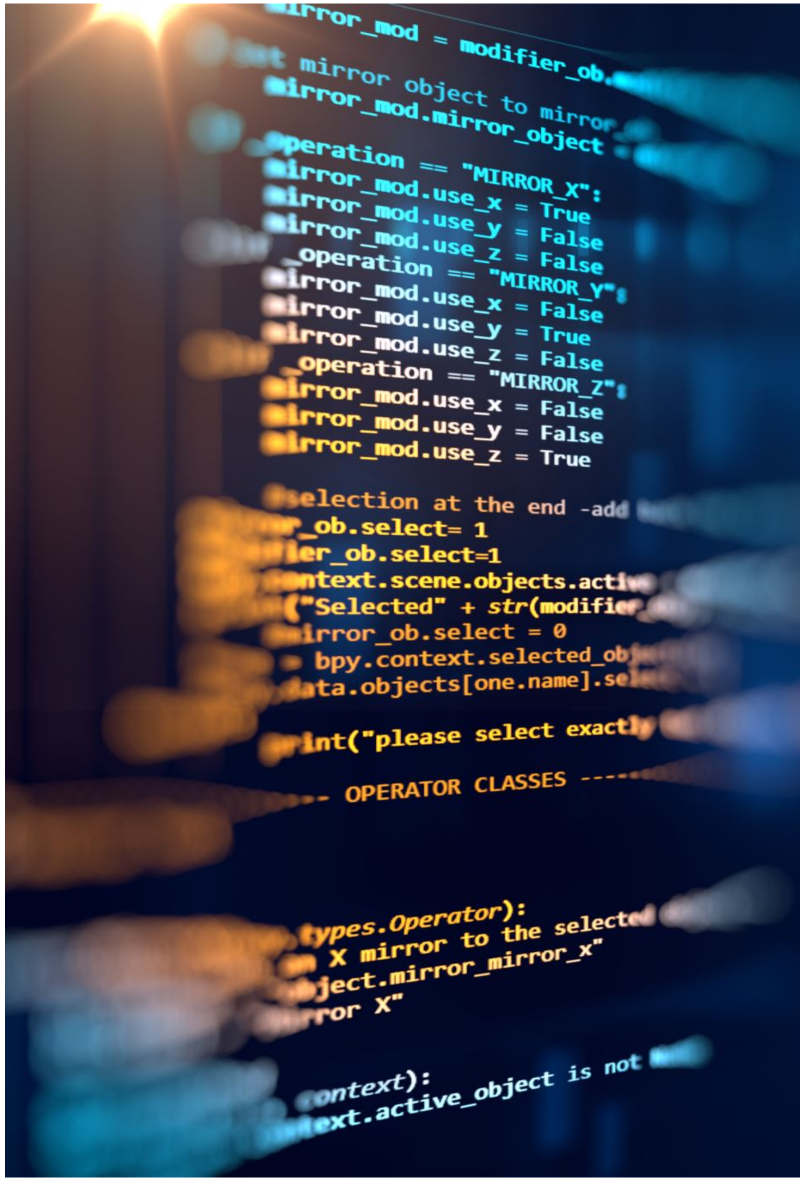
Design Pattern

김희성 20122372
김경태 20175119
주광우 20165953

13조



Jsoup Introduction

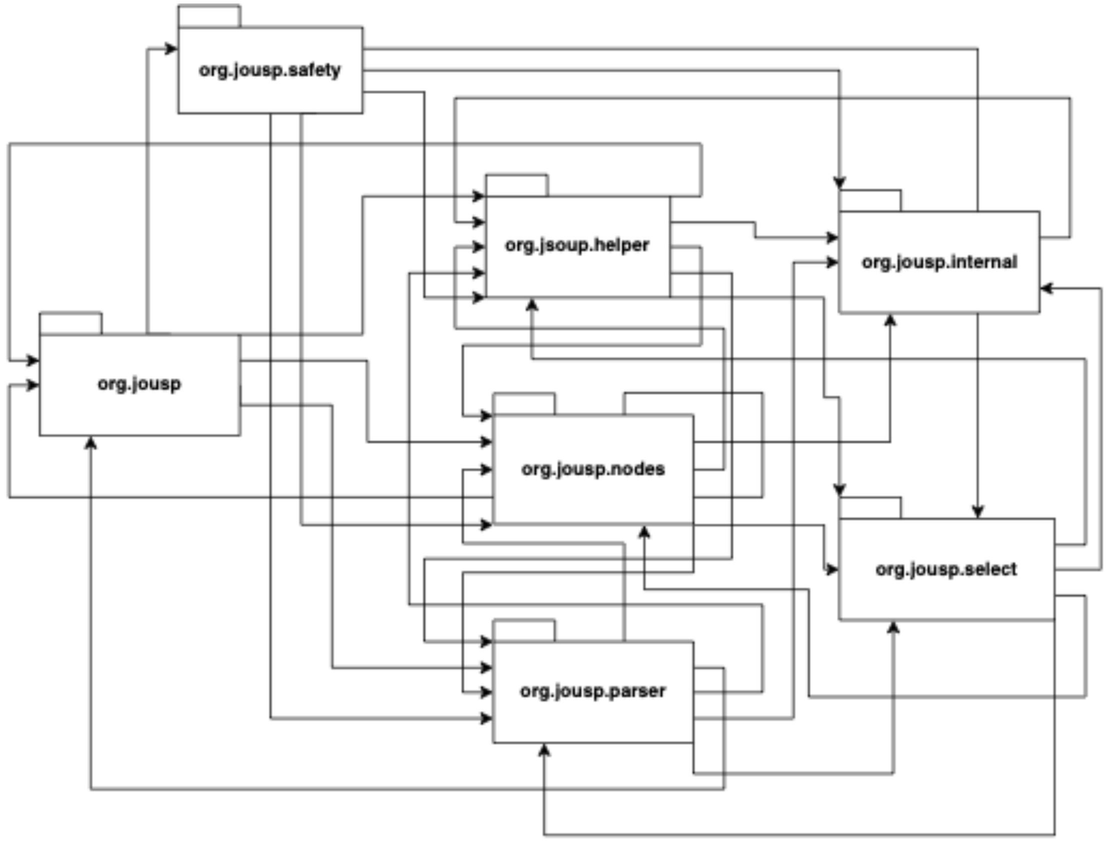
Jsoup는 HTML 문서에 있는 데이터를 parse, extract, manipulate하기 위해 사용되는 오픈 소스 자바 라이브러리이다.

MIT License를 따르고 있으며, 따라서 코드 공개 의무가 없다. 또한 크롤링 등의 목적으로 여러 프로젝트에 사용되고 있다.

Jsoup는 WHATWG에서 정한 HTML5 specification에 맞춰 개발하였고, 브라우저와 같이 HTML을 DOM으로 parse 하는 역할을 한다.

사용자들은 Jsoup을 통해 HTML의 특정 요소들을 parsing해서 얻어 낼 수 있다.

Design Overview



parser

Input Stream을 통해 받은 data를 Tokeniser를 통해 토큰으로 구분한다. 이후, 이 토큰으로부터 TreeBuilder를 통해 HTML이나 XML의 Document Tree를 생성하는 역할.

nodes

HTML과 XML의 Element를 표현하는 클래스들의 집합체로서 역할.

각 클래스에는 Document를 생성할 때 필요한 insert, remove 등의 메소드를 제공.

select

select 패키지 안에는 CSS select query를 parsing한 후, 요청한 노드를 가져오는 역할.

safety

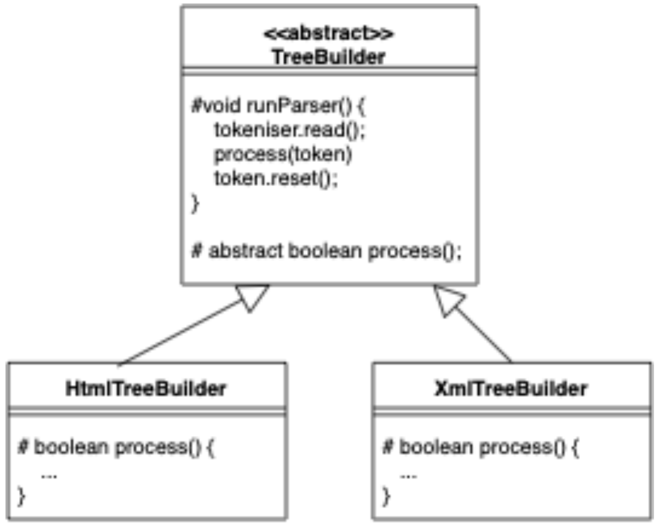
safety에는 whitelist 규칙을 정의하는 Whitelist와 HTML 내용중 그 whitelist 규칙에 어긋나는 것을 없애는 Cleaner 클래스가 있음.

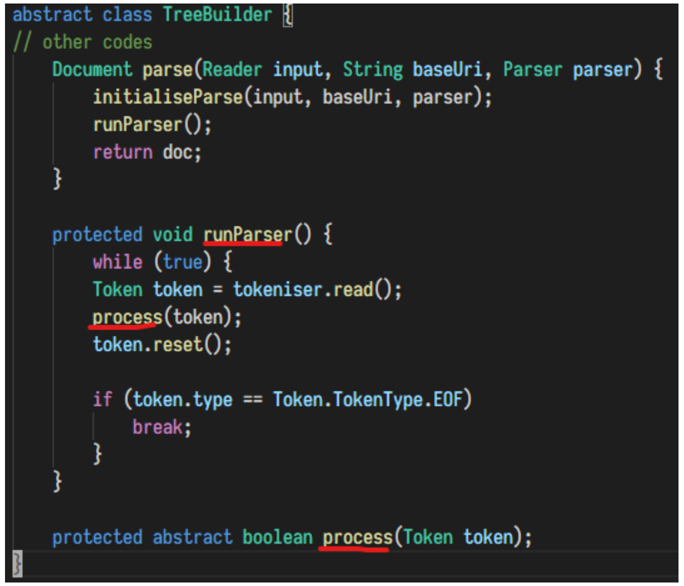
Cleaner를 통해 User가 원하는 Element와 attribute를 포함하고 있는 HTML을 제공받을 수 있음. 또한, cross-site scripting attack을 막음.

