**Design pattern**

**Jsoup pattern analyse & improvement**



Subject : Design pattern (01분반)

Professor : 이 찬 근 교수님

Team name : Team02

Members : 20145523 김상헌

20140786 조장연

20145720 임기찬

20140165 천영재

1. 개요

Jsoup : java HTML Parser. 로서, 실제 HTML과 작용하는 오픈소스 API입니다.

기본적으로는 parse 와 extract 등의 역할을 수행합니다.

우리 프로젝트는 이 오픈소스의 Design pattern 을 확인하고, 이 패턴을 통해서 어떻게 구성을 바꾸고, 개선해서 더 나은 방향으로 진행할 수 있는지에 대한 프로젝트를 진행하였습니다.

1. Jsoup 설계 및 구현 조사

1차적으로 Jsoup의 패키지를 받아 그 안에 무슨 구조로 이루어져 있는지에 대해서 분석, 및 구조를 정리하여 문서화를 진행.

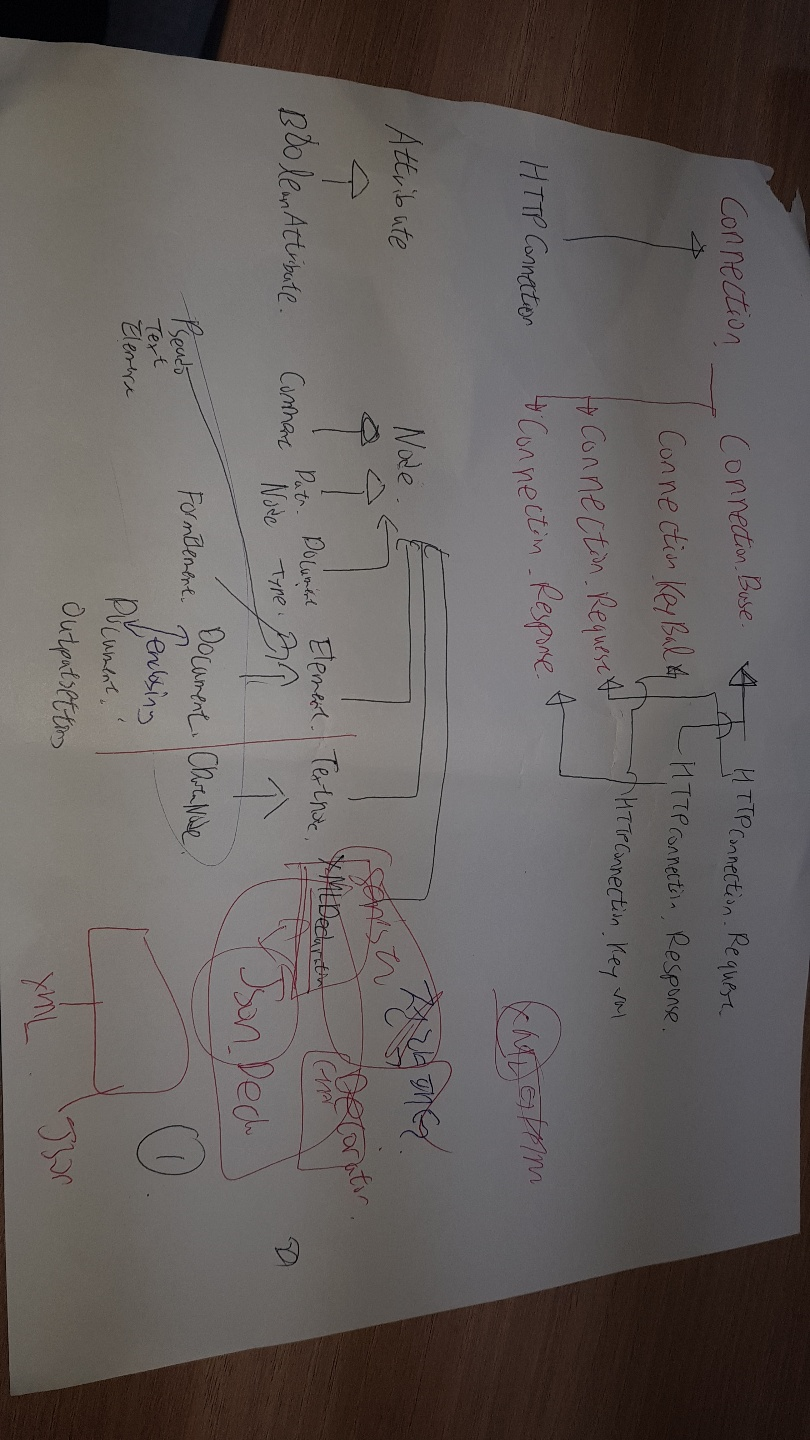
<https://jsoup.org/> 의 문서를 활용하여 진행하였으며, 각각의 클래스를 실제로 보며 배운 패턴과 관련해서 분석을 진행하였고, Jsoup 안의 각각의 클래스에 적용된 패턴을 분석해 보았습니다.

