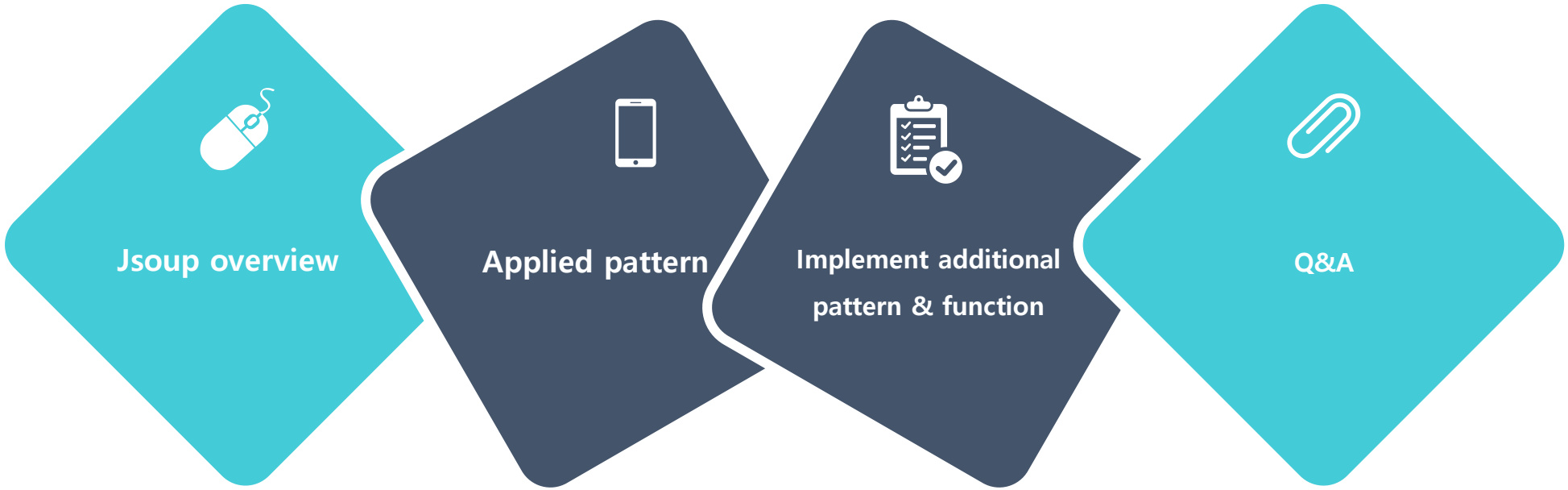
Jsoup DesignPattern

김지민
이성민
발표자 : 하태윤

Jsoup Designpattern

INDEX

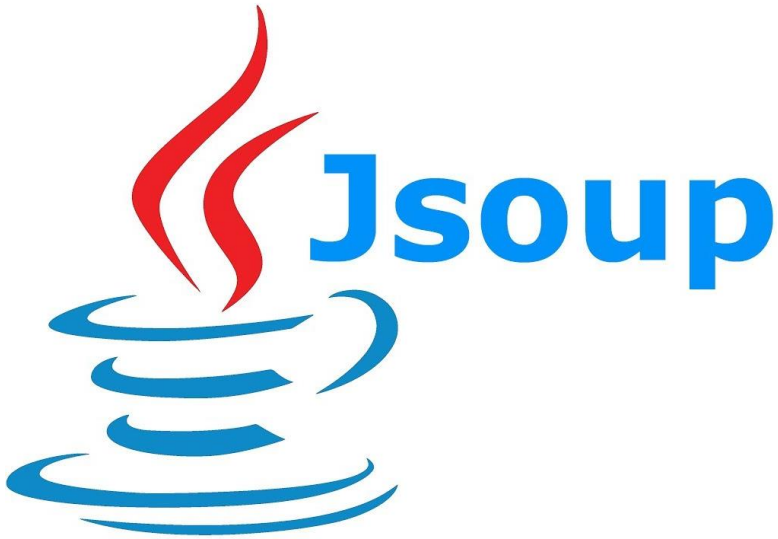


Jsoup OVERVIEW

Jsoup Designpattern

Jsoup Overview

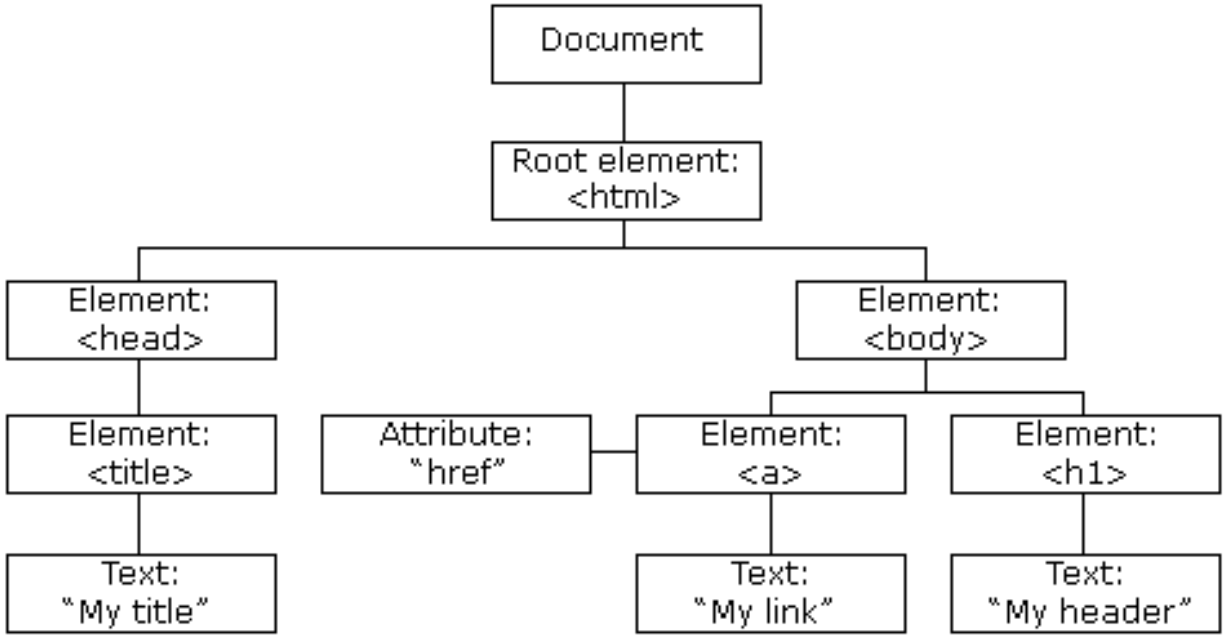
Jsoup은 기본적으로 HTML형식의 string을 받아와 JAVA에서 사용할 수 있는
DOM객체로 만들어 주는 source이다.