Existing Patterns in Jsoup

Template Method Pattern

TreeBuilder

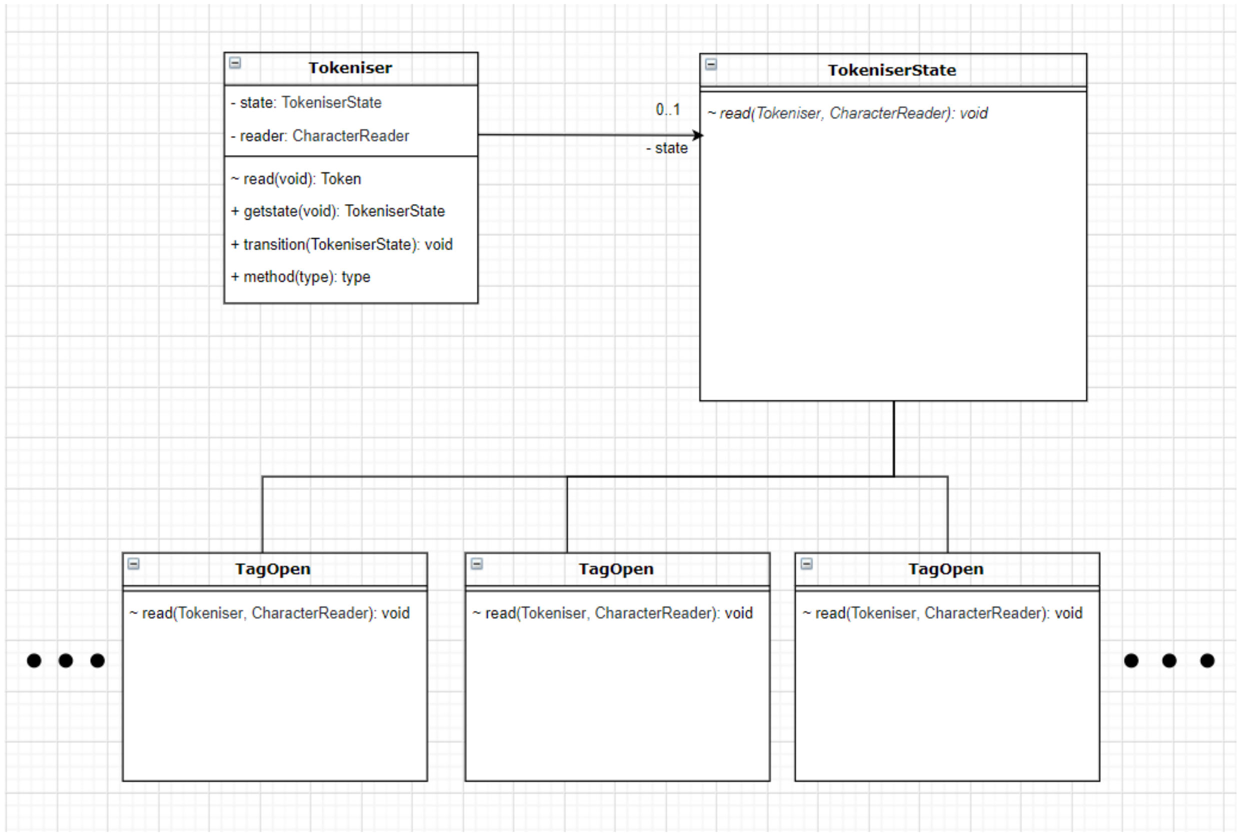




```
abstract class TreeBuilder {
// other codes
    Document parse(Reader input, String baseUri, Parser parser) {
        initialiseParse(input, baseUri, parser);
        runParser();
        return doc;
    }

    protected void runParser() {
        while (true) {
        Token token = tokeniser.read();
        process(token);
        token.reset();

        if (token.type == Token.TokenType.EOF)
            break;
        }
    }

    protected abstract boolean process(Token token);
}
```

State Pattern

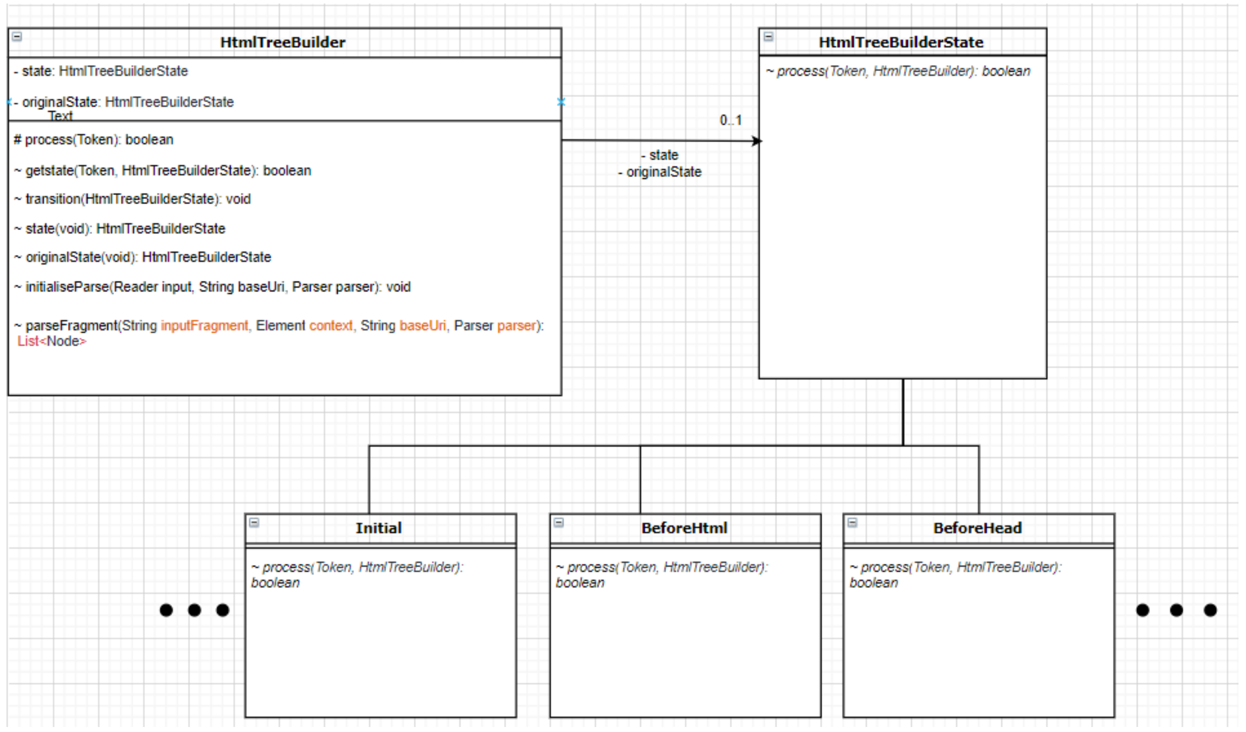
TokeniserState



Tokeniser
- state: TokeniserState
- reader: CharacterReader
~ read(void): Token
+ getstate(void): TokeniserState
+ transition(TokeniserState): void
+ method(type): type
0..1
- state
TokeniserState
~ read(Tokeniser, CharacterReader): void
TagOpen
~ read(Tokeniser, CharacterReader): void
TagOpen
~ read(Tokeniser, CharacterReader): void
TagOpen
~ read(Tokeniser, CharacterReader): void
• • •
• • •

