

2조

21300492 윤재원

21400611 이 혁

21600767 한현수

21600302 박지호

목차

1. 역할분담
2. 추가한 기능
3. Use Case Diagram
4. 기능별 설명
5. 총 구현 기능
6. 실행
7. 3rd iteration 계획

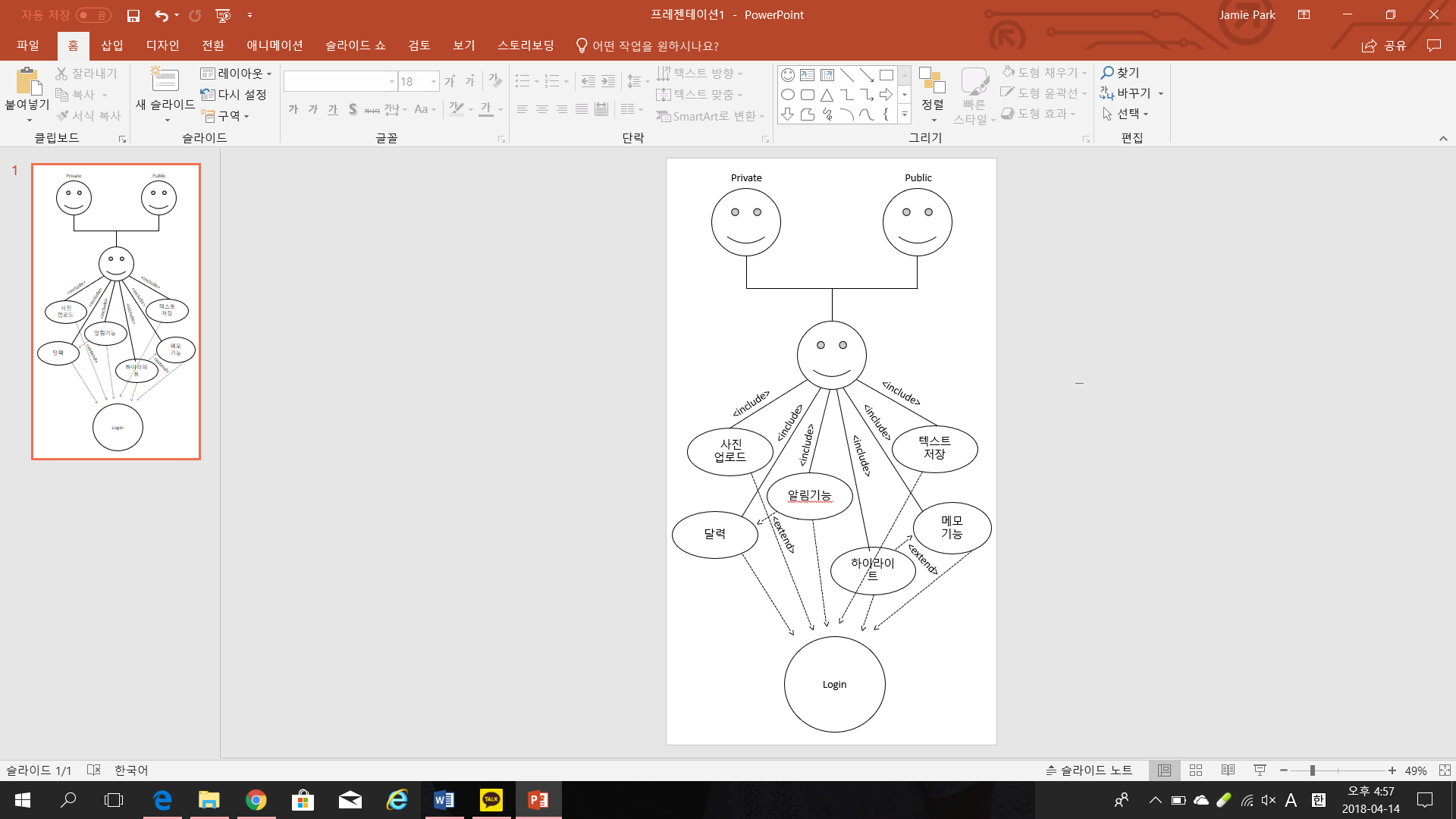
역할분담

* 윤재원 – 앨범 기능 구현 , Class Diagram 작성
* 이 혁 – 달력 기능 구현, ppt및 보고서 작성, 발표
* 한현수 – 텍스트 기능 구현, Class Diagram 작성
* 박지호 – 로그인 기능 구현, UseCase diagram 작성

