KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

Facade - 단순한 창구를 만든다

[KB] IT's Your Life

불어
건물의 정면

KB국민은행

Façade 패턴

Façade(파사드) 패턴

겉보이는건 간단

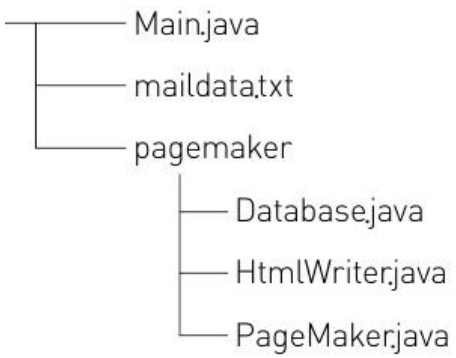
- 건물의 정면이라는 뜻
  - 뒤에 건물의 복잡한 구조가 있음
- 많은 객체들이 복잡하게 얽혀서 너저분한 세부 내용을 정리하여 높은 수준의 인터페이스(API)를 제공
- 시스템 외부에 간단한 인터페이스(API)를 보여줌
- 시스템 내부의 각 클래스의 역할과 의존 관계를 고려하여 올바른 순서로 클래스를 사용하게 함

Façade 패턴

예제 프로그램

- 사용자의 웹 페이지를 작성하는 것
- 각 클래스의 역할

패키지	이름	설명
pagemaker	Database	이메일 주소에서 사용자 이름을 얻는 클래스
pagemaker	HtmlWriter	HTML 파일을 작성하는 클래스
pagemaker	PageMaker	이메일 주소로 사용자의 웹 페이지를 작성하는 클래스
이름 없음	Main	동작 테스트용 클래스