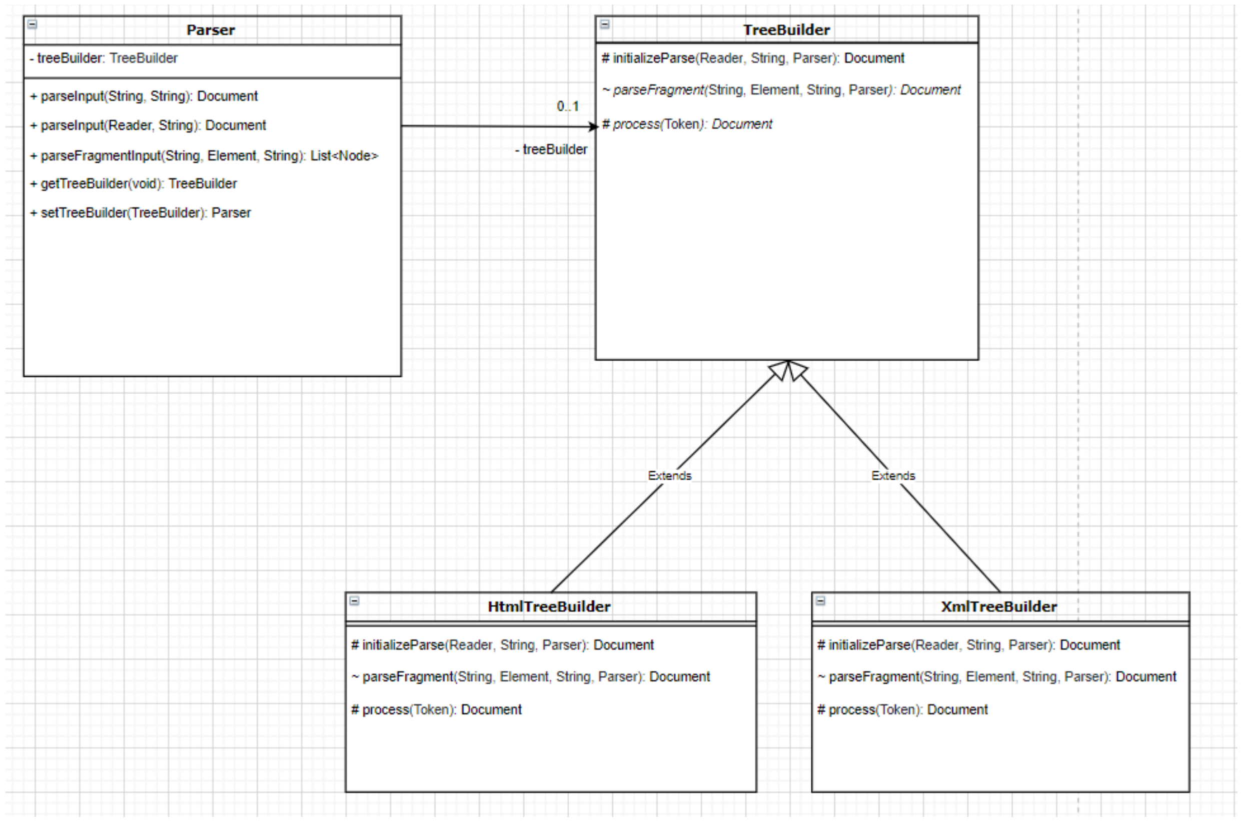
State Pattern

HtmlTreeBuilderState


HtmlTreeBuilder
- state: HtmlTreeBuilderState
- originalState: HtmlTreeBuilderState
Text
# process(Token): boolean
~ getstate(Token, HtmlTreeBuilderState): boolean
~ transition(HtmlTreeBuilderState): void
~ state(void): HtmlTreeBuilderState
~ originalState(void): HtmlTreeBuilderState
~ initialiseParse(Reader input, String baseUri, Parser parser): void
~ parseFragment(String inputFragment, Element context, String baseUri, Parser parser): List<Node>
0..1
- state
- originalState
HtmlTreeBuilderState
~ process(Token, HtmlTreeBuilder): boolean
Initial
~ process(Token, HtmlTreeBuilder): boolean
BeforeHtml
~ process(Token, HtmlTreeBuilder): boolean
BeforeHead
~ process(Token, HtmlTreeBuilder): boolean
• • •
• • •

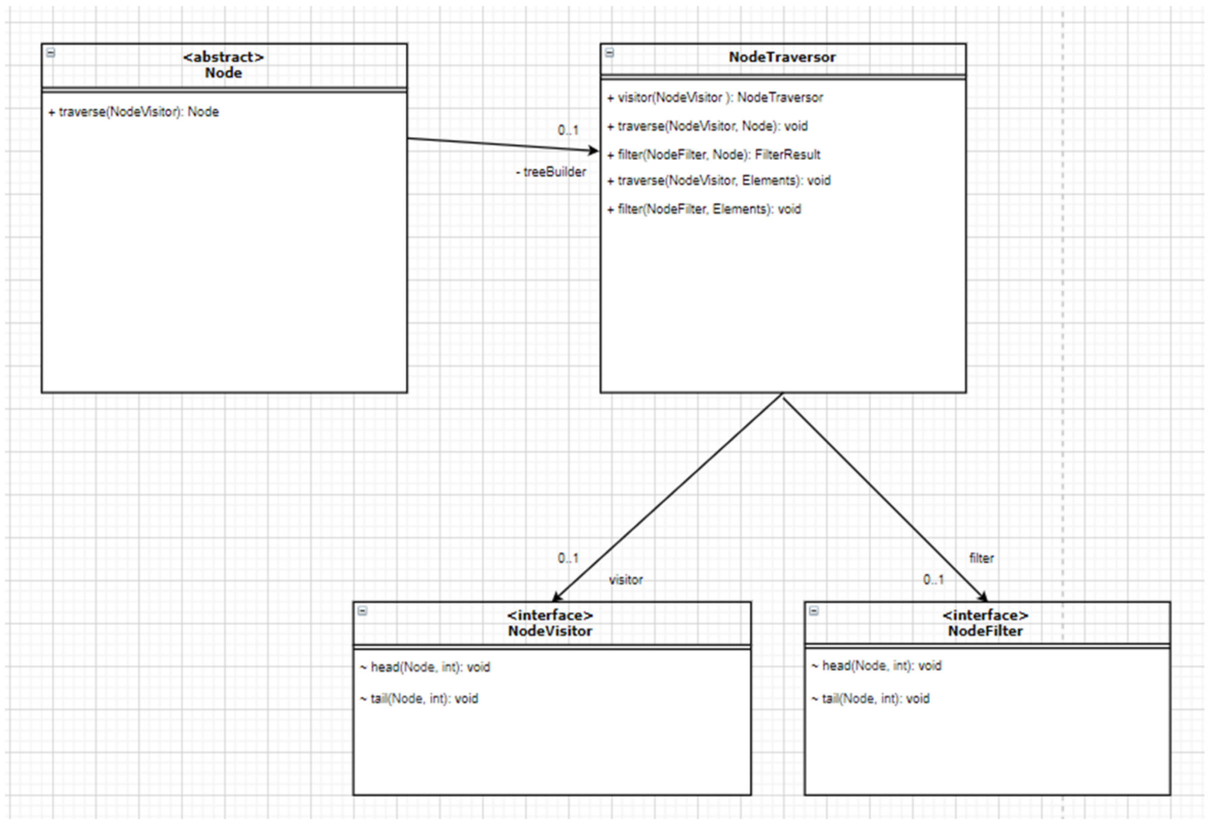
Strategy Pattern

Parser


Parser
- treeBuilder: TreeBuilder
+ parseInput(String, String): Document
+ parseInput(Reader, String): Document
+ parseFragmentInput(String, Element, String): List<Node>
+ getTreeBuilder(void): TreeBuilder
+ setTreeBuilder(TreeBuilder): Parser
0..1
- treeBuilder
TreeBuilder
# initializeParse(Reader, String, Parser): Document
~ parseFragment(String, Element, String, Parser): Document
# process(Token): Document
Extends
Extends
HtmlTreeBuilder
# initializeParse(Reader, String, Parser): Document
~ parseFragment(String, Element, String, Parser): Document
# process(Token): Document
XmlTreeBuilder
# initializeParse(Reader, String, Parser): Document
~ parseFragment(String, Element, String, Parser): Document
# process(Token): Document

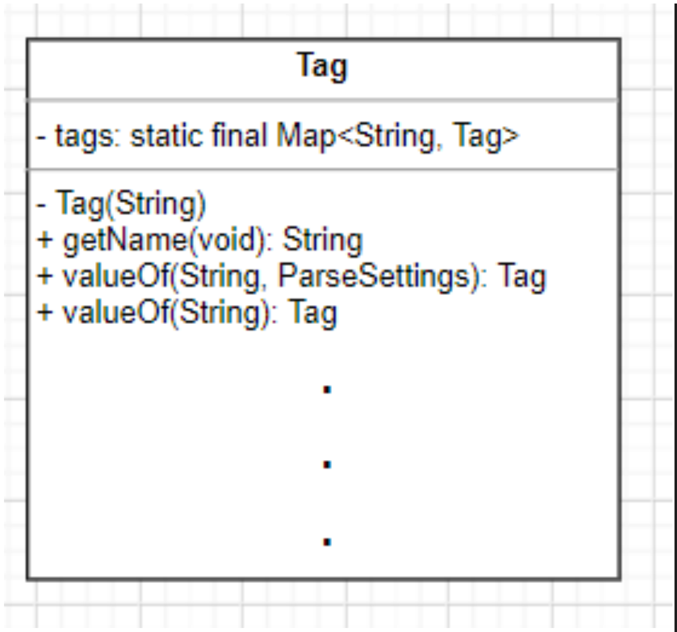
Strategy Pattern

NodeTraversor


<abstract>
Node
+ traverse(NodeVisitor): Node
0..1
- treeBuilder
NodeTraversor
+ visitor(NodeVisitor): NodeTraversor
+ traverse(NodeVisitor, Node): void
+ filter(NodeFilter, Node): FilterResult
+ traverse(NodeVisitor, Elements): void
+ filter(NodeFilter, Elements): void
0..1
visitor
filter
0..1
<interface>
NodeVisitor
~ head(Node, int): void
~ tail(Node, int): void
<interface>
NodeFilter
~ head(Node, int): void
~ tail(Node, int): void

Singleton Pattern

Tag


Tag

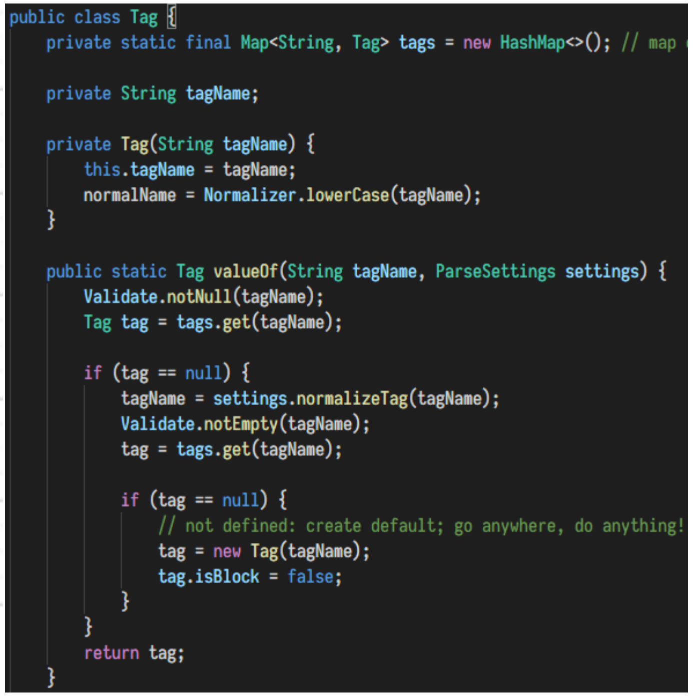
- tags: static final Map<String, Tag>

- Tag(String)
+ getName(void): String
+ valueOf(String, ParseSettings): Tag
+ valueOf(String): Tag

.