Jsoup Designpattern

Jsoup Overview

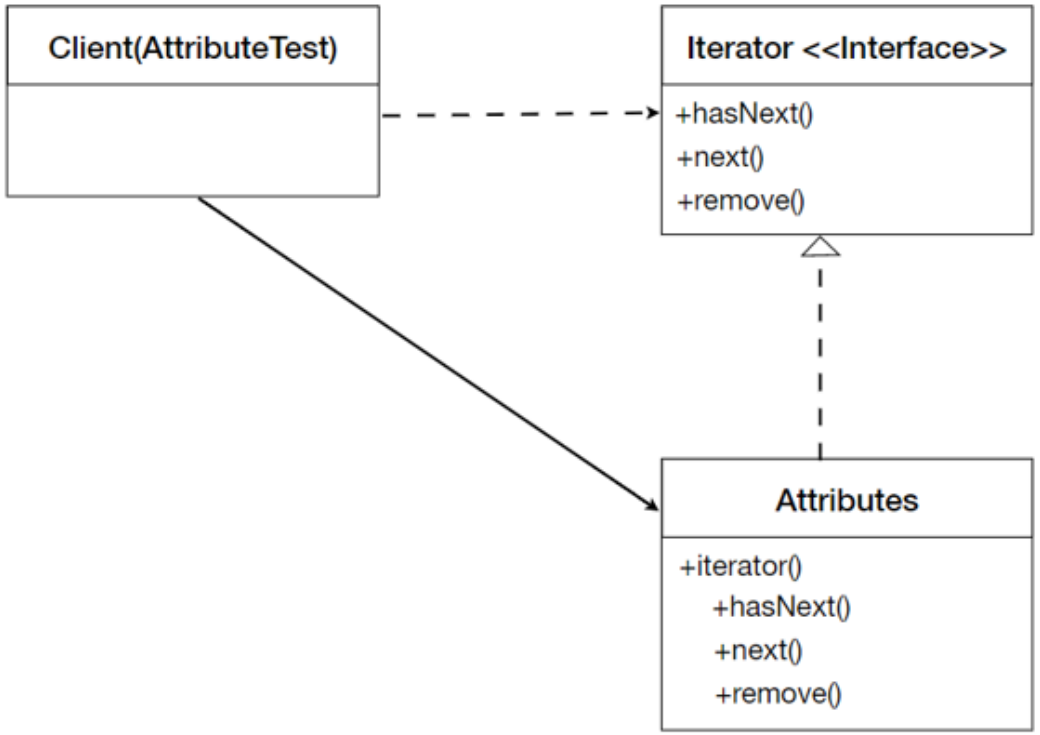
DOM(DOCUMENT OBJECT MODEL)의 개체 구조는 NODE TREE형태로 나타나게 된다.



Applied Pattern

Jsoup Designpattern

Applied Pattern (iterator pattern)

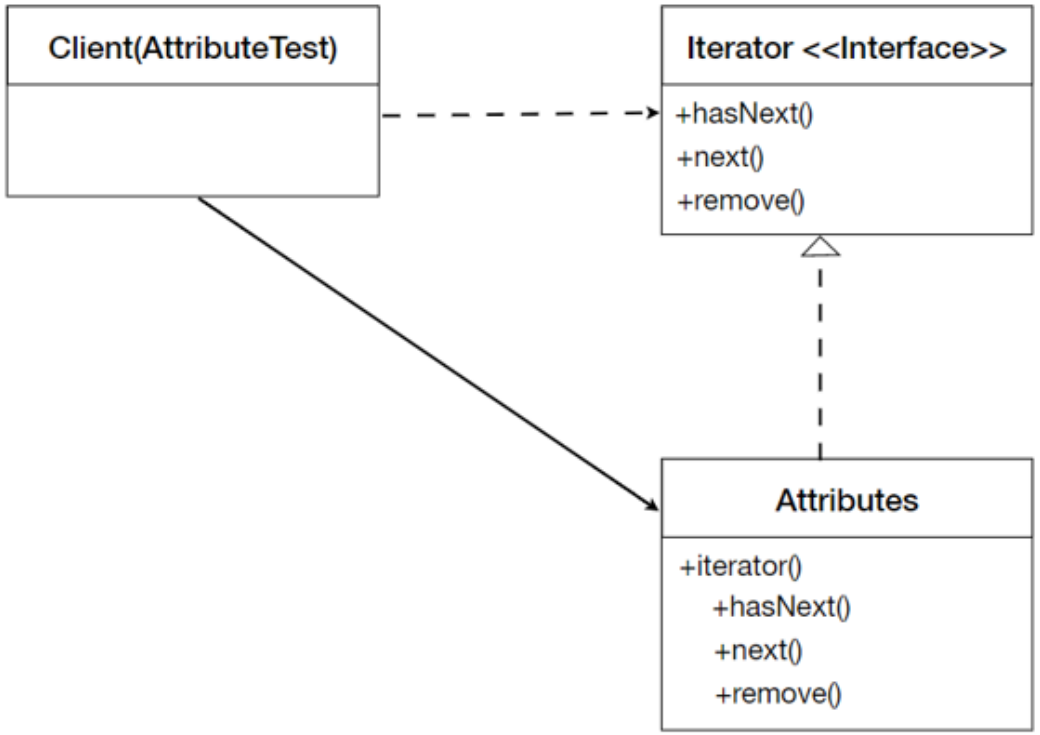


Attributes class가 Iterator interface를 implement 한다.

(hasNext(), next(), remove())