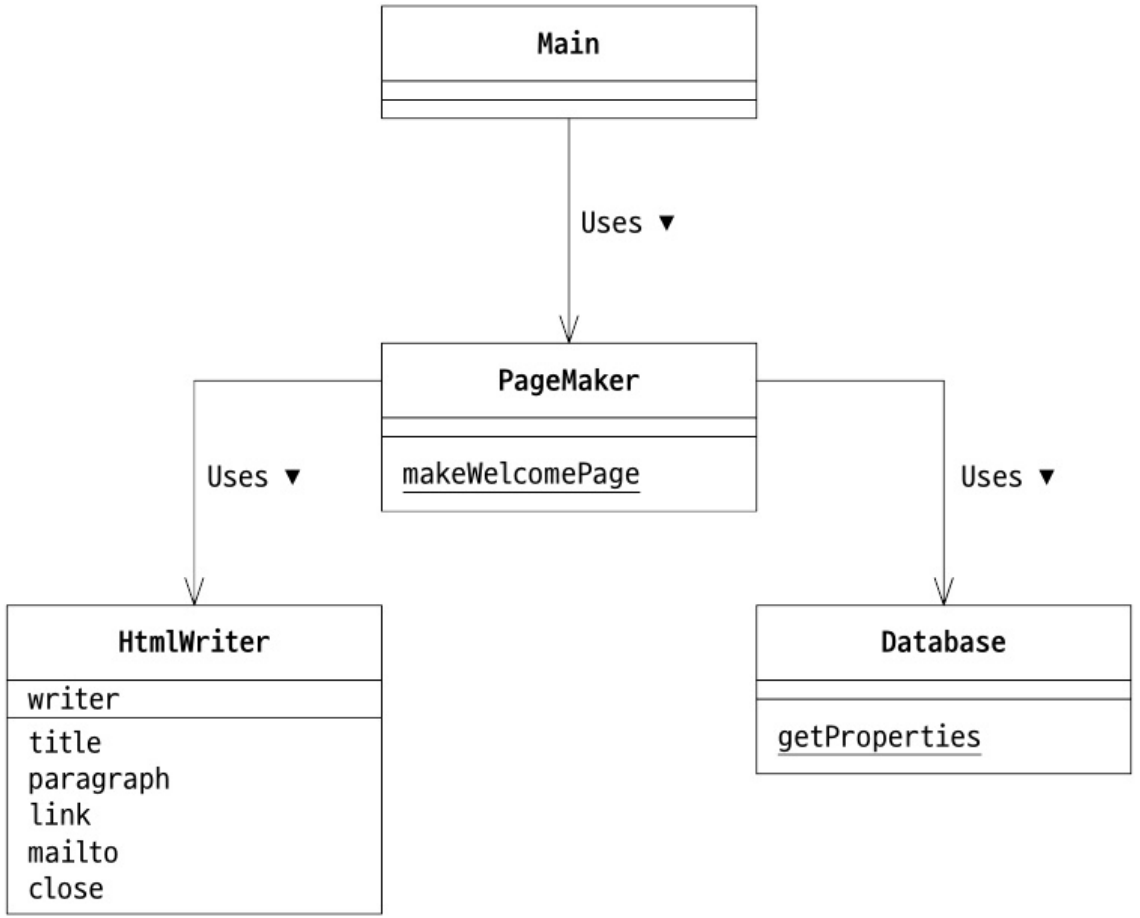


이 예제는 SRP라고 가정하고

Façade 패턴

예제 프로그램의 클래스 다이어그램

우리가 만들 클래스들의 다이어그램



maildata.txt

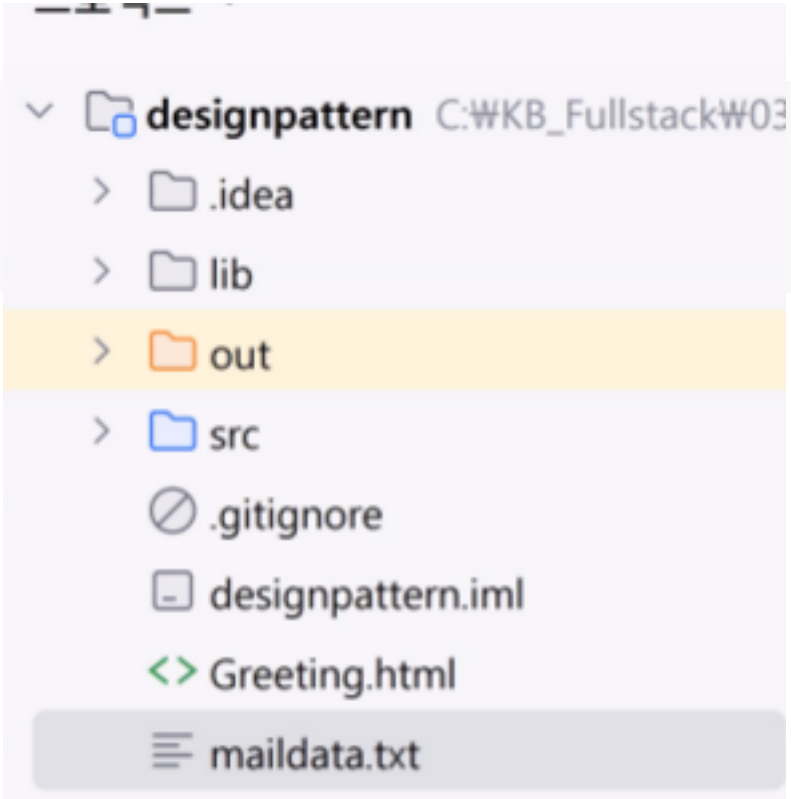
```
hyuki@example.com=Hiroshi Yuki
youngjin@example.com=Kim Youngjin
dojun@example.com=dojun
gildong@example.com=Hong Gildong
```

공백 넣지마라
공백도 데이터로 간주된다

키 밸류 타입이네

자바에서 properties파일이라고 불리죠

상대경로로 접근 예정.
루트경로에 만들어 해당 데이터



Façade 패턴

pagemaker/Database.java

```
public class Database {
  private Database() {
  }

  // 데이터베이스 이름에서 Properties를 얻는다
  public static Properties getProperties(String dbname) throws IOException {
    String filename = dbname + ".txt";
    Properties prop = new Properties();
    prop.load(new FileReader(filename));
    return prop;
  }
}
```

프로퍼티들을 쉽게 얻게 하는 facade 패턴.

api하나 만들어진 것.