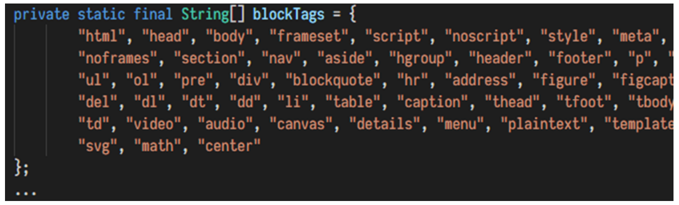
.

.

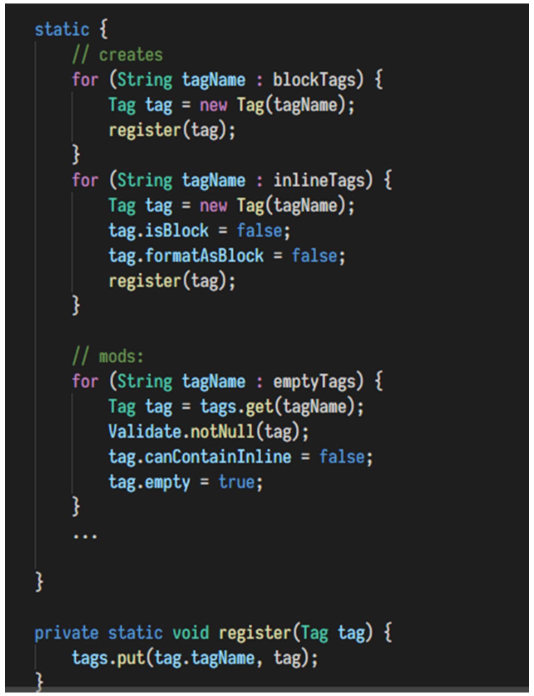

```java
public class Tag {
    private static final Map<String, Tag> tags = new HashMap<>(); // map

    private String tagName;

    private Tag(String tagName) {
        this.tagName = tagName;
        normalName = Normalizer.lowerCase(tagName);
    }

    public static Tag valueOf(String tagName, ParseSettings settings) {
        Validate.notNull(tagName);
        Tag tag = tags.get(tagName);

        if (tag == null) {
            tagName = settings.normalizeTag(tagName);
            Validate.notEmpty(tagName);
            tag = tags.get(tagName);

            if (tag == null) {
                // not defined: create default; go anywhere, do anything!
                tag = new Tag(tagName);
                tag.isBlock = false;
            }
        }
        return tag;
    }
```

Singleton Pattern

Tag



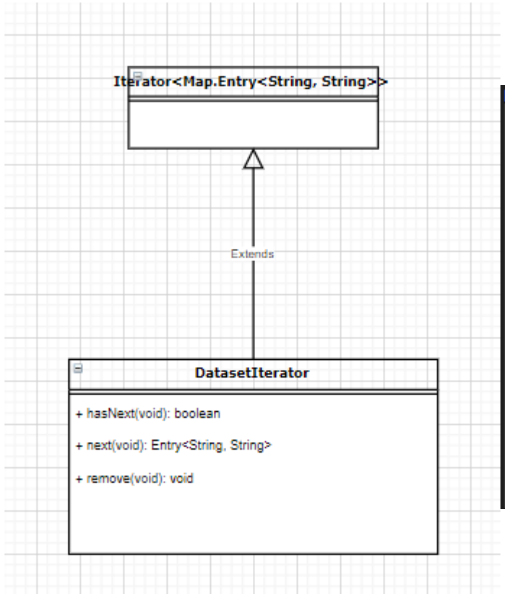
```
private static final String[] blockTags = {
        "html", "head", "body", "frameset", "script", "noscript", "style", "meta", 
        "noframes", "section", "nav", "aside", "hgroup", "header", "footer", "p", "
        "ul", "ol", "pre", "div", "blockquote", "hr", "address", "figure", "figcapt
        "del", "dl", "dt", "dd", "li", "table", "caption", "thead", "tfoot", "tbody
        "td", "video", "audio", "canvas", "details", "menu", "plaintext", "template
        "svg", "math", "center"
};
...
```



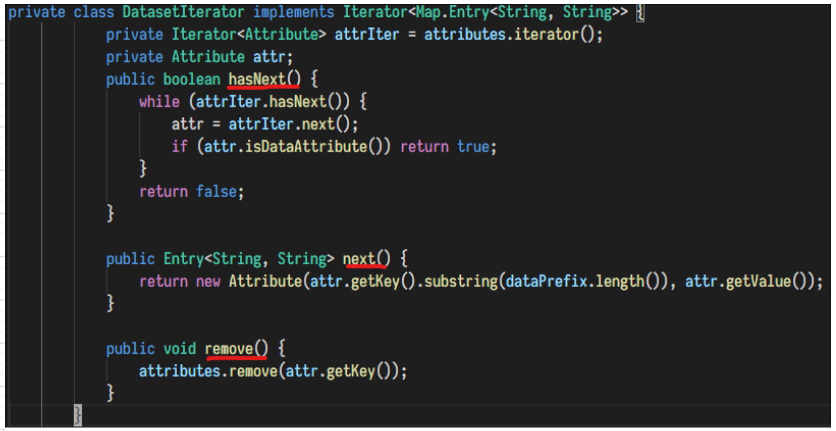
```
static {
    // creates
    for (String tagName : blockTags) {
        Tag tag = new Tag(tagName);
        register(tag);
    }
    for (String tagName : inlineTags) {
        Tag tag = new Tag(tagName);
        tag.isBlock = false;
        tag.formatAsBlock = false;
        register(tag);
    }

    // mods:
    for (String tagName : emptyTags) {
        Tag tag = tags.get(tagName);
        Validate.notNull(tag);
        tag.canContainInline = false;
        tag.empty = true;
    }
    ...

}

private static void register(Tag tag) {
    tags.put(tag.tagName, tag);
}
```

Iterator Pattern

DatesetIterator



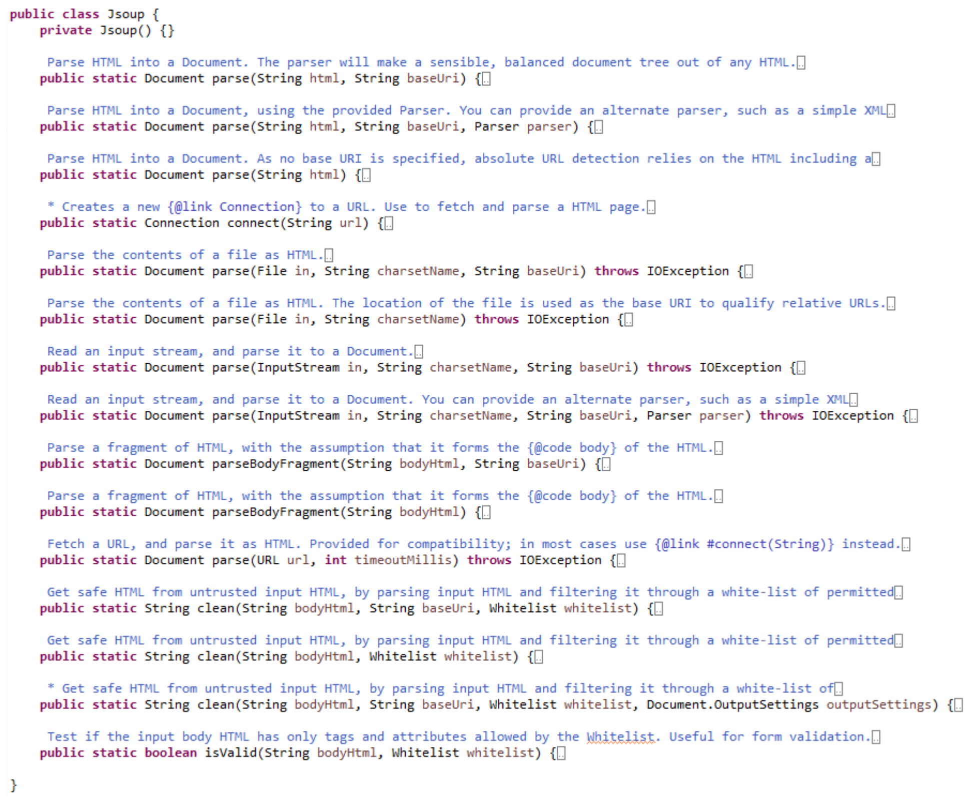
Iterator<Map.Entry<String, String>>
Extends
DatasetIterator
+ hasNext(void): boolean
+ next(void): Entry<String, String>
+ remove(void): void



```java
private class DatasetIterator implements Iterator<Map.Entry<String, String>> {
            private Iterator<Attribute> attrIter = attributes.iterator();
            private Attribute attr;
            public boolean hasNext() {
                while (attrIter.hasNext()) {
                    attr = attrIter.next();
                    if (attr.isDataAttribute()) return true;
                }
                return false;
            }

            public Entry<String, String> next() {
                return new Attribute(attr.getKey().substring(dataPrefix.length()), attr.getValue());
            }

            public void remove() {
                attributes.remove(attr.getKey());
            }
        }
```