이번 iteration에서 추가한 기능

* 앨범 슬라이드쇼
* 텍스트 포스팅
* 회원가입
* 날짜 선택

Use Case Diagram



기능별 설명

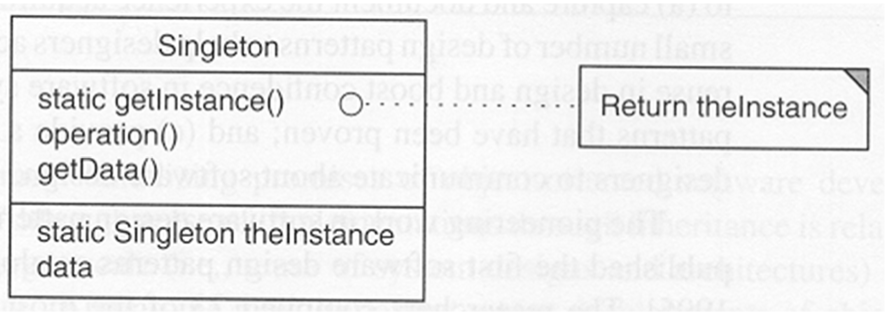
1. 로그인

적용 Design Patter : Singleton Pattern, Template Method Design Pattern, Factory Design Pattern

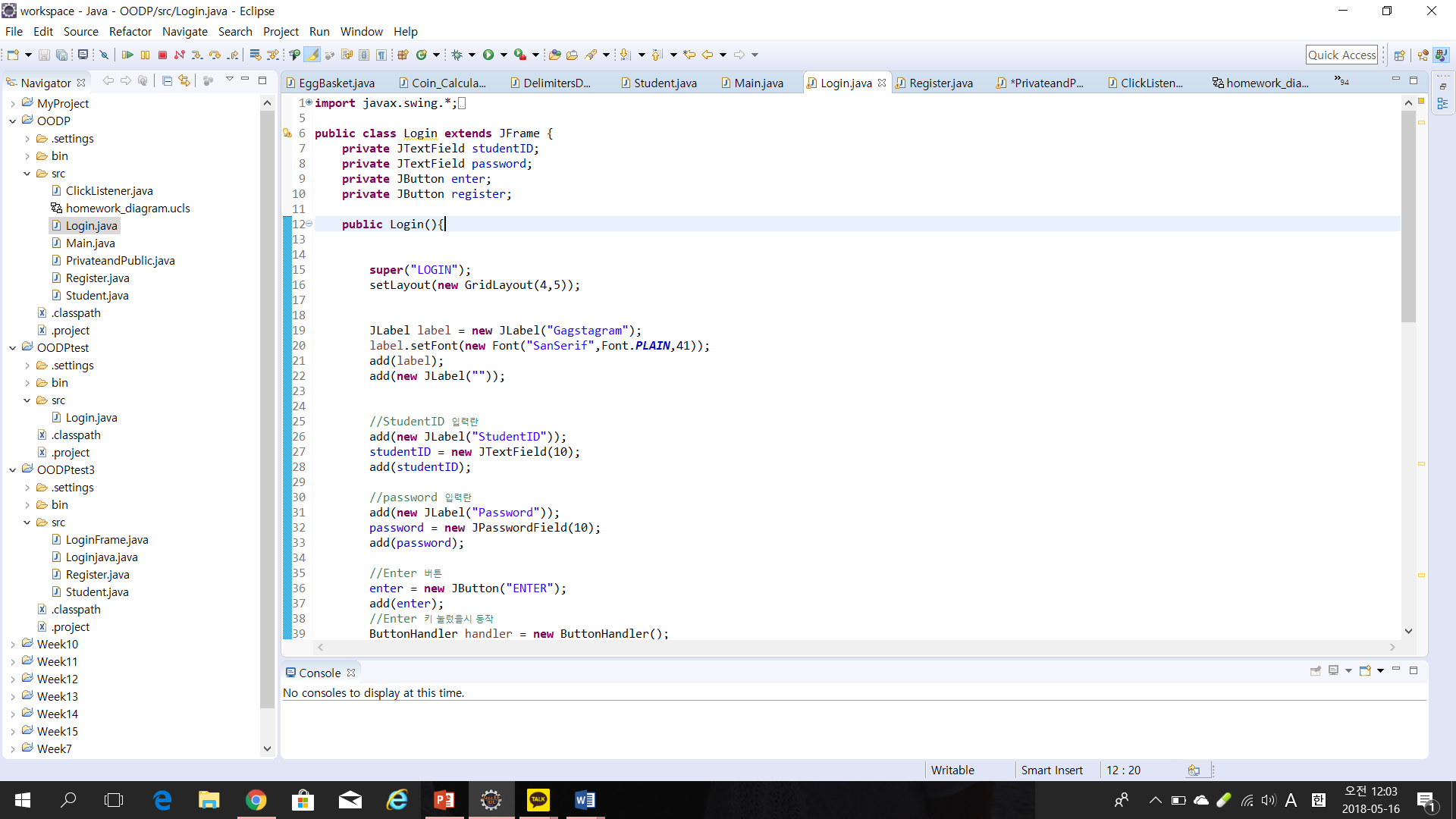