Jsoup Designpattern

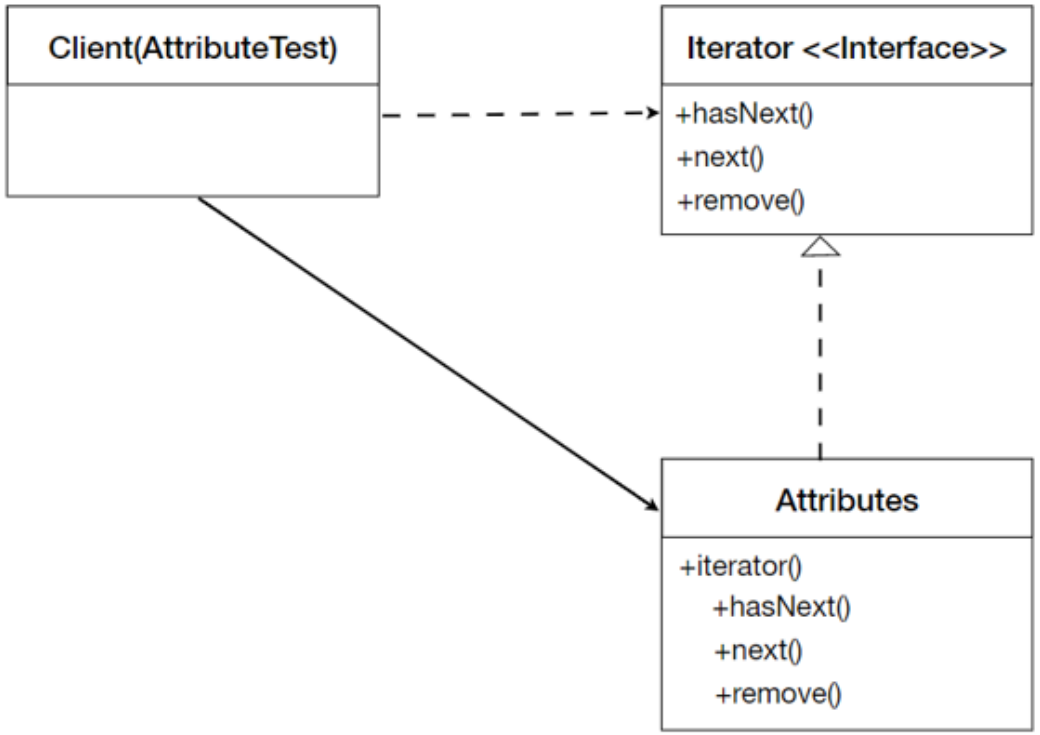
Applied Pattern (iterator pattern)



Attributes class의 iterator()는 iterator의 method를 override하는 동시에 create한다.

Jsoup Designpattern

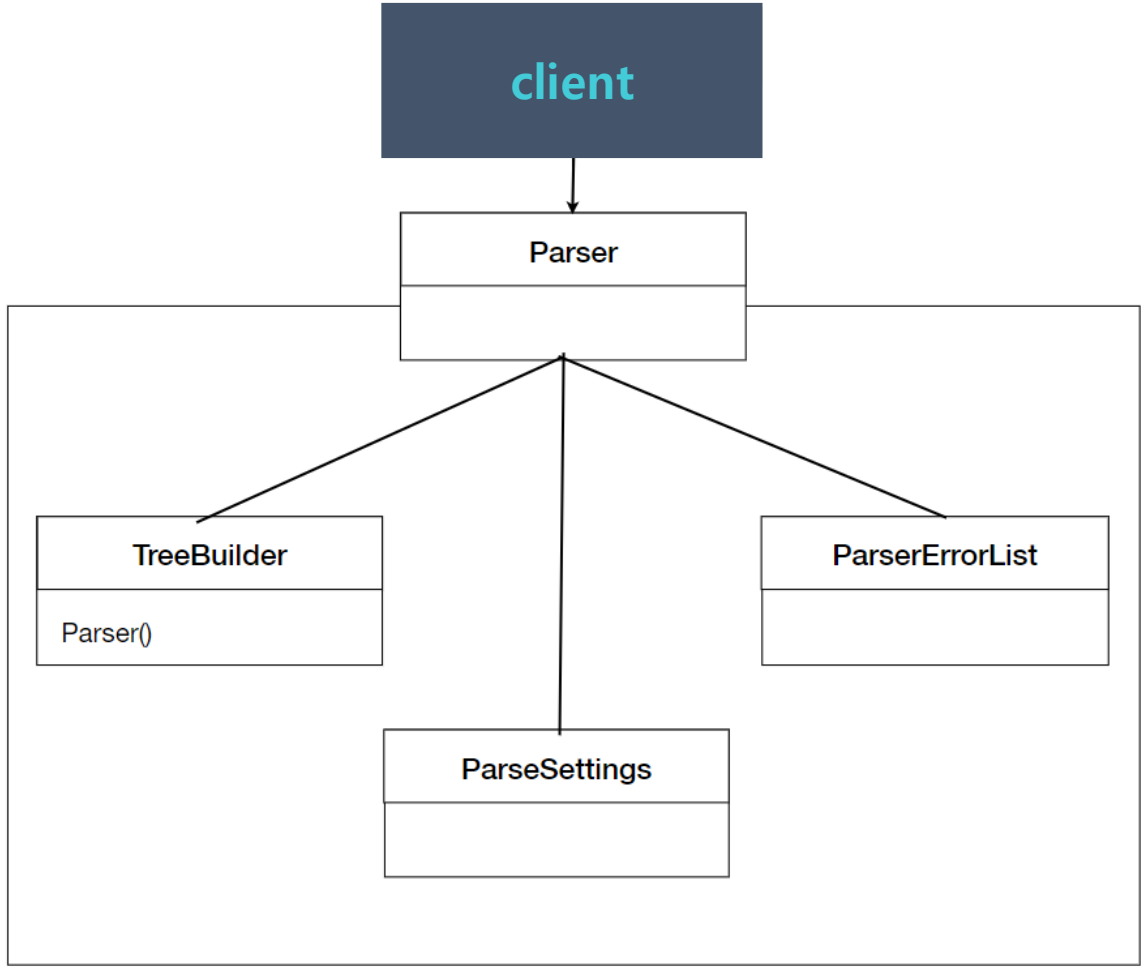
Applied Pattern (iterator pattern)



AttributeTest class에서 Attribute 객체를 사용하여 implement한 iterator를 사용한다.