Parser는 Builder Pattern

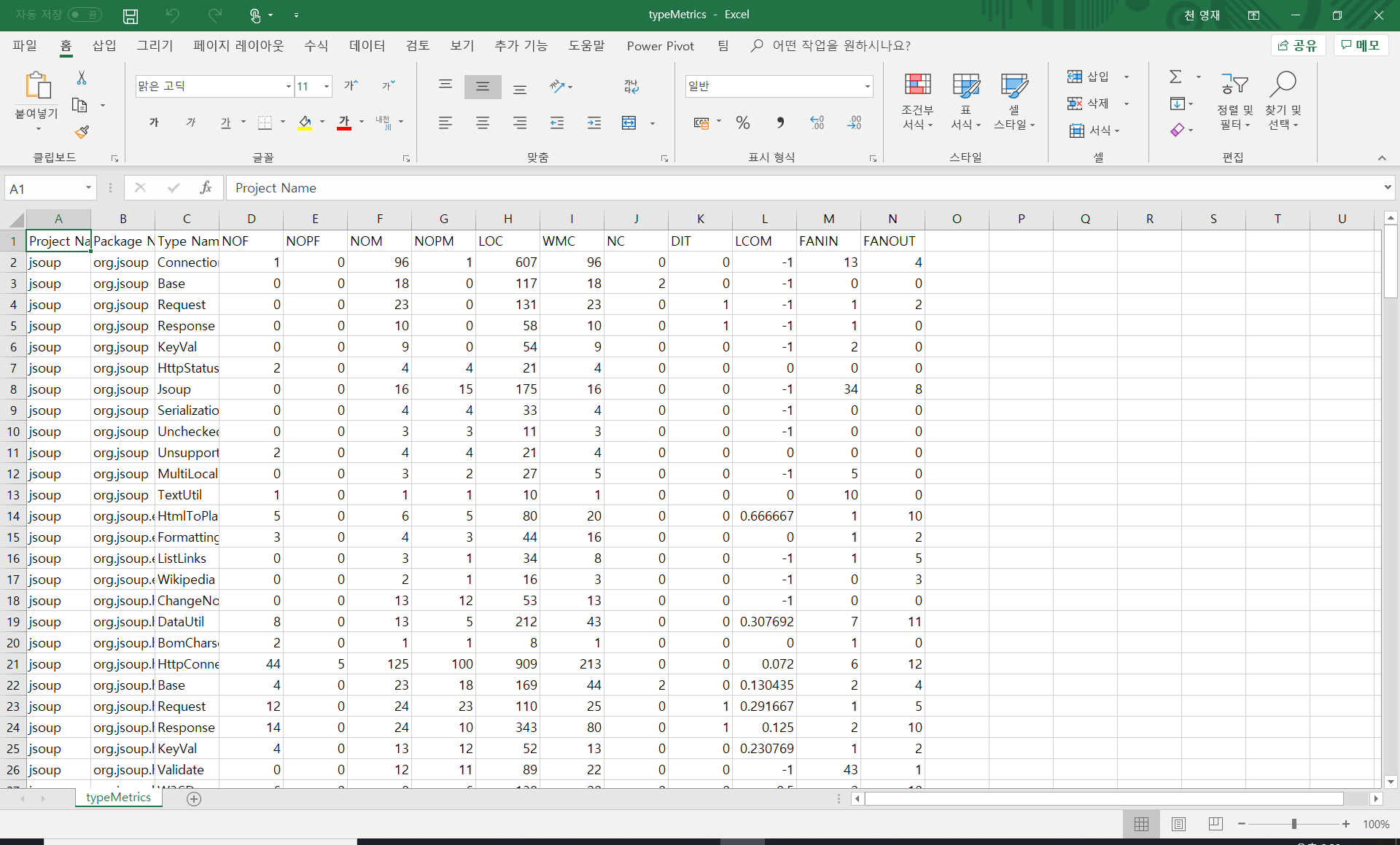
Node는 Decorator Pattern