Façade Pattern

Jsoup



```java
public class Jsoup {
    private Jsoup() {}

     Parse HTML into a Document. The parser will make a sensible, balanced document tree out of any HTML.
    public static Document parse(String html, String baseUri) {

     Parse HTML into a Document, using the provided Parser. You can provide an alternate parser, such as a simple XML
    public static Document parse(String html, String baseUri, Parser parser) {

     Parse HTML into a Document. As no base URI is specified, absolute URL detection relies on the HTML including a
    public static Document parse(String html) {

     * Creates a new {@link Connection} to a URL. Use to fetch and parse a HTML page.
    public static Connection connect(String url) {

     Parse the contents of a file as HTML.
    public static Document parse(File in, String charsetName, String baseUri) throws IOException {

     Parse the contents of a file as HTML. The location of the file is used as the base URI to qualify relative URLs.
    public static Document parse(File in, String charsetName) throws IOException {

     Read an input stream, and parse it to a Document.
    public static Document parse(InputStream in, String charsetName, String baseUri) throws IOException {

     Read an input stream, and parse it to a Document. You can provide an alternate parser, such as a simple XML
    public static Document parse(InputStream in, String charsetName, String baseUri, Parser parser) throws IOException {

     Parse a fragment of HTML, with the assumption that it forms the {@code body} of the HTML.
    public static Document parseBodyFragment(String bodyHtml, String baseUri) {

     Parse a fragment of HTML, with the assumption that it forms the {@code body} of the HTML.
    public static Document parseBodyFragment(String bodyHtml) {

     Fetch a URL, and parse it as HTML. Provided for compatibility; in most cases use {@link #connect(String)} instead.
    public static Document parse(URL url, int timeoutMillis) throws IOException {

     Get safe HTML from untrusted input HTML, by parsing input HTML and filtering it through a white-list of permitted
    public static String clean(String bodyHtml, String baseUri, Whitelist whitelist) {

     Get safe HTML from untrusted input HTML, by parsing input HTML and filtering it through a white-list of permitted
    public static String clean(String bodyHtml, Whitelist whitelist) {

     * Get safe HTML from untrusted input HTML, by parsing input HTML and filtering it through a white-list of
    public static String clean(String bodyHtml, String baseUri, Whitelist whitelist, Document.OutputSettings outputSettings) {

     Test if the input body HTML has only tags and attributes allowed by the Whitelist. Useful for form validation.
    public static boolean isValid(String bodyHtml, Whitelist whitelist) {

}
```

Improvement

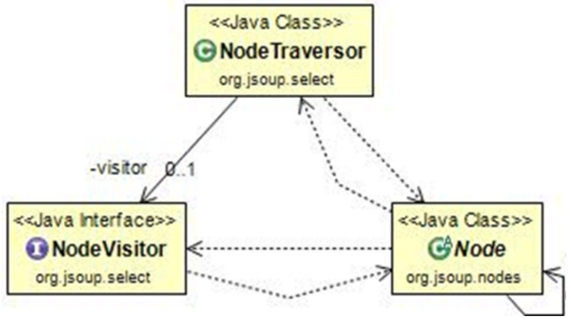
Add functions

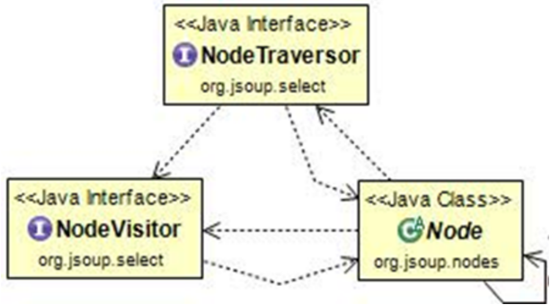
Purpose : Find the last element

Approach : Add Node Traversal Algorithm

Add
Node Traversal
Algorithm

Add Visitor Pattern





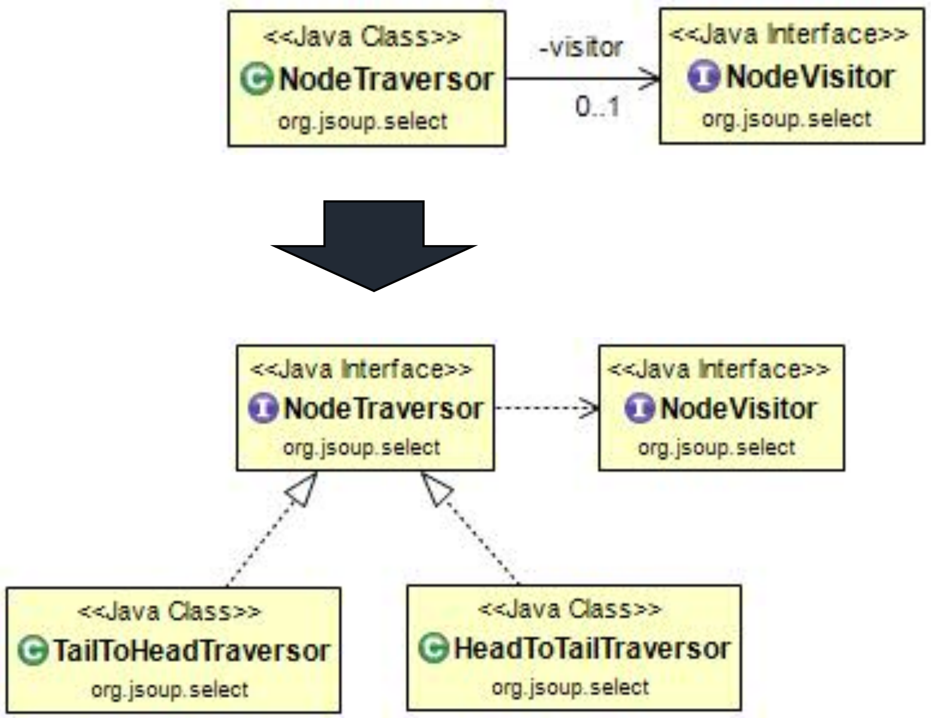
```
public Node traverse(NodeVisitor nodeVisitor) {
    Validate.notNull(nodeVisitor);
    NodeTraversor.traverse(nodeVisitor, this);
    return this;
}
```

```
public Node traverse(NodeVisitor nodeVisitor) {
    Validate.notNull(nodeVisitor);
    NodeTraversor nodeTraversor
                        = new HeadToTailTraversor();
    nodeTraversor.traverse(nodeVisitor, this);
    return this;
}
```

```
public Node traverse(   NodeTraversor nodeTraversor,
                        NodeVisitor nodeVisitor) {
    Validate.notNull(nodeTraversor);
    Validate.notNull(nodeVisitor);
    nodeTraversor.traverse(nodeVisitor, this);
    return this;
}
```

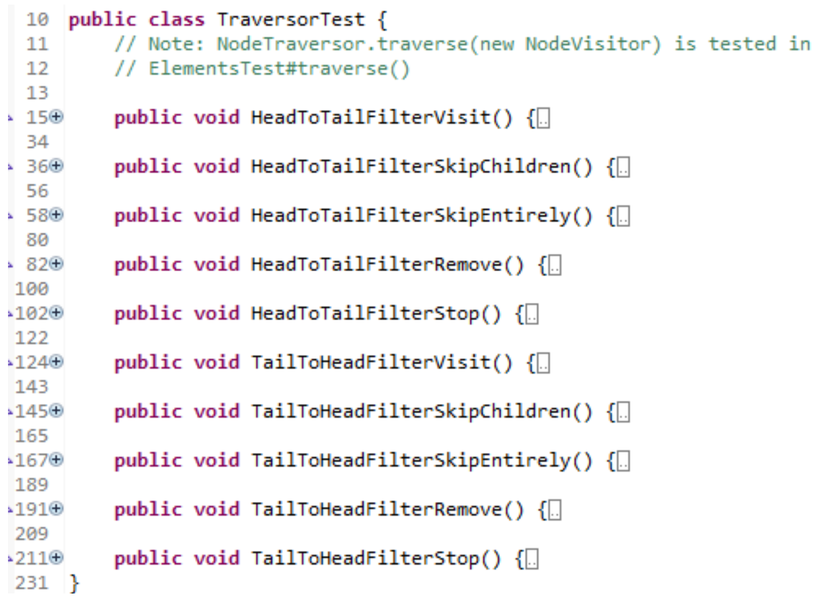
Add
Node Traversal
Algorithm

Add Concrete Class



Add
Unit Test

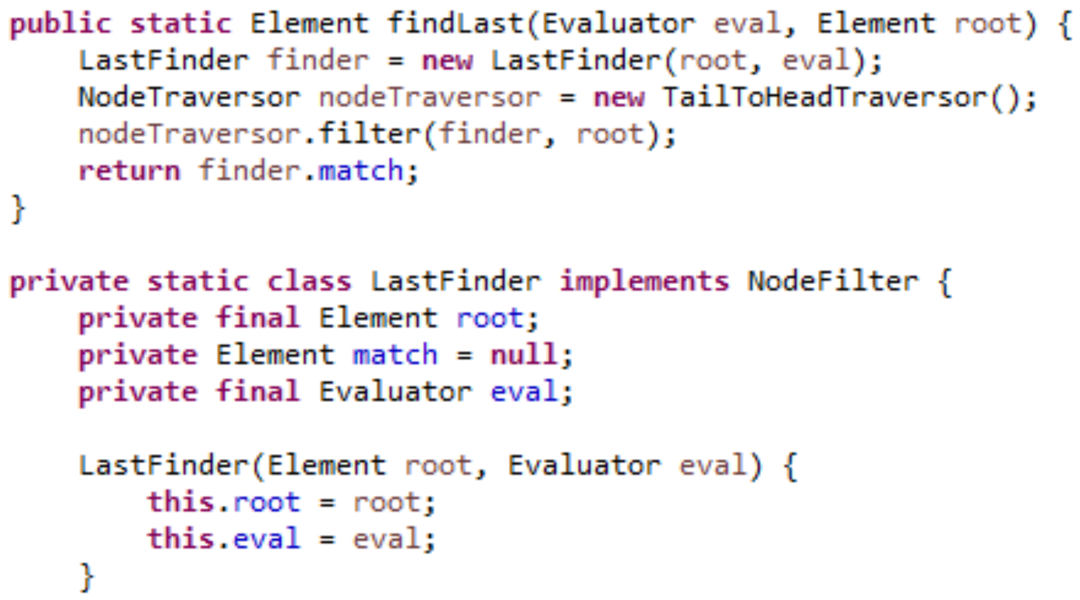
Unit Test for Concrete Class of NodeTraversoer



```
public class TraversorTest {
    // Note: NodeTraversor.traverse(new NodeVisitor) is tested in
    // ElementsTest#traverse()

    public void HeadToTailFilterVisit() {

    public void HeadToTailFilterSkipChildren() {

    public void HeadToTailFilterSkipEntirely() {

    public void HeadToTailFilterRemove() {

    public void HeadToTailFilterStop() {

    public void TailToHeadFilterVisit() {

    public void TailToHeadFilterSkipChildren() {

    public void TailToHeadFilterSkipEntirely() {

    public void TailToHeadFilterRemove() {

    public void TailToHeadFilterStop() {
}
```