* Singleton Pattern

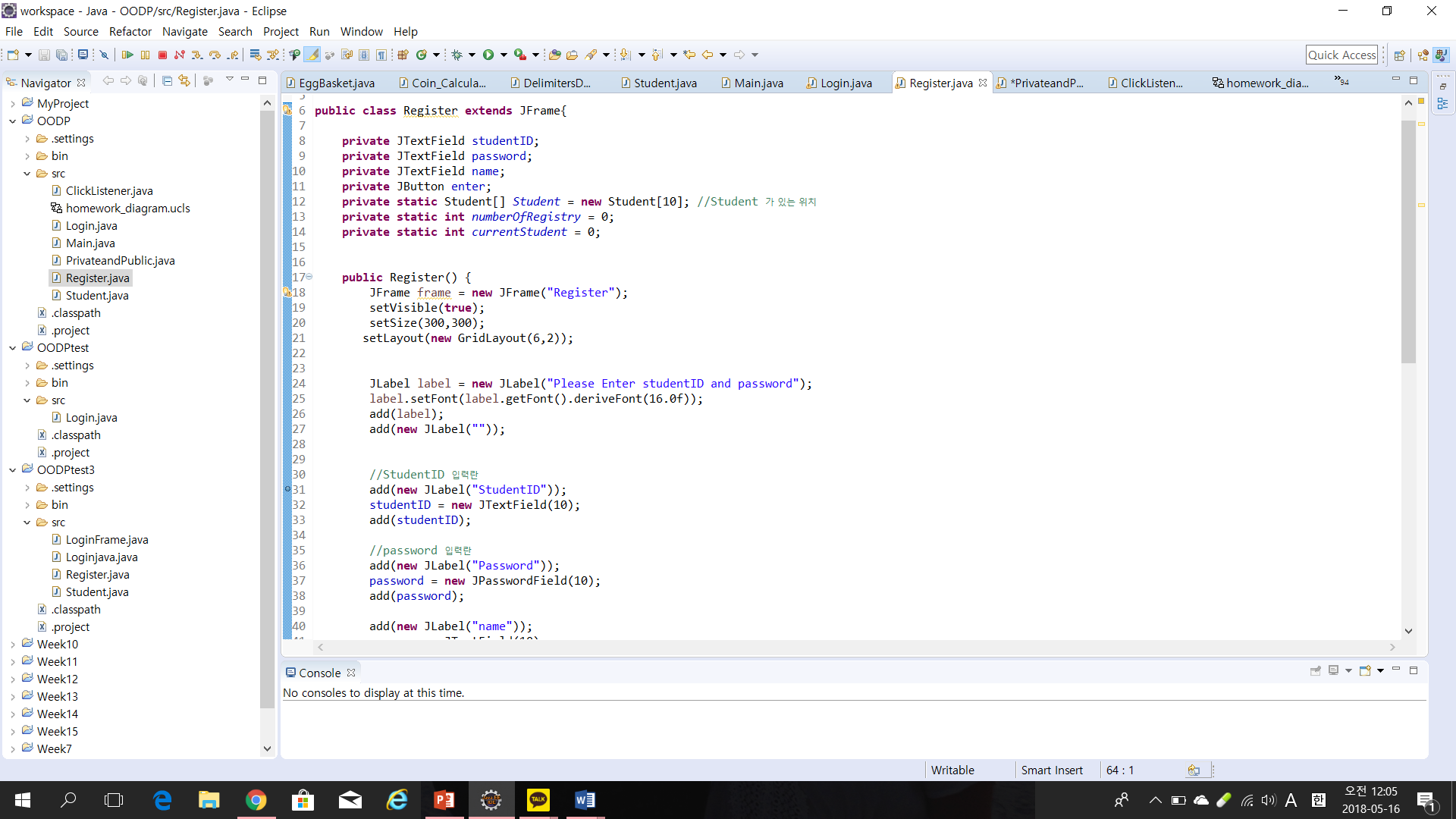
싱클턴 패턴은 해당 클래스의 인스턴스가 하나만 만들어 지고, 어디서든지 그 인스턴스에 접근할 수 있도록 하기 위한 패턴이다.

클래스에서 자신의 단 하나뿐인 인스턴스를 관리하도록 하면 된다. 다른 어떤 클래스에서도 자신의 인스턴스를 추가로 만들지 못하도록 해야 한다. 그리고 어디서든 그 인스턴스에 접근할 수 있도록 만들어야 한다.