Façade 패턴

pagemaker/HtmlWriter.java

```
public class HtmlWriter {
  private Writer writer;

  public HtmlWriter(Writer writer) {
    this.writer = writer;
  }

  // 타이틀 출력
  public void title(String title) throws IOException {
    writer.write("<!DOCTYPE html>");
    writer.write("<html>");
    writer.write("<head>");
    writer.write("<title>" + title + "</title>");
    writer.write("</head>");
    writer.write("<body>");
    writer.write("\n");
    writer.write("<h1>" + title + "</h1>");
    writer.write("\n");
  }

  // 단락 출력
  public void paragraph(String msg) throws IOException {
    writer.write("<p>" + msg + "</p>");
    writer.write("\n");
  }
```

어떤 라이터인지 전달 받고

둘다 퍼사드 패턴!

무언갈 만들기 위해 복잡한 과정들을
메서드로 묶어서
재사용성 캡슐화된 api ㅓ로 만듬

->
복잡한 것을 단순화

계속 일일이 출력하면 느릴텐데. 다 조립하고(빌더로)
한꺼번에 한번 출력하는게 좋을 텐데.

어디다가 쓸건지 여기서 정하지 않음.
생성자로 받은 라이터에서 정해짐.
바깥쪽에서 지정함. -> 이 패턴이 중요함.
여기서 라이터를 만들며 경로를 지정하면 안됨.
->여기서 만들면 기능이 제한됨.
->밖에서 여러 종류의 라이터를 받게 한다면
폭넓게 사용할 수 있음

pagemaker/HtmlWriter.java

```
  // 링크 출력
  public void link(String href, String caption) throws IOException {
    paragraph("<a href=\"" + href + "\">" + caption + "</a>");
  }

  // 이메일 주소 출력
  public void mailto(String mailaddr, String username) throws IOException {
    link("mailto:" + mailaddr, username);
  }

  // HTML 닫기
  public void close() throws IOException {
    writer.write("</body>");
    writer.write("</html>");
    writer.write("\n");
    writer.close();
  }
}
```

다 단순화

다 퍼사드 패턴

늘상 하던것들인데
퍼사드 패턴인가
애초에 프로그래밍의 대부분이
퍼사드 패턴 아닌가

Façade 패턴

pagemaker/PageMaker.java

```
public class PageMaker {
  private PageMaker() {
  }

  public static void makeWelcomePage(String mailaddr, String filename) {
    try {
      Properties mailprop = Database.getProperties("maildata");
      String username = mailprop.getProperty(mailaddr);
      HtmlWriter writer = new HtmlWriter(new FileWriter(filename));

      writer.title(username + "'s web page");
      writer.paragraph("Welcome to " + username + "'s web page!");
      writer.paragraph("Nice to meet you!");
      writer.mailto(mailaddr, username);
      writer.close();
      System.out.println(filename + " is created for " + mailaddr + " (" + username + ")");
    } catch (IOException e) {
      e.printStackTrace();
    }
  }
}
```

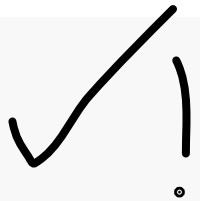
얘도 퍼사드임 api임

모두 api들이지
퍼사드 패턴이지

앞에서 말했듯이
바깥에서 라이터를
생성하며
경로를 지정함.
파일라이터(
파일에서 입력받는)
말고
스트링라이터나
바이트아웃풋스트림처럼
메모리에서 작업하는
것을 사용하면
더 빠르게쬬?