Connection, Evaluator는 Strategy Pattern

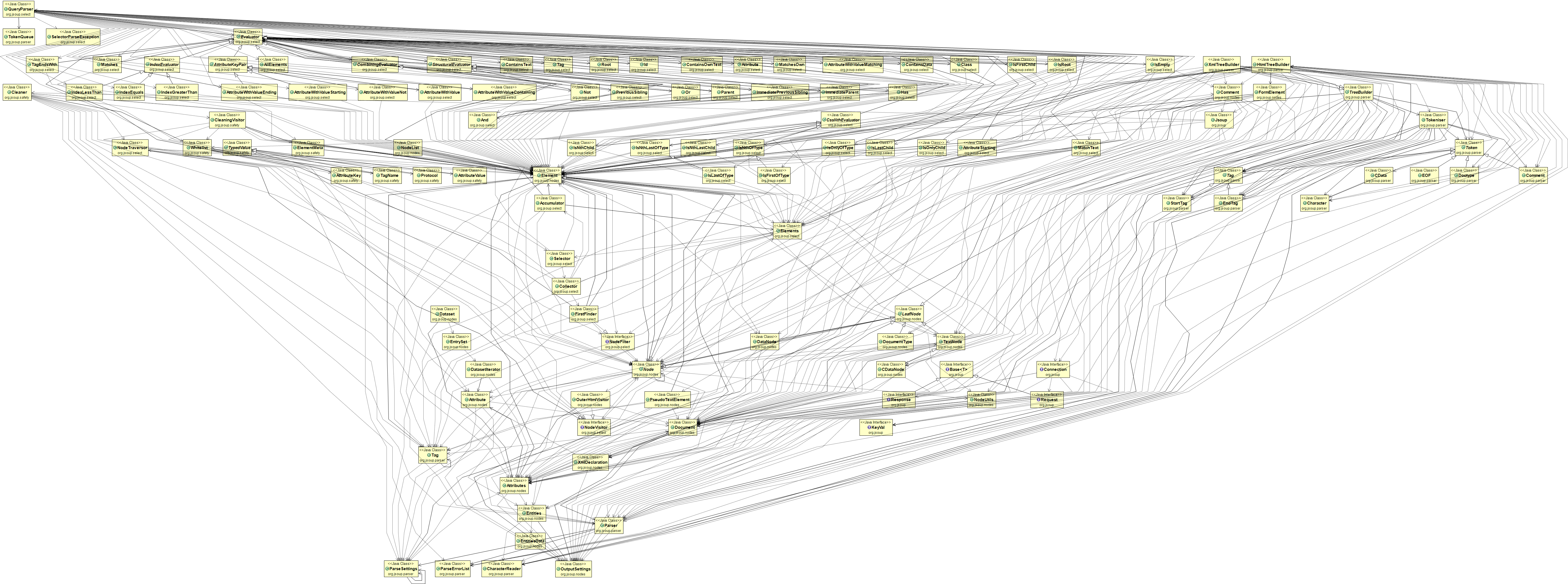
와 같이 결론을 내렸습니다.