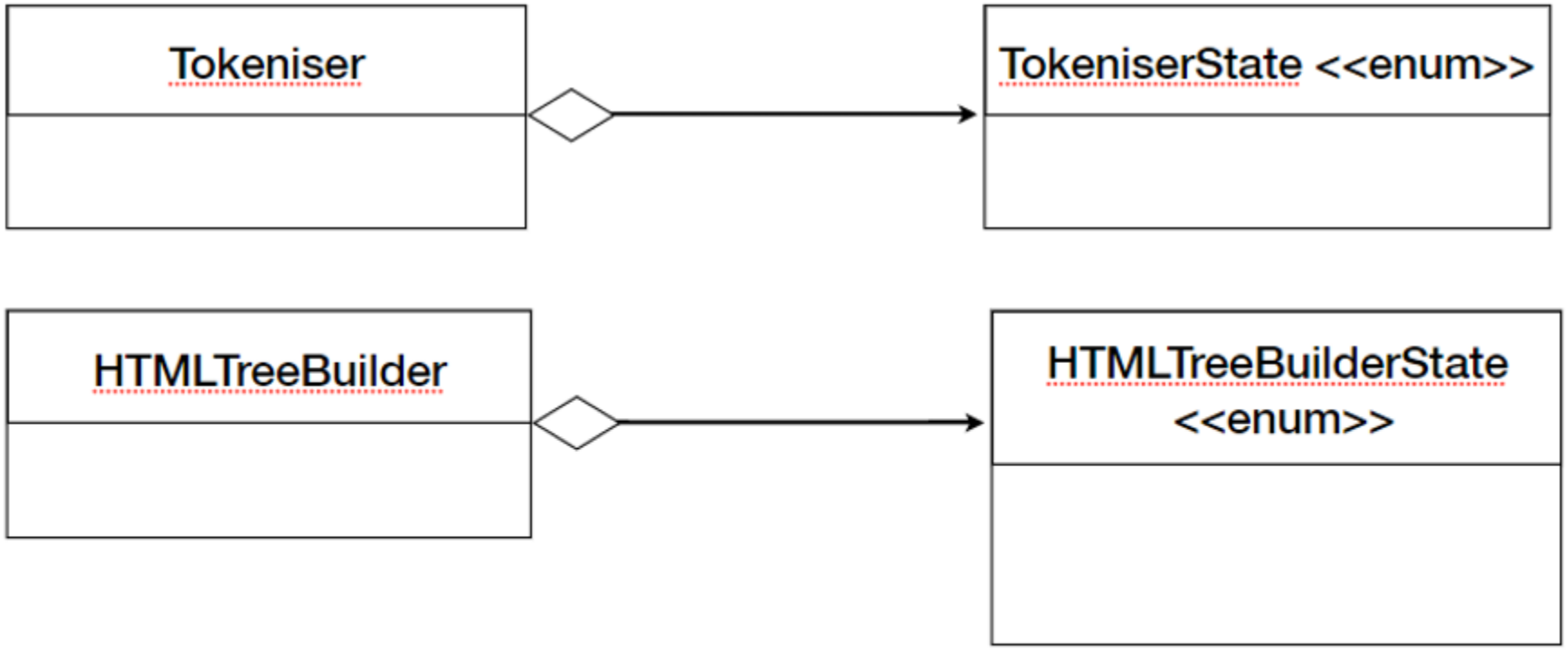
Jsoup Designpattern

Applied Pattern (Façade Pattern)



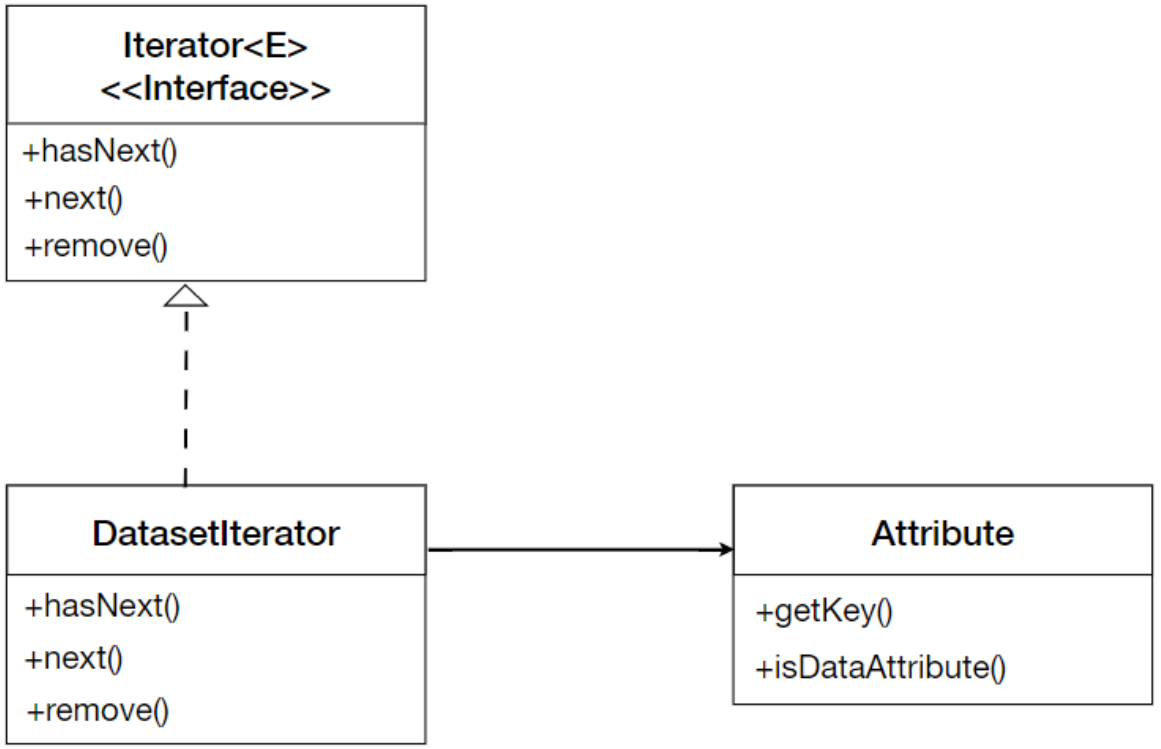
Jsoup Designpattern

Applied Pattern (State Pattern)



Jsoup Designpattern

Applied Pattern (Adaptor Pattern)



Jsoup Designpattern

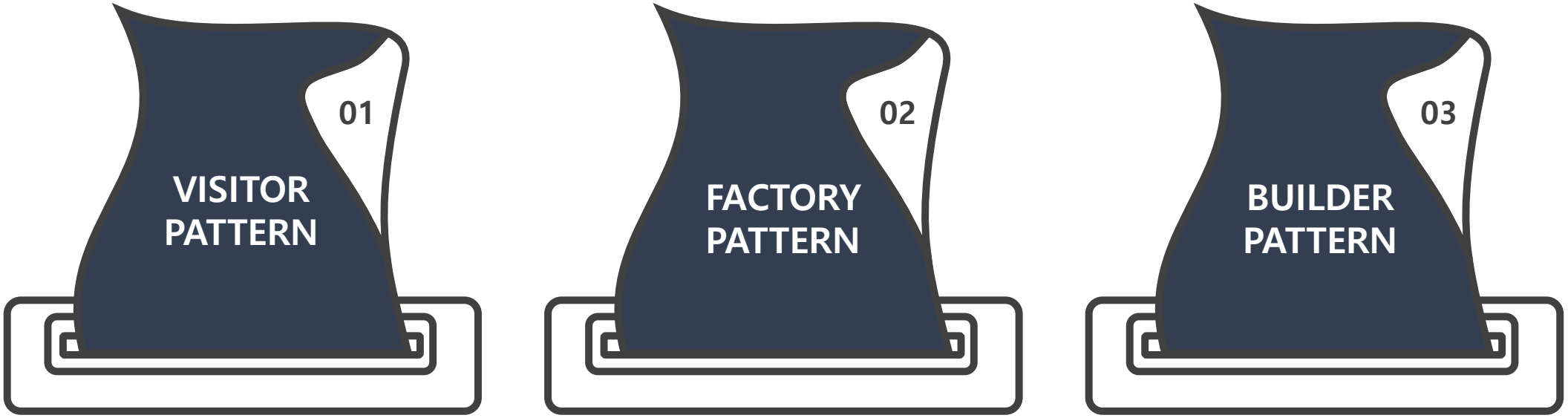
Applied Pattern (Adaptor Pattern)

```
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()); }
}
```