적용한 부분

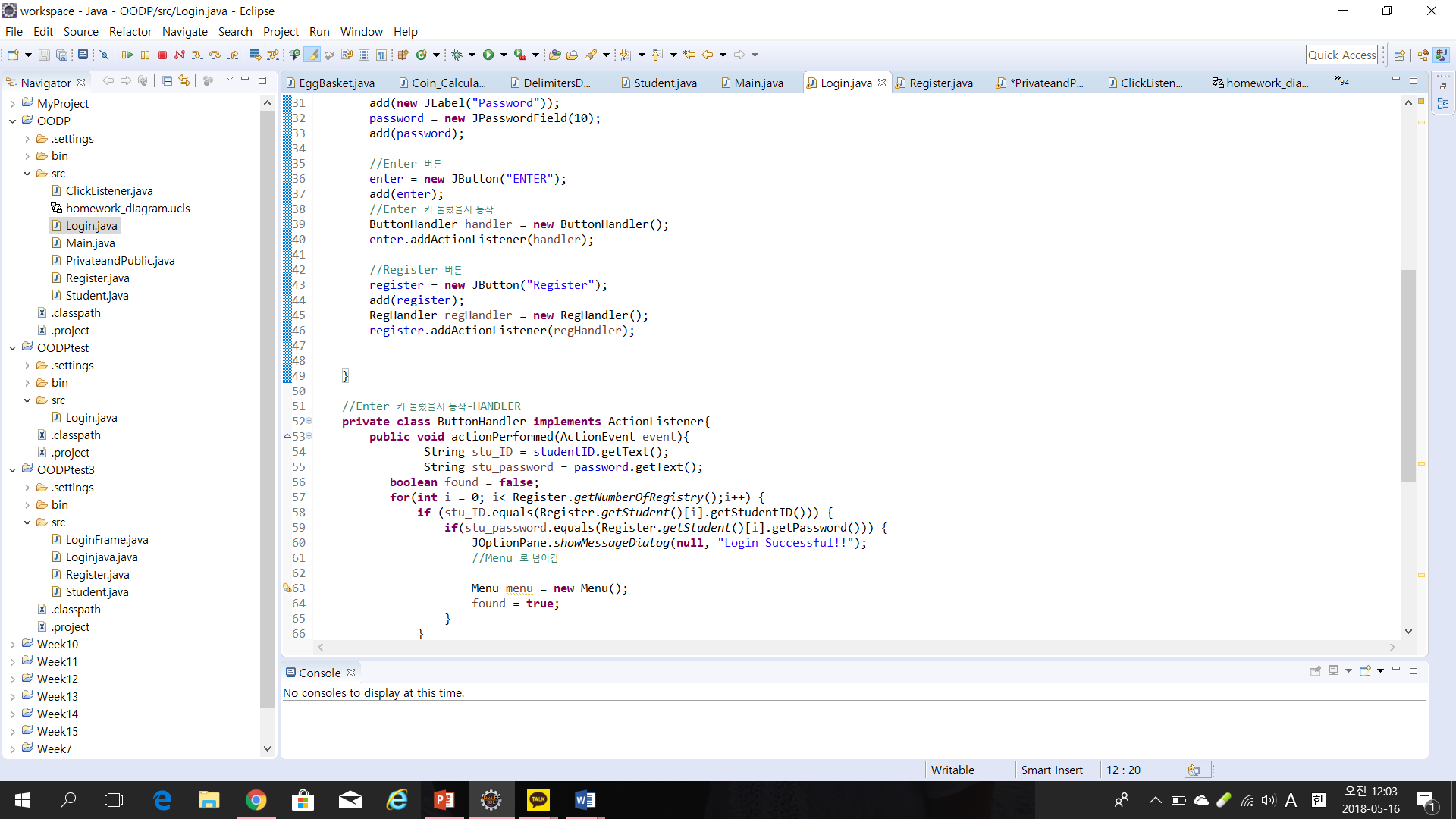




* Template Method Design Pattern

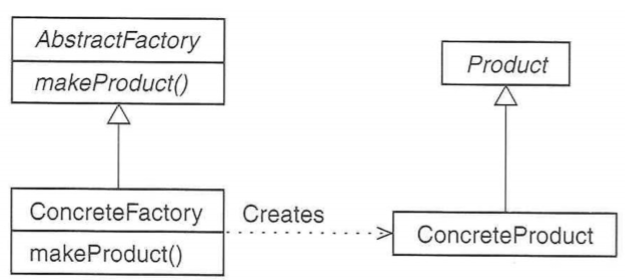
Method에서 알고리즘의 골격을 정의한다. 알고리즘의 여러 단계 중 일부는 서브 클래스에서 구현할 수 있다. 템플릿 메소드를 이용하면 알고리즘의 구조는 그대로 유지하면서 서브 클래스에서 특정 단계를 재정의 할 수 있다. 여러 단계 가운데 하나 이상이 추상 메소드로 정의되며, 그 추상 메소드는 서브클래스에서 구현된다.