Main.java

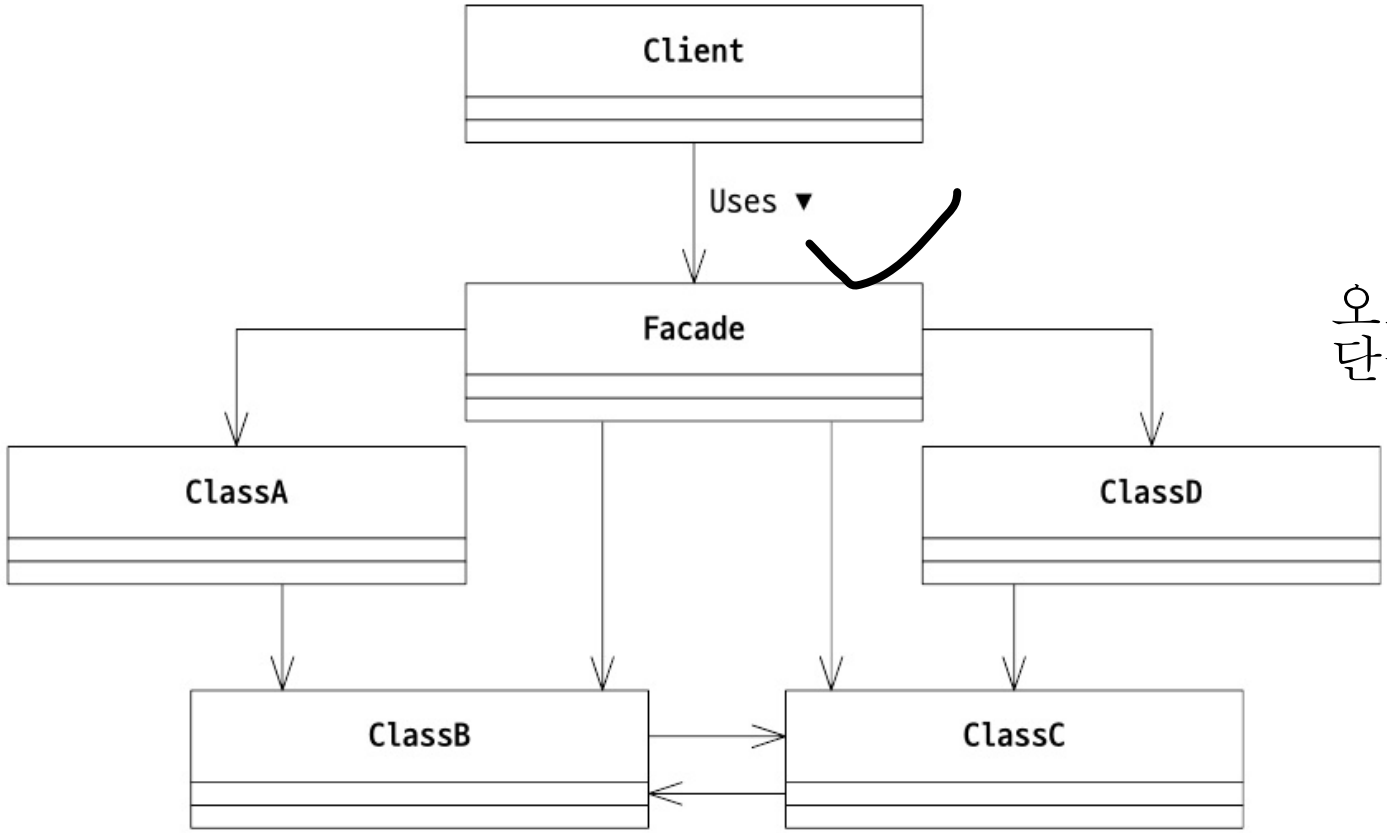
```
public class Main {
  public static void main(String[] args) {
    PageMaker.makeWelcomePage("hyuki@example.com", "welcome.html");
  }
}
```



```
welcome.html is created for hyuki@example.com (Hiroshi Yuki)
```

Façade 패턴

Façade 패턴의 클래스 다이어그램



오호
단순하게

Façade 패턴

- **Façade 패턴의 역할**
  - 인터페이스(API) 수를 줄이는 것