Implement additional pattern & function

Jsoup Designpattern

Additional Pattern

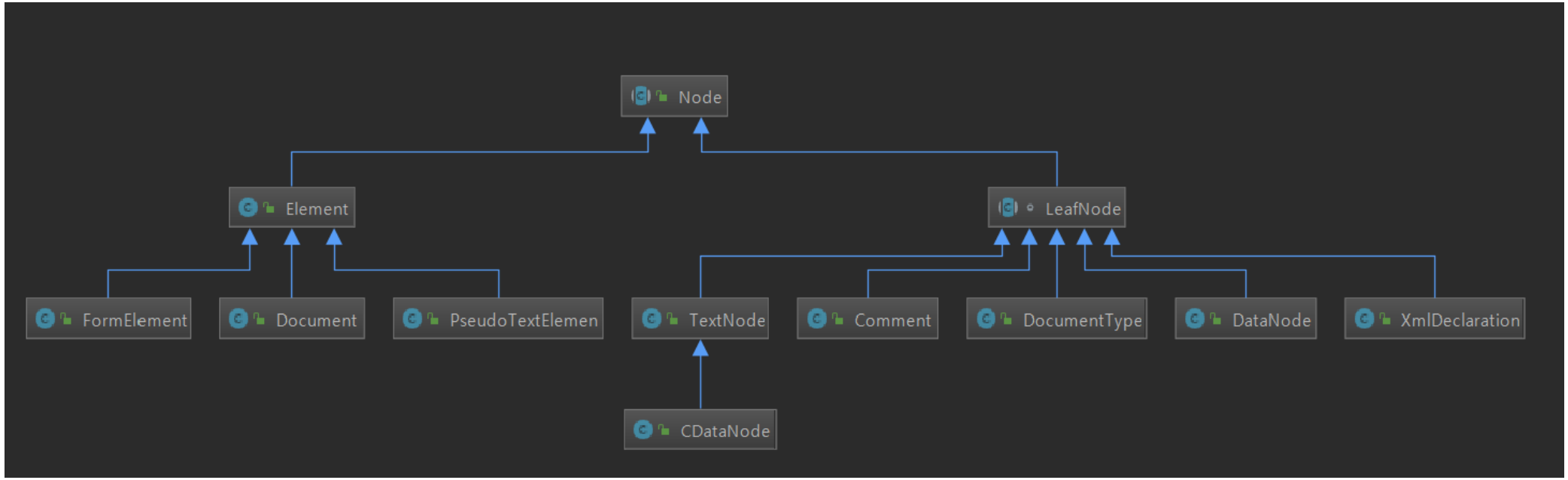


Jsoup Designpattern

Additional Pattern (Visitor Pattern)

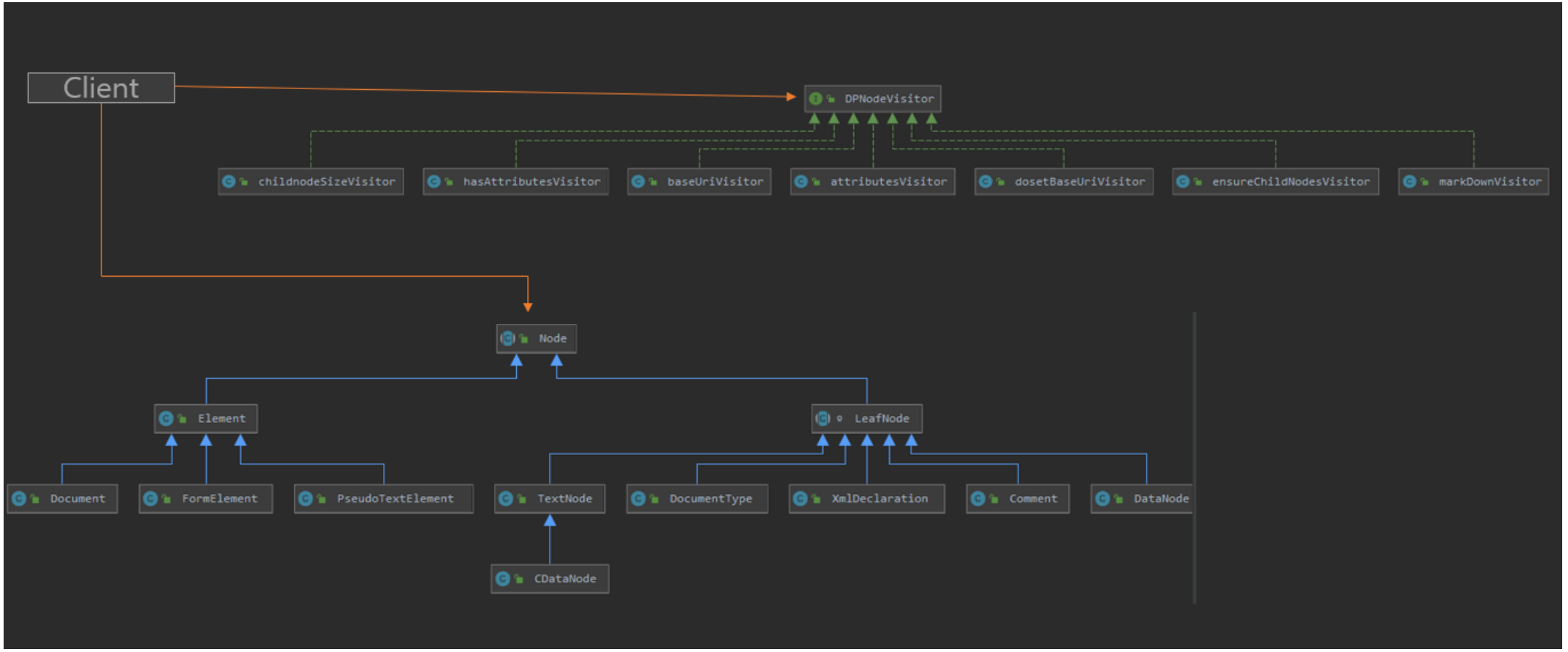
Jsoup에서 NODE라는 abstract class를 Element와 Leafnode라는 두가지의 class가 상속을 받아 해당 abstract method를 구현한다.