적용한 부분

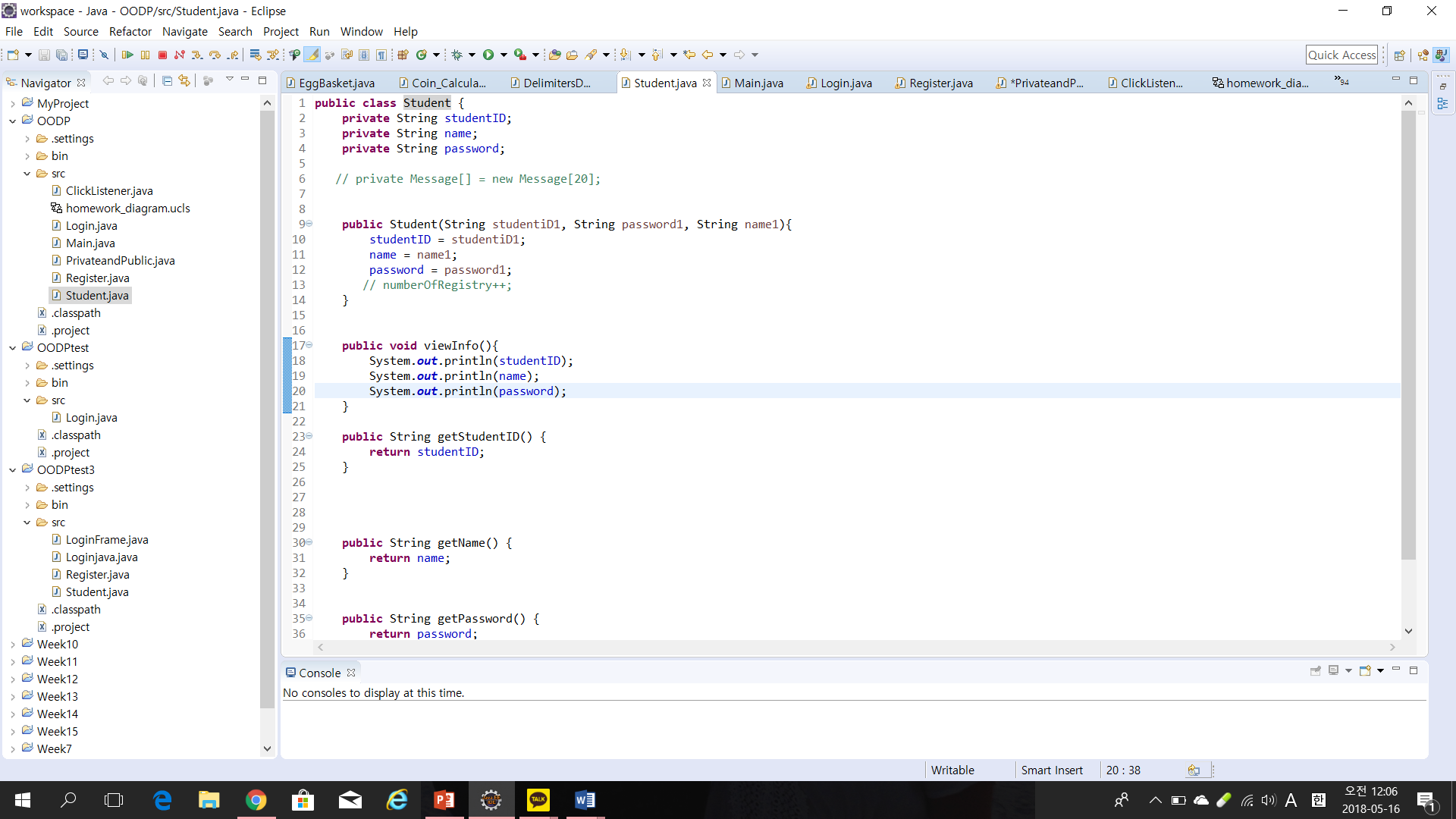


* Factory Design Pattern

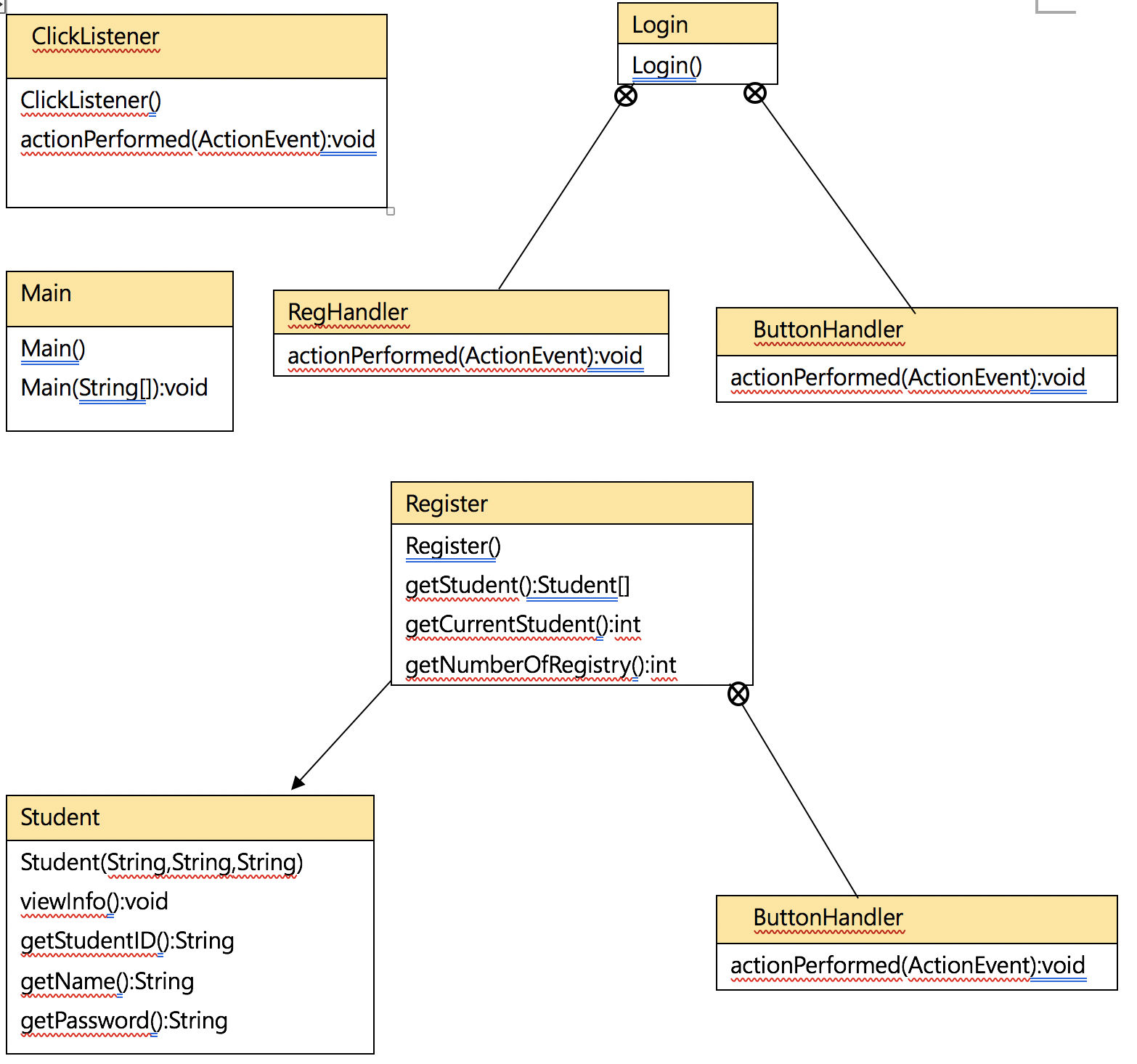
팩토리 메소드 패턴에서는 객체를 생성하기 위한 인터페이스를 정의하는데, 어떤 클래스의 인스턴스를 만드는지는 서브클래스에서 결정하게 만든다. 팩토리 메소드를 이용하면 객체를 생성하는 작업을 서브 클래스에 캡슐화 시킬 수 있다.