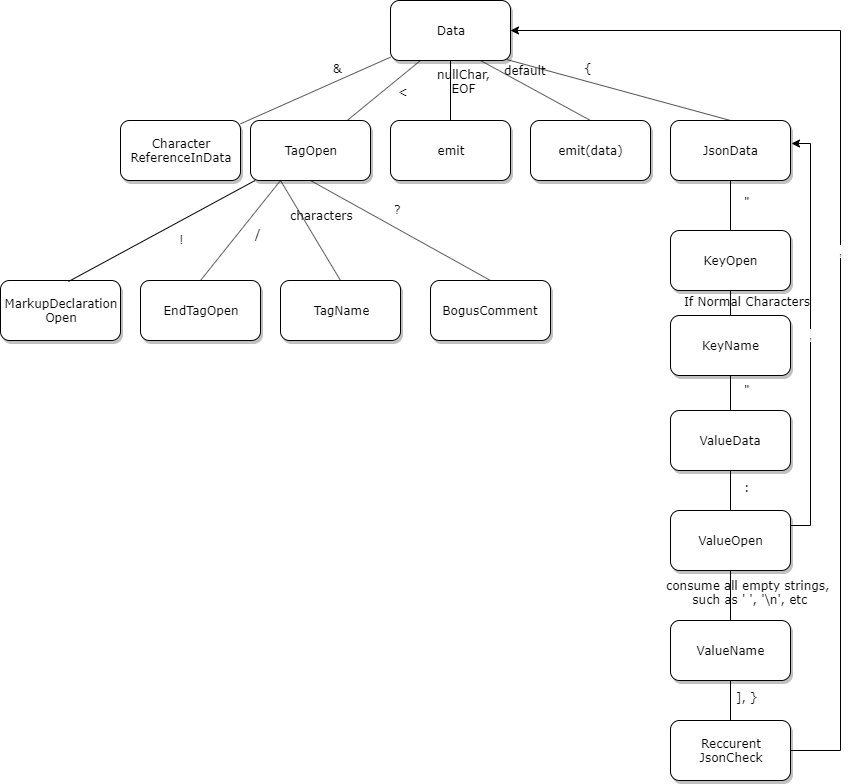
그 이후에, 저희가 수업시간에 배운 Lcom에 관해서 클래스간의 구조를 살펴보기 위해서 툴을 사용해서 적용시켰습니다. 적용시킨 툴은 오픈소스로 나와있는, <https://github.com/tushartushar/DesigniteJava> 의 jar파일로 적용시켜 csv파일로 분석해서 뽑아냈으며, 결과는 아래와 같았습니다.



이를통해 Lcom값과 code smell을 뽑아냈고, 이를 더 정확히 분석하기 위해서 클래스간의 의존성을 분석했습니다.



또한 Jsoup관련 확장을 저희 팀이 준비하였기 떄문에, Jsoup관련해서 treebuilder이후에 treebuilder와 연계해서 진행하는 tokeniser와 tokeniserstate에 대해서 분석을 진행하였습니다.



tokeniserstate.java 안에는 파일을 어떤 state로 읽고 진행하는지에 대해서 분석한 결과입니다.

Json에 대한 부분이 오면 Jsondata로 state가 빠져서 진행되는 형식입니다.

분석결과에 대해서 value와 키에대해서 확장을 진행했습니다.

1. 설계 overview, 팀이 수행한 기능 확장과 설계개선.