또한, Element , Leafnode class 를 상속을 받아 다양한 concrete한 class가 만들어진다.



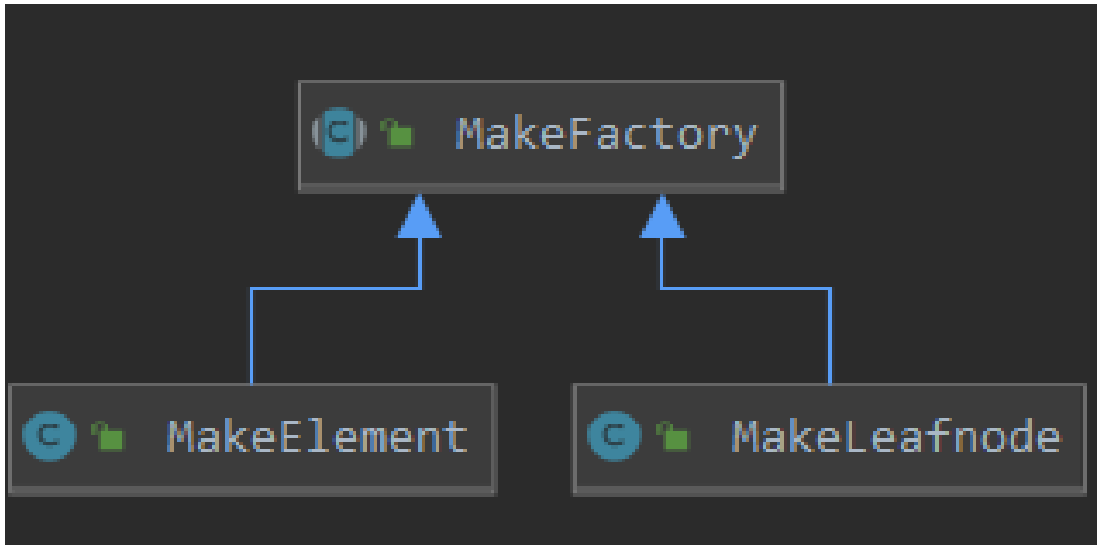
Jsoup Designpattern

Additional Pattern (Visitor Pattern)



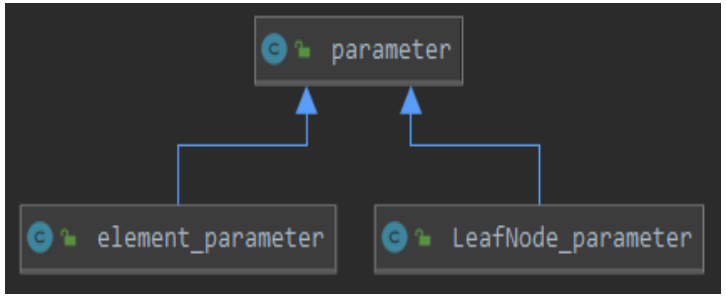
Jsoup Designpattern

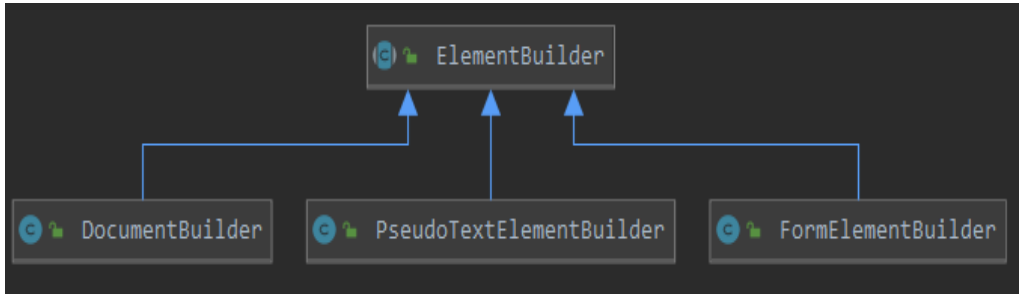
Additional Pattern (factory Pattern)

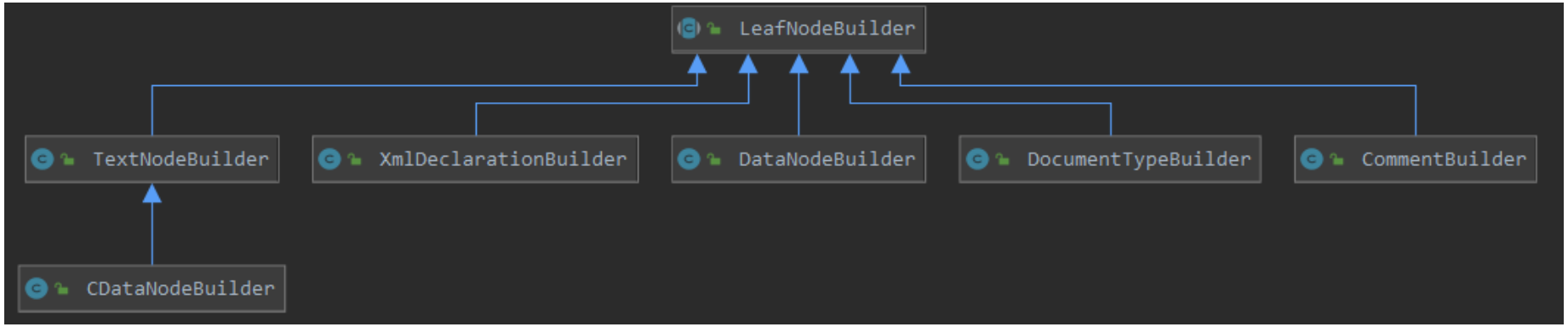


Jsoup Designpattern

Additional Pattern (Builder Pattern)







Jsoup Designpattern

Additional Function

Mark Down

Html ->

DOM ->

MarkDown

(visitor pattern)

Image node

additional LeafNode

for save Image.