적용한 부분



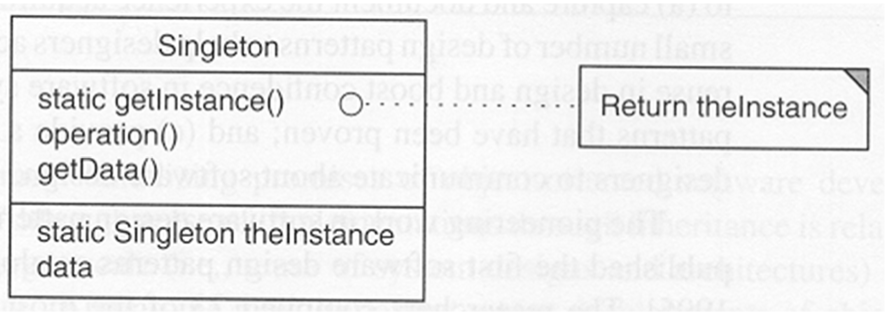
Class Diagram



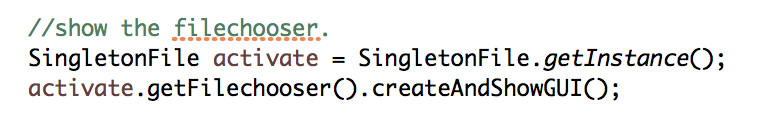
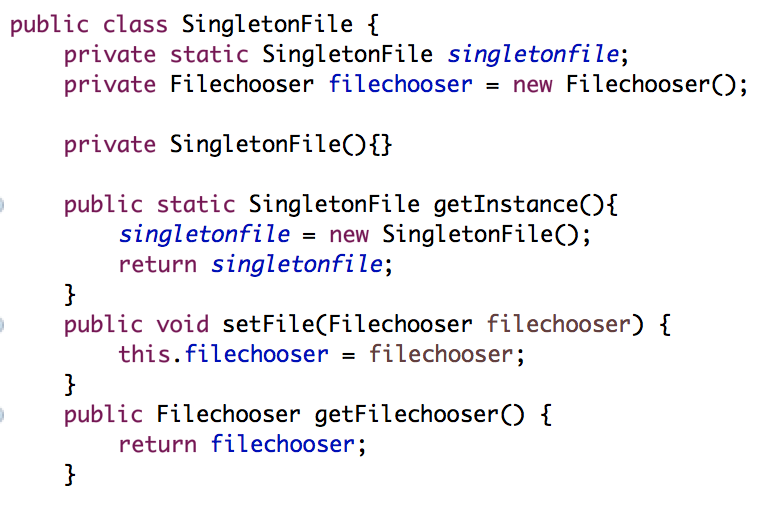
1. 슬라이드쇼

* Singleton Pattern

싱클턴 패턴은 해당 클래스의 인스턴스가 하나만 만들어 지고, 어디서든지 그 인스턴스에 접근할 수 있도록 하기 위한 패턴이다. 클래스에서 자신의 단 하나뿐인 인스턴스를 관리하도록 하면 된다. 다른 어떤 클래스에서도 자신의 인스턴스를 추가로 만들지 못하도록 해야 한다. 그리고 어디서든 그 인스턴스에 접근할 수 있도록 만들어야 한다.