Add
Method

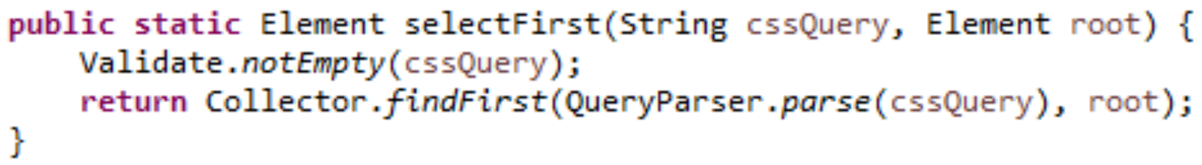
Add Collector.findLast



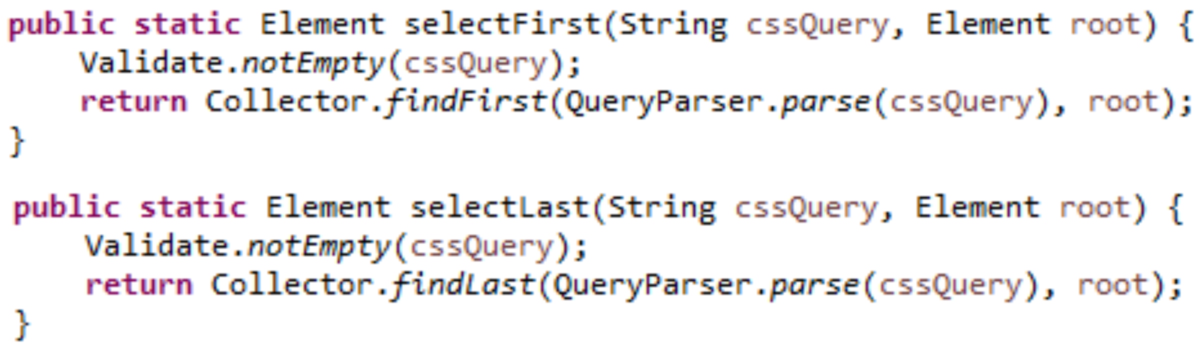
```
public static Element findLast(Evaluator eval, Element root) {
    LastFinder finder = new LastFinder(root, eval);
    NodeTraversor nodeTraversor = new TailToHeadTraversor();
    nodeTraversor.filter(finder, root);
    return finder.match;
}

private static class LastFinder implements NodeFilter {
    private final Element root;
    private Element match = null;
    private final Evaluator eval;

    LastFinder(Element root, Evaluator eval) {
        this.root = root;
        this.eval = eval;
    }
```

Add
Method

Add Selector.selectLast



```java
public static Element selectFirst(String cssQuery, Element root) {
    Validate.notEmpty(cssQuery);
    return Collector.findFirst(QueryParser.parse(cssQuery), root);
}
```



```java
public static Element selectFirst(String cssQuery, Element root) {
    Validate.notEmpty(cssQuery);
    return Collector.findFirst(QueryParser.parse(cssQuery), root);
}

public static Element selectLast(String cssQuery, Element root) {
    Validate.notEmpty(cssQuery);
    return Collector.findLast(QueryParser.parse(cssQuery), root);
}
```

Add
Method

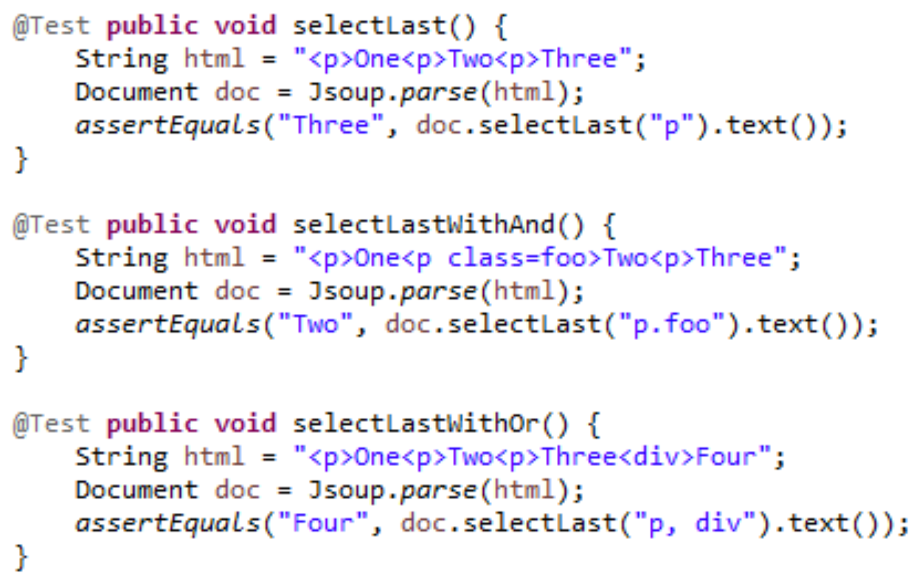
Add Element.selectLast

```java
public Element selectFirst(String cssQuery) {
    return Selector.selectFirst(cssQuery, this);
}
```

```java
public Element selectFirst(String cssQuery) {
    return Selector.selectFirst(cssQuery, this);
}

public Element selectLast(String cssQuery) {
    return Selector.selectLast(cssQuery, this);
}
```

Add
Unit Test

Unit Test for selectLast method



```
@Test public void selectLast() {
    String html = "<p>One<p>Two<p>Three";
    Document doc = Jsoup.parse(html);
    assertEquals("Three", doc.selectLast("p").text());
}

@Test public void selectLastWithAnd() {
    String html = "<p>One<p class=foo>Two<p>Three";
    Document doc = Jsoup.parse(html);
    assertEquals("Two", doc.selectLast("p.foo").text());
}

@Test public void selectLastWithOr() {
    String html = "<p>One<p>Two<p>Three<div>Four";
    Document doc = Jsoup.parse(html);
    assertEquals("Four", doc.selectLast("p, div").text());
}
```

Extension

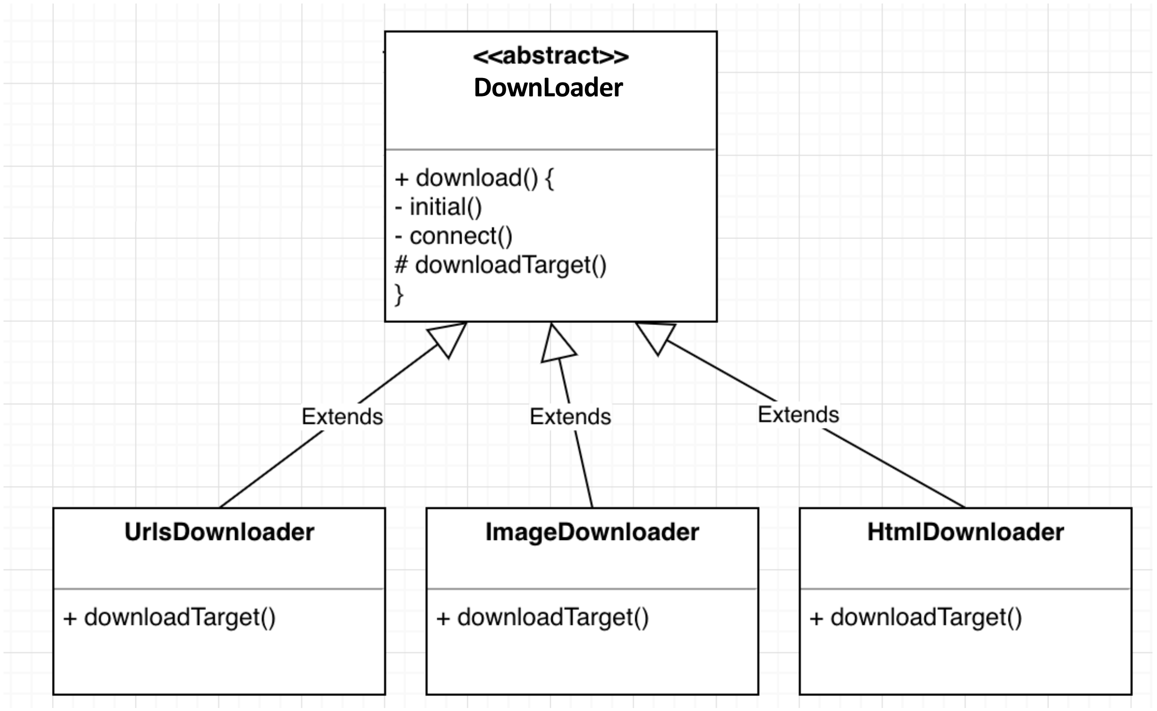
Add package

Purpose : Download contents of page

Approach : Add Content Downloader classes

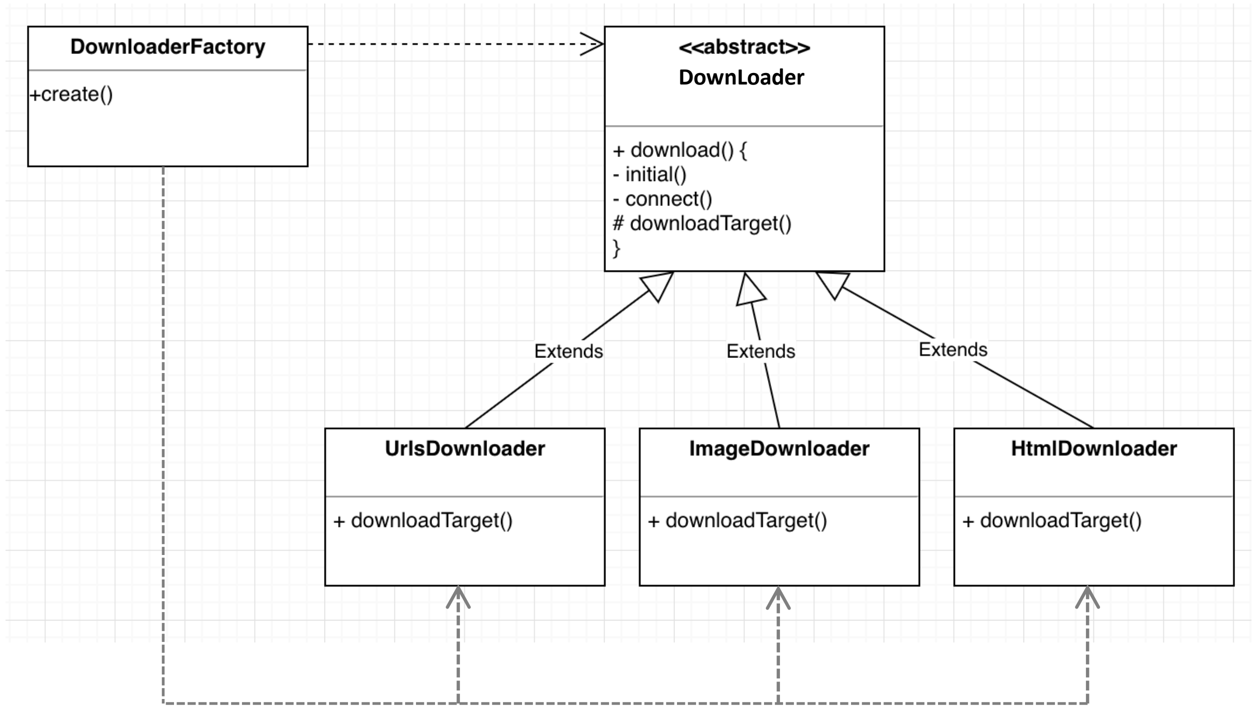
Add
Contents Downloader
Class

Add Template Method Pattern



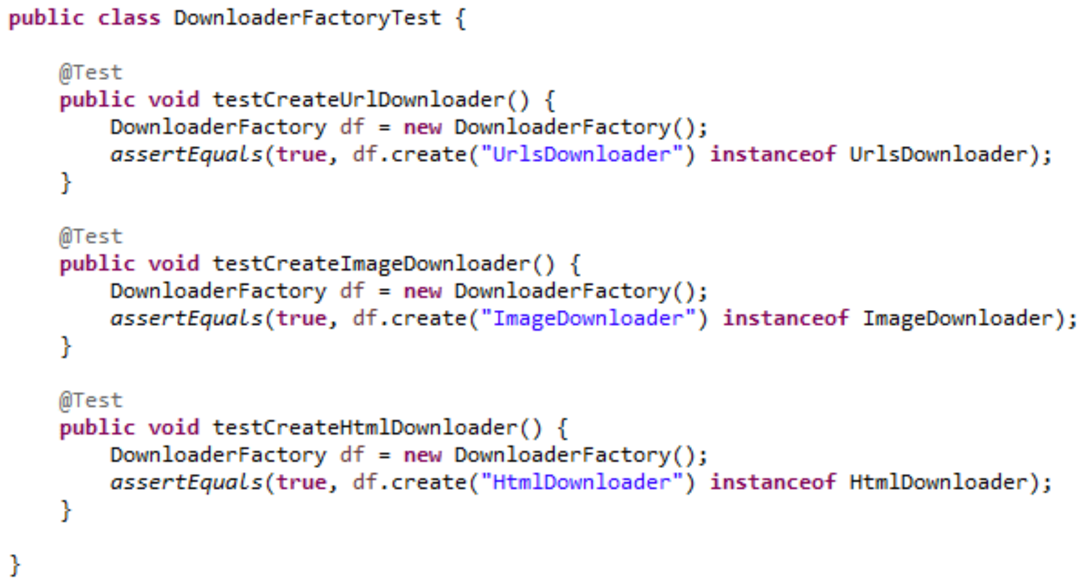
Add
Contents Downloader
Class

Add Façade Pattern with Simple Factory



Add
Unit Test

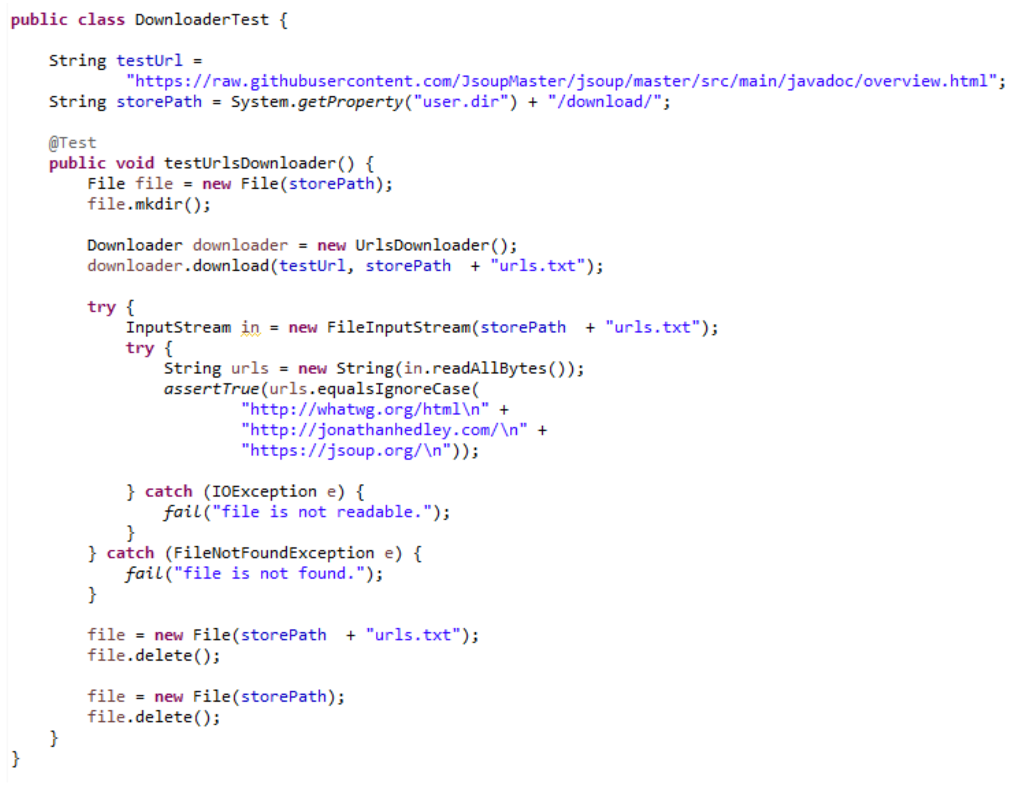
Add Unit Test for DownloaderFacotry



```
public class DownloaderFactoryTest {

    @Test
    public void testCreateUrlDownloader() {
        DownloaderFactory df = new DownloaderFactory();
        assertEquals(true, df.create("UrlsDownloader") instanceof UrlsDownloader);
    }

    @Test
    public void testCreateImageDownloader() {
        DownloaderFactory df = new DownloaderFactory();
        assertEquals(true, df.create("ImageDownloader") instanceof ImageDownloader);
    }

    @Test
    public void testCreateHtmlDownloader() {
        DownloaderFactory df = new DownloaderFactory();
        assertEquals(true, df.create("HtmlDownloader") instanceof HtmlDownloader);
    }

}
```

Add
Unit Test

Add Unit Test for UrlsDownloader

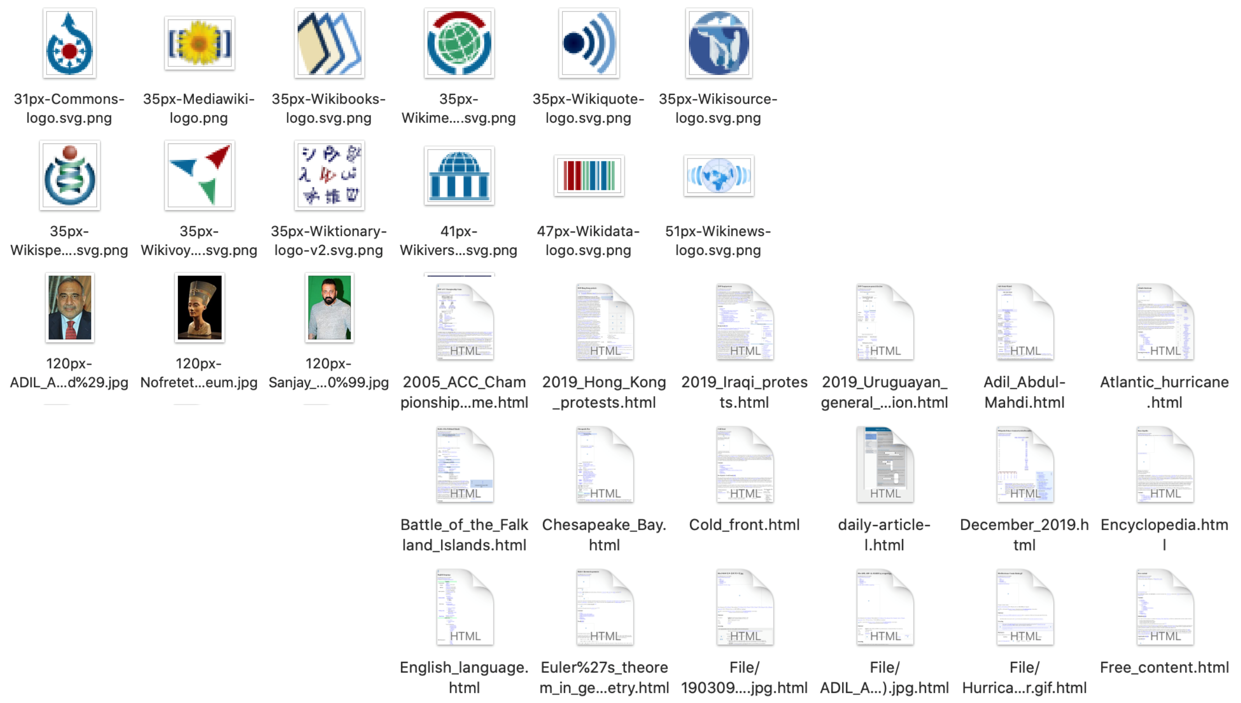


```java
public class DownloaderTest {

    String testUrl =
            "https://raw.githubusercontent.com/JsoupMaster/jsoup/master/src/main/javadoc/overview.html";
    String storePath = System.getProperty("user.dir") + "/download/";

    @Test
    public void testUrlsDownloader() {
        File file = new File(storePath);
        file.mkdir();

        Downloader downloader = new UrlsDownloader();
        downloader.download(testUrl, storePath  + "urls.txt");

        try {
            InputStream in = new FileInputStream(storePath  + "urls.txt");
            try {
                String urls = new String(in.readAllBytes());
                assertTrue(urls.equalsIgnoreCase(
                        "http://whatwg.org/html\n" +
                        "http://jonathanhedley.com/\n" +
                        "https://jsoup.org/\n"));

            } catch (IOException e) {
                fail("file is not readable.");
            }
        } catch (FileNotFoundException e) {
            fail("file is not found.");
        }

        file = new File(storePath  + "urls.txt");
        file.delete();

        file = new File(storePath);
        file.delete();
    }
}
```