(Factory & Builder)

Jsoup Designpattern

Additional function (MarkDown)

COOKIETEST.md — C:₩Users₩JIMIN₩Desktop₩designpattern-master₩designpattern-master₩jsoup-master — Atom

File Edit View Selection Find Packages Help

COOKIETEST.md

```
#초코 공룡쿠기 만들기
####간편하지만 정성이 가득한 만들기 세트를 만나보세요.

![](https://i.imgur.com/OHAZxMa.jpg)
###재료
* 하트쿠키믹스
* 달걀 노른자 1개
* 버터 40g
* 우유 20ml
* 다크커버처초콜릿 30g
* 초코펜(화이트/3개)
* 공룡쿠키커터 3종세트
<br/>

1. 볼에 실온상태의 버터, 달걀노른자, 우유, 녹인초콜릿, 하트쿠키믹스를 넣고 핸드믹서 거품기로30초 섞어주세요.
1. 반죽을 비닐에 넣고 손으로 눌러가며 한 덩어리로 뭉친 다음 냉장고에서 30분 이상 휴지시켜주세요.
1. 반죽을 두께 3mm정도가 되도록 밀대로 얇게 밀어 펴주세요.
1. 쿠키커터로 공룡 모양을 찍어준 뒤, 180도로 예열한 오븐에서 10-12분간 구워 식혀준뒤, 초코펜으로 장식해주면 완성!

---------------------------------------------------------------------------<br/>


태윤:공룡쿠키잘만들었어요^^
```

COOKIETEST.md preview

재료

- 하트쿠키믹스
- 달걀 노른자 1개
- 버터 40g
- 우유 20ml
- 다크커버처초콜릿 30g
- 초코펜(화이트/3개)
- 공룡쿠키커터 3종세트

1. 볼에 실온상태의 버터, 달걀노른자, 우유, 녹인초콜릿, 하트쿠키믹스를 넣고 핸드믹서 거품기로30초 섞어주세요.
2. 반죽을 비닐에 넣고 손으로 눌러가며 한 덩어리로 뭉친 다음 냉장고에서 30분 이상 휴지시켜주세요.
3. 반죽을 두께 3mm정도가 되도록 밀대로 얇게 밀어 펴주세요.
4. 쿠키커터로 공룡 모양을 찍어준 뒤, 180도로 예열한 오븐에서 10-12분간 구워 식혀준뒤, 초코펜으로 장식해주면 완성!

태윤:공룡쿠키잘만들었어요^^

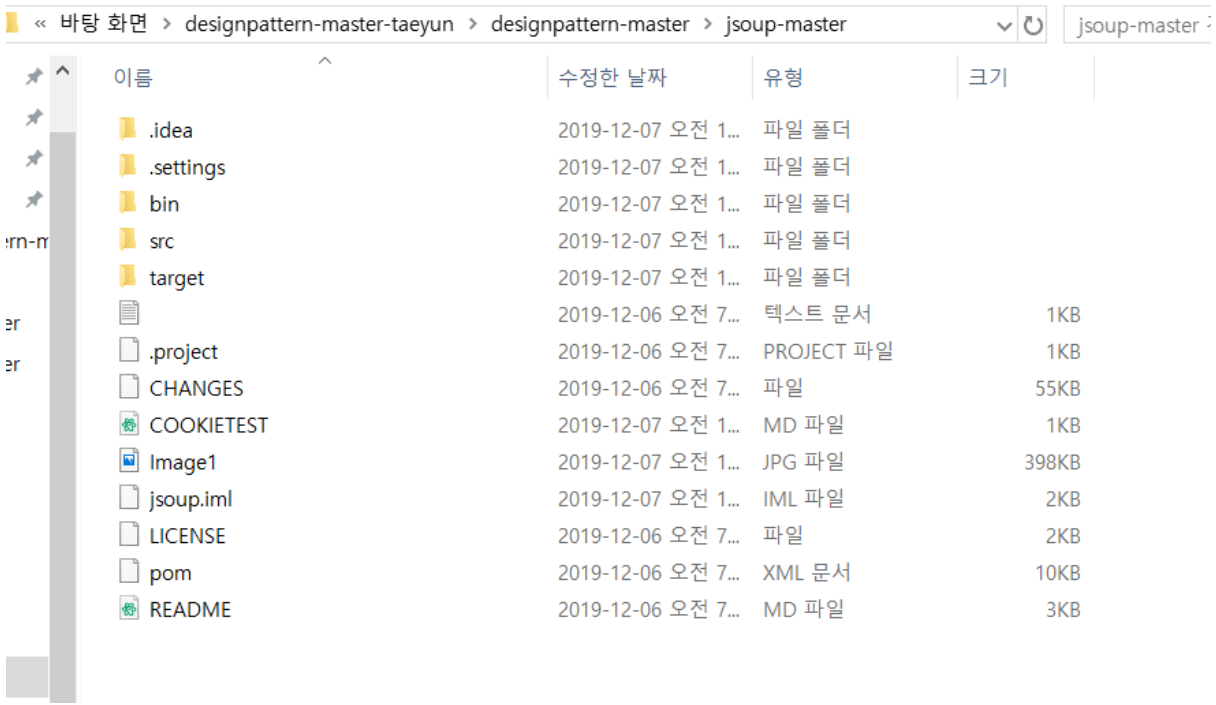
Jsoup Designpattern

Additional function (MarkDown)

```
public class markDownVisitor implements DPNodeVisitor{

    @Override
    public Object visit(Element element) {
        if(element.tag.toString() == "h1") {
            return "#" + element.text() + "\n";
        } else if(element.tag.toString() == "h2") {
            return "##" + element.text() + "\n";
        } else if(element.tag.toString() == "h3") {
            return "###" + element.text() + "\n";
        }else if(element.tag.toString() == "h4") {
            return "####" + element.text() + "\n";
        }else if(element.tag.toString() == "br") {
            return "<br/>\n\n";
        }else if(element.tag.toString() == "img") {
            return "  \n![]("+element.absUrl( attributeKey: "src").toString()+")\n";
        }else if(element.tag.toString() == "li") {
            if(element.parentNode().toString().startsWith("<ol>"))
                return "* " + element.text() + "\n";
            else if(element.parentNode().toString().startsWith("<ul>"))
                return "1. " + element.text() + "\n";
            return null;
        }
```