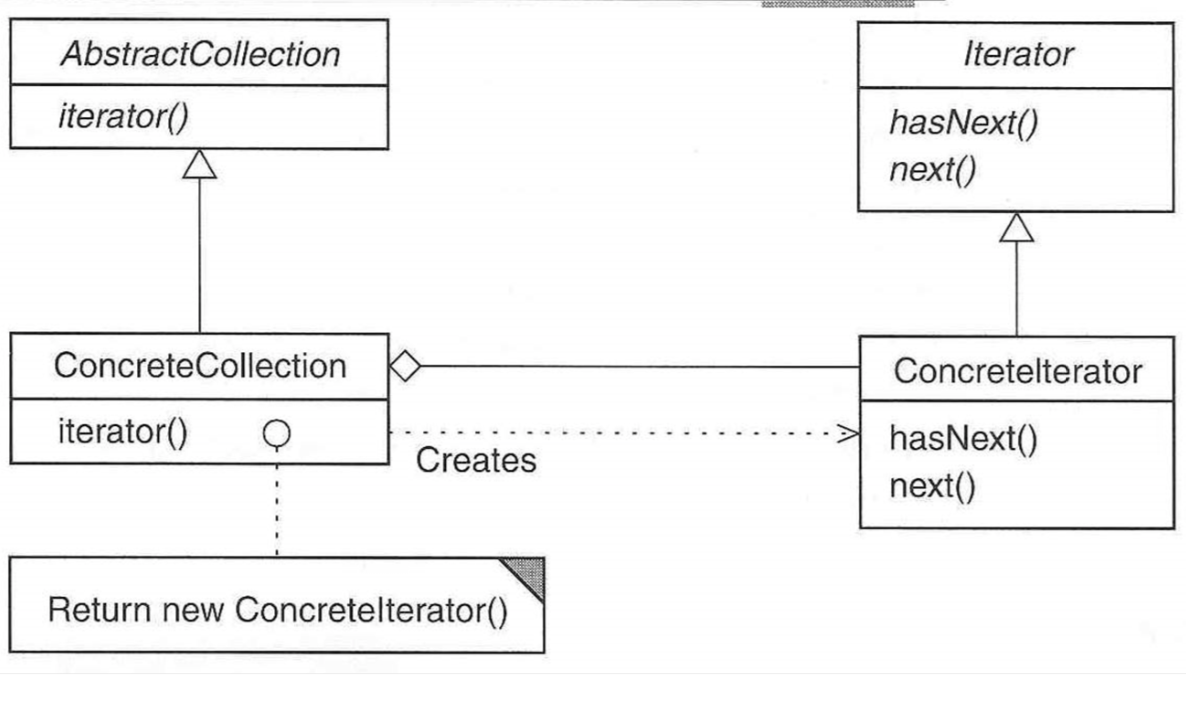


적용한 부분

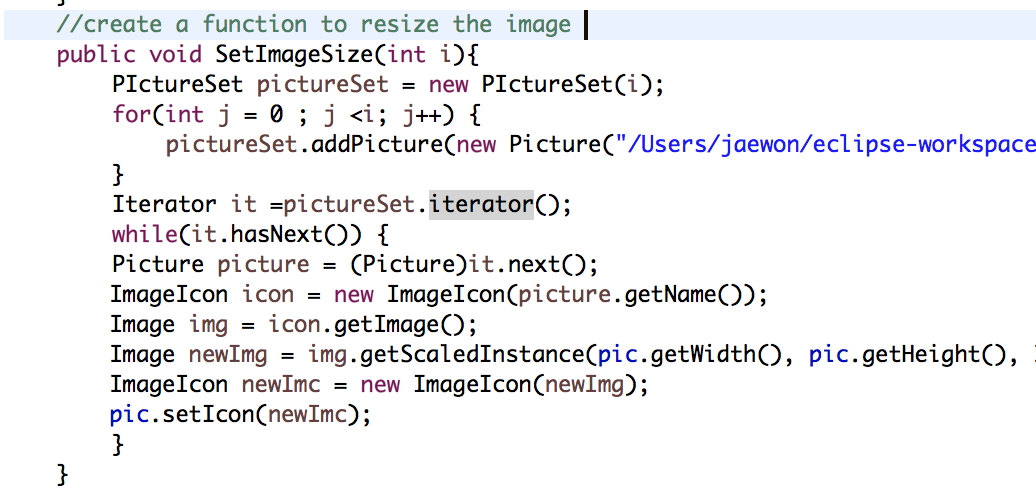


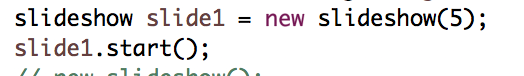
* Iterator Design Pattern

iterator패턴을 사용하면 컬렉션 구현 방법을 노출시키지 않고도 그 집합체에 있는 모든 항목에 접근할 수 있다. 인터페이스 및 구현이 간단해진다.