이를 통해서 무엇을 바꿀지를 파악하려고 했고, json에 대한 부분의 확장을 진행하였습니다. 현재 적용 되어있는 경우는 json에 대한 부분이 빠져있었기에 좀더 효율적으로 유저의 방향에 맞는 경우로 개선하기 위해서는 추가하는 것이 나쁘지 않다고 파악하여 방향을 잡았습니다.

1. 팀이 수행한 기능 확장과 설계 개선

Jsoup관련 기능을 추가하기 위해서 xmltreebuilder의 패턴을 적용해서 jsontreebuilder를 작성하였습니다. json에서는 xml과 달리, endtag에 대한 내용이 없다는 것이 가장 큰 차이점 이였습니다. 이부분을 컨트롤 하기위해서 json에 관련된 endtag에 관한 내용을 추가하였고, 이부분이 가장 큰 차이점이라고 생각합니다. 처음에 태그가 열리는 방식이 다르고, 또한 endtag로 빠지는 부분을 컨트롤하기 위해서 tonkeniser.java와 tokeniserstate.java에서 관련된 token의 태그와 token에 state 관련된 부분을 추가하였고, 이를통해 테스트를 진행하였습니다.

1. 확장된 기능 요약

Json으로 이루어진 string과 파일을 json형식으로 읽을 수 있도록 기능 확장을 진행하였습니다.

1. 상세한 변경 내용 설명 및 기존 설계/코드와의 비교

Json을 읽기 위한 Json 관련된 파일 읽고, tree를 짜주는 Jsontreebuilder는 xmltreebuilder처럼 builder/stratagy패턴을 적용하여 진행하였고, XML에서 이름이 정해지지않는 tag를 인식하는 것과같이, json object에서의 key값을 이처럼읽을 수 있도록하였습니다. 그리고 endtag에 관련된 조건이 xml과 다르기 때문에 endtag에관한 조건을 처리할 수 있도록 진행하였습니다.

예를 들어서, endtag에 관련된 부분을 xml은 </~>라는 형식으로 진행되지만, json의 경우 ], }와 같은 부분을, tag에 대한 이름정보 없이도 닫아줘야된다는 차이점이 있는데, 이를 핸들링하는 부분을 중점적으로 진행하였습니다.

또한, tokeniserstate.java에서 state에 대한 부분을 json에 맞게 수정을 진행하였습니다. 마찬가지로 end, json object 추가, key값과 value값을 따로 적용시키기 위해서 token의 state를 json 의 형식에 맞게 state 추가를 진행하였습니다. 예를 들어 xml에서는 <tagname> string</tagname>의 형식으로 진행되는 것을 json에서는 {“tagname” : string}과같이 적용되기 때문에, data에서 {, [ 와 같은 jsonobject에서 진행되는경우에서 나오는 것으로 인식하고 json과 관련된 state로 state를 transition하도록 진행하였습니다.

또한, Json의 vaule값에 또 Jsonobject가 올 수 있기 때문에 이에 대한 부분도 읽을 수 있도록 진행하였고, “,”로 나뉘어진 key value에 대해서도 제대로 진행할 수 있도록 state를 만들어서 진행하였습니다.