Add Example

DepthOneCrawler

(ImageDownloader, HtmlDownloader)

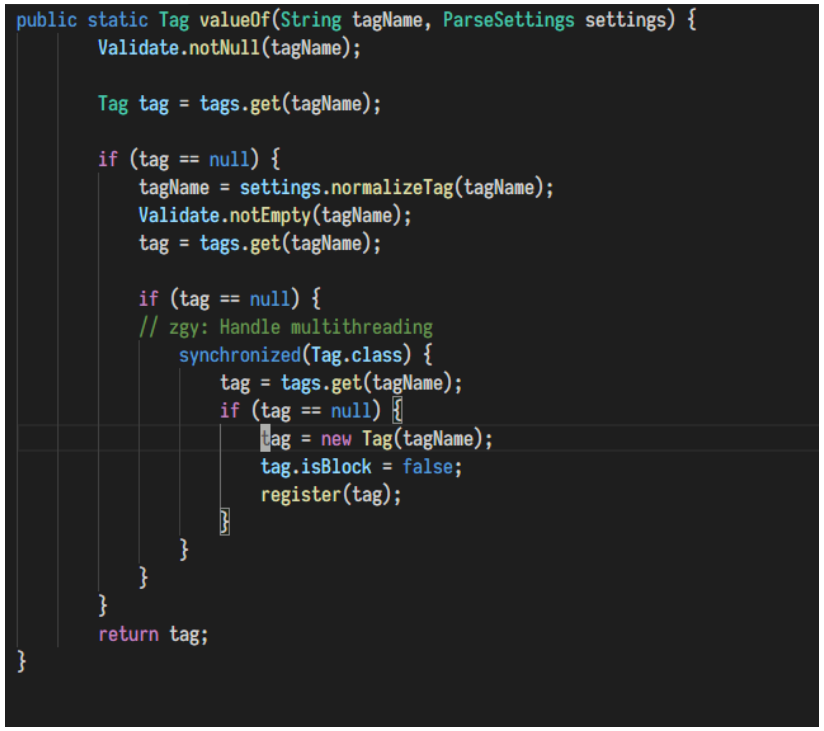


31px-Commons-logo.svg.png
35px-Mediawiki-logo.png
35px-Wikibooks-logo.svg.png
35px-Wikime....svg.png
35px-Wikiquote-logo.svg.png
35px-Wikisource-logo.svg.png
35px-Wikispe....svg.png
35px-Wikivoy....svg.png
35px-Wiktionary-logo-v2.svg.png
41px-Wikivers...svg.png
47px-Wikidata-logo.svg.png
51px-Wikinews-logo.svg.png
120px-ADIL_A...d%29.jpg
120px-Nofretet...eum.jpg
120px-Sanjay_...0%99.jpg
2005_ACC_Championship...me.html
2019_Hong_Kong_protests.html
2019_Iraqi_protests.html
2019_Uruguayan_general_...ion.html
Adil_Abdul-Mahdi.html
Atlantic_hurricane.html
Battle_of_the_Falkland_Islands.html
Chesapeake_Bay.html
Cold_front.html
daily-article-l.html
December_2019.html
Encyclopedia.html
English_language.html
Euler%27s_theorem_in_ge...etry.html
File/190309....jpg.html
File/ADIL_A...).jpg.html
File/Hurrica...r.gif.html
Free_content.html

Improvement that Failed

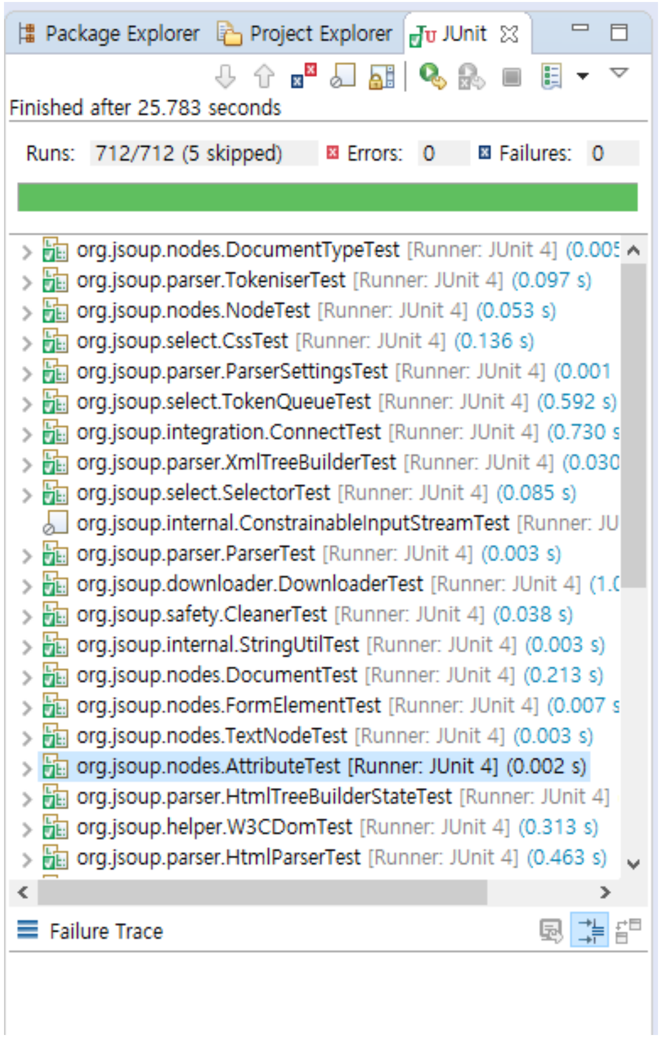
Singleton
Pattern

Tag


```java
public static Tag valueOf(String tagName, ParseSettings settings) {
        Validate.notNull(tagName);

        Tag tag = tags.get(tagName);

        if (tag == null) {
            tagName = settings.normalizeTag(tagName);
            Validate.notEmpty(tagName);
            tag = tags.get(tagName);

            if (tag == null) {
            // zgy: Handle multithreading
                synchronized(Tag.class) {
                    tag = tags.get(tagName);
                    if (tag == null) {
                        tag = new Tag(tagName);
                        tag.isBlock = false;
                        register(tag);
                    }
                }
            }
        }
        return tag;
}
```

Unit Test

Test Result



```
Finished after 25.783 seconds

Runs: 712/712 (5 skipped)   Errors: 0   Failures: 0

org.jsoup.nodes.DocumentTypeTest [Runner: JUnit 4] (0.005
org.jsoup.parser.TokeniserTest [Runner: JUnit 4] (0.097 s)
org.jsoup.nodes.NodeTest [Runner: JUnit 4] (0.053 s)
org.jsoup.select.CssTest [Runner: JUnit 4] (0.136 s)
org.jsoup.parser.ParserSettingsTest [Runner: JUnit 4] (0.001
org.jsoup.select.TokenQueueTest [Runner: JUnit 4] (0.592 s)
org.jsoup.integration.ConnectTest [Runner: JUnit 4] (0.730 s
org.jsoup.parser.XmlTreeBuilderTest [Runner: JUnit 4] (0.030
org.jsoup.select.SelectorTest [Runner: JUnit 4] (0.085 s)
org.jsoup.internal.ConstrainableInputStreamTest [Runner: JU
org.jsoup.parser.ParserTest [Runner: JUnit 4] (0.003 s)
org.jsoup.downloader.DownloaderTest [Runner: JUnit 4] (1.0
org.jsoup.safety.CleanerTest [Runner: JUnit 4] (0.038 s)
org.jsoup.internal.StringUtilTest [Runner: JUnit 4] (0.003 s)
org.jsoup.nodes.DocumentTest [Runner: JUnit 4] (0.213 s)
org.jsoup.nodes.FormElementTest [Runner: JUnit 4] (0.007 s
org.jsoup.nodes.TextNodeTest [Runner: JUnit 4] (0.003 s)
org.jsoup.nodes.AttributeTest [Runner: JUnit 4] (0.002 s)
org.jsoup.parser.HtmlTreeBuilderStateTest [Runner: JUnit 4]
org.jsoup.helper.W3CDomTest [Runner: JUnit 4] (0.313 s)
org.jsoup.parser.HtmlParserTest [Runner: JUnit 4] (0.463 s)

Failure Trace
```