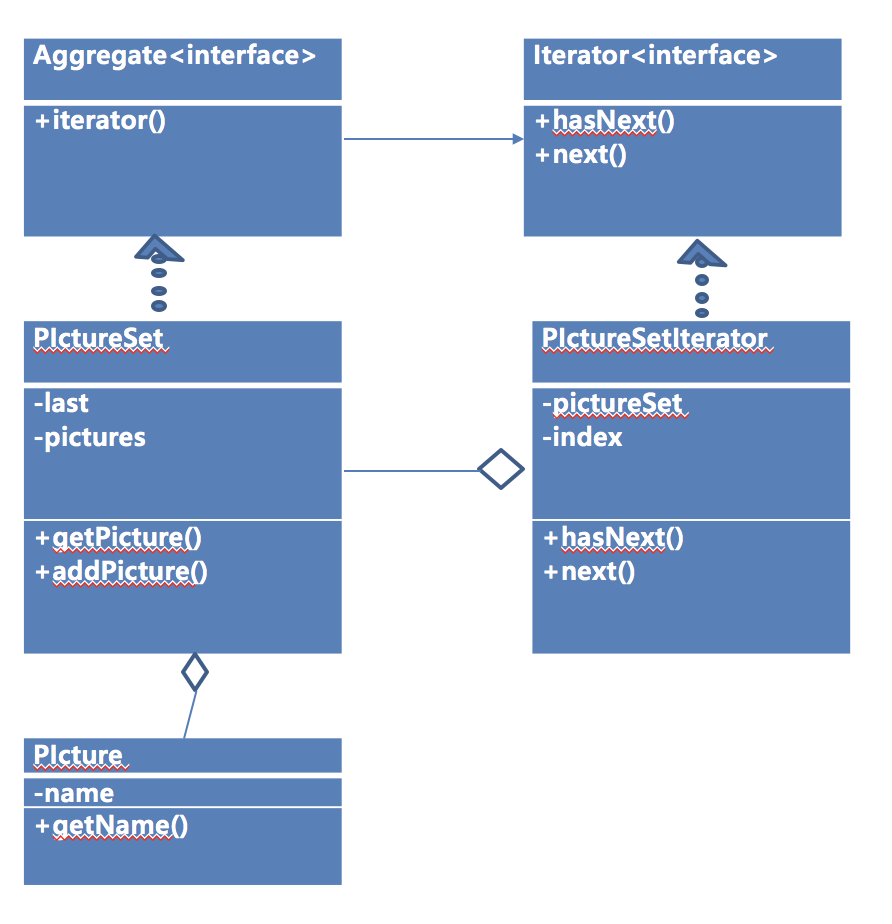


적용한 부분





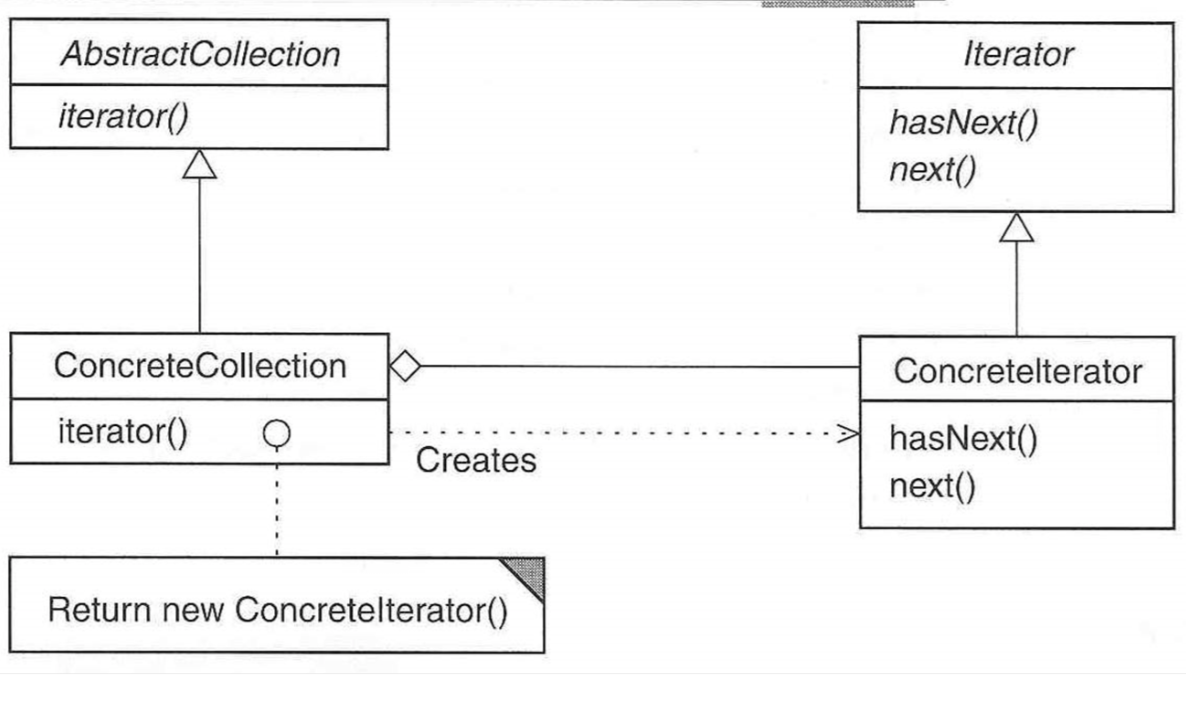
Class Diagram



1. 텍스트 포스팅

* Iterator Design Pattern