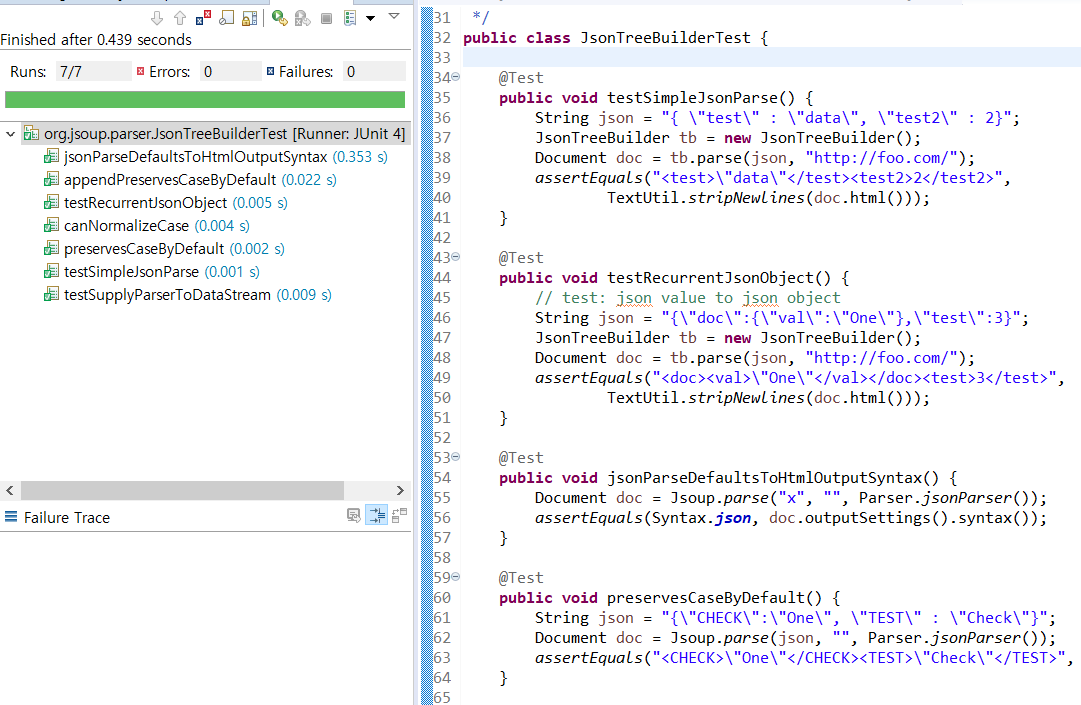
기존 코드에는 Jsonobject를 읽을 수 있도록 해주는 부분이 존재하지 않고, 그에 대한 state처리도 하나도 되어있지 않기 때문에, 이에 대한 부분이 가장 큰 차이입니다.

1. 도입된 설계 패턴 및 설계 원칙과 적용 이유

jsontreebuilder에서는 builder 패턴과, strategy패턴을 적용하였는데, 이 이유는 xmltreebuilder나 htmltreebuilder 들중에 자신들이 원하는 것을 선택하여 적용할 수 있도록 하고, tree를 building 해줘야 하는 부분이 있기 때문에 builder패턴과 strategy 패턴을 적용시켜서 진행하였습니다.

tokeniserstate.java에서 확장을 진행한 부분은, 기존 패턴과 동일하게 진행하였습니다.

1. 테스트 수행 내역



junit을 사용하여 상황을 적용시켜서 테스트를 진행하였습니다.

1번case : “,”로 나뉘어진 Json파일의 경우를 제대로 인식하는지 테스트하고 확인

2번case: jsonobject의 값이 jsonobject가 들어올 수 있기 때문에 이 경우를 인식하는지 확인

3번case: jsonparser를 통해서 제대로 jsonsyntex로 보내는지 확인

4번case: 대소문자를 유지하고 가는지?



5번case: Json을 추가해줄 수 있는지 확인

6번case: 대소문자를 normalize할 수 있는지

7번case : Json파일을 읽어서 제대로 작동 할 수 있는지

1. 깃허브 프로젝트 활용 요약

깃허브를 이용해서 회의록 및, 리포트, 코드를 진행하였습니다. 각자의 역할을 맡아서 진행하였는데, 깃허브를 통해서 각자 진행하거나 기록을 진행한 부분을 올리고, 또한 코드수정에 대한 부분을 진행하였는데, 코드에 대한 부분은 사실 각자 진행하다가 모여서 진행하는 경우도 있어서 모여서 진행할 때에는 커밋을 통합해서 진행하였습니다.

1. 깃허브 주소

<https://github.com/comnamu18/DPTP2019_Team2>

1. progress history 스크린샷