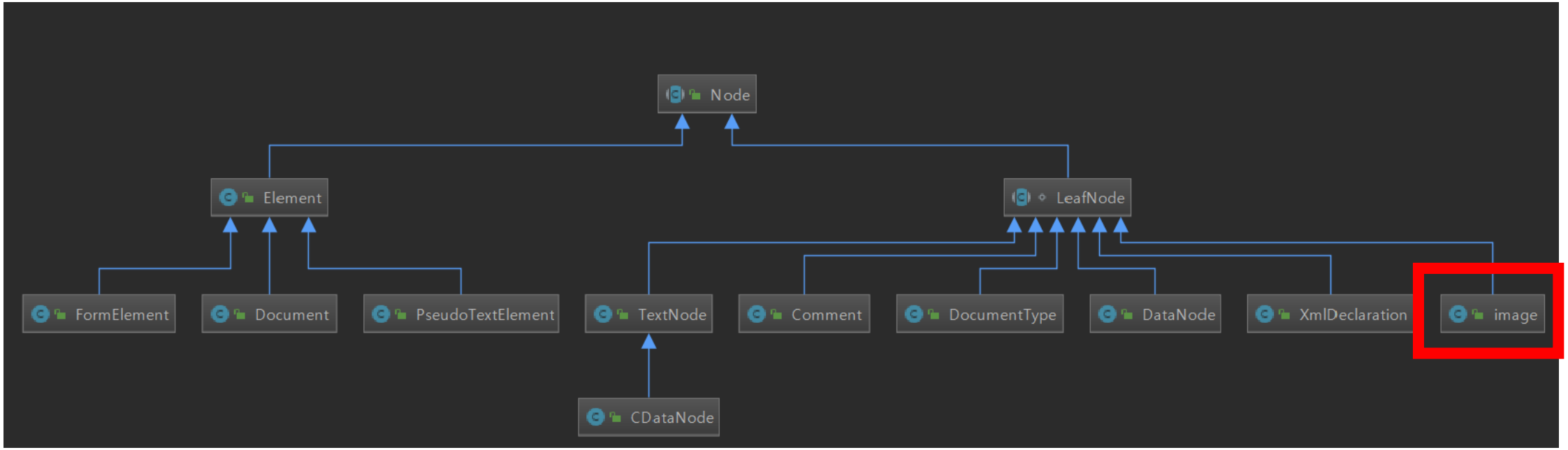
Additional function (MarkDown)

```
doc.markdown("COOKIETEST.md");
```



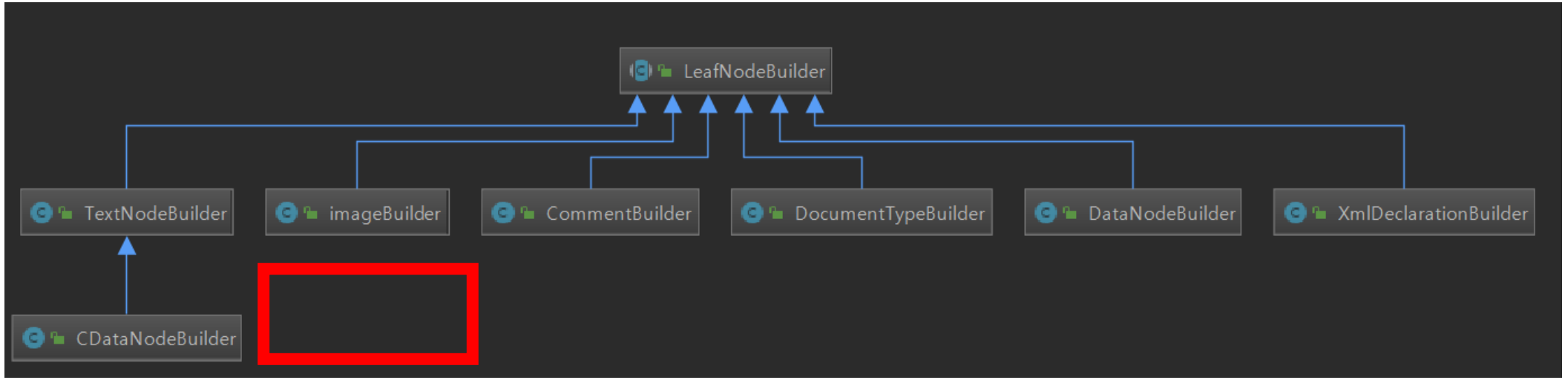
Jsoup Designpattern

Additional function (Image Node)



Jsoup Designpattern

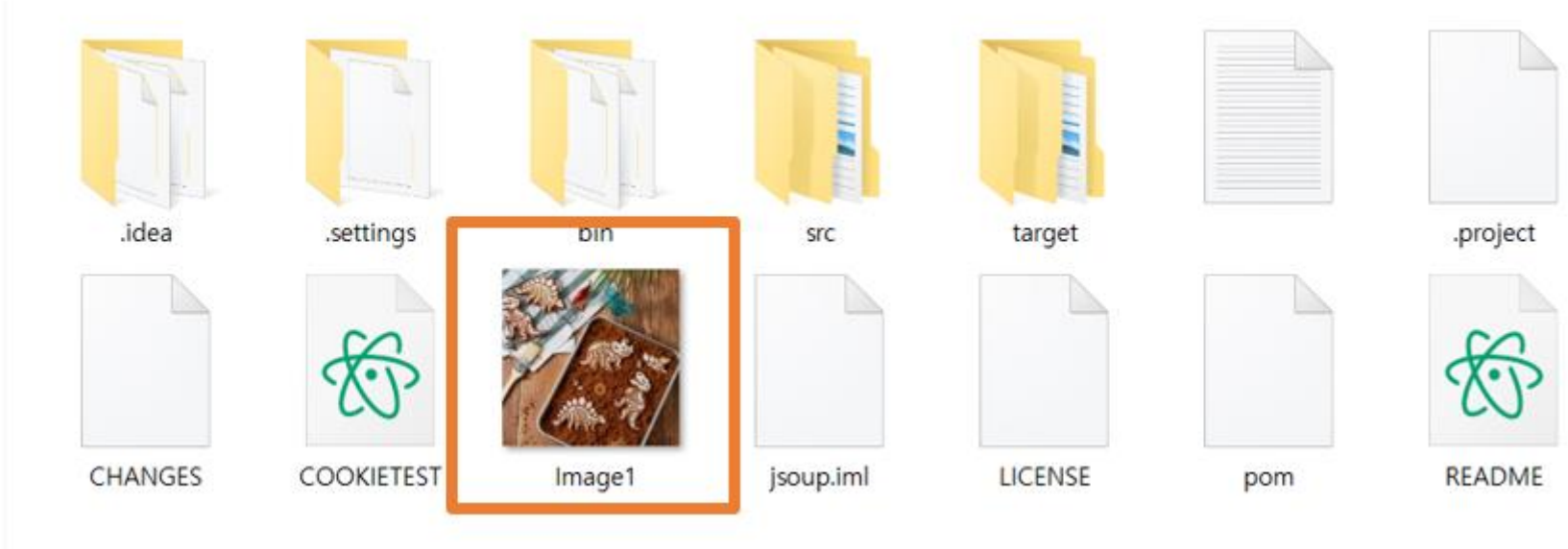
Additional function (Image Node)




```
if(t.tagPending.tagName.equals("img")){

    LeafNodeDirector leaf = new LeafNodeDirector();
    LeafNodeBuilder image = new imageBuilder( _type: "image",value.toString());
    leaf.setLeafNodeBuilder(image);
    leaf.constructparameter();
    LeafNode_parameter params = leaf.getelement();

    MakeLeafnode factory = new MakeLeafnode();
    org.jsoup.nodes.image a  = (image) factory.createnode(params);
}
```

Jsoup Designpattern

Additional function (Image Node)



Q & A