Document는 클라이언트 인터페이스로 작용하면서, 빌더패턴에서의 Product로도 지정된다.

Product는 xml document, html document

                      v.         Node.java

-      Composite 패턴

-      Safety 보다 transparency에 중점을 둠

3. 디자인 패턴 개선 내용

4. 확장기능

도큐먼트에 파서가 있음.

첫 도큐먼트 파싱 이후에는 Jsoup 파서 인터페이스가 아닌 파서 인터페이스를 사용.

xml파서는 별도로 도큐먼트 내에 있는 파서의 xmlParse함수를 호출하여, xml 트리빌더로 교체해야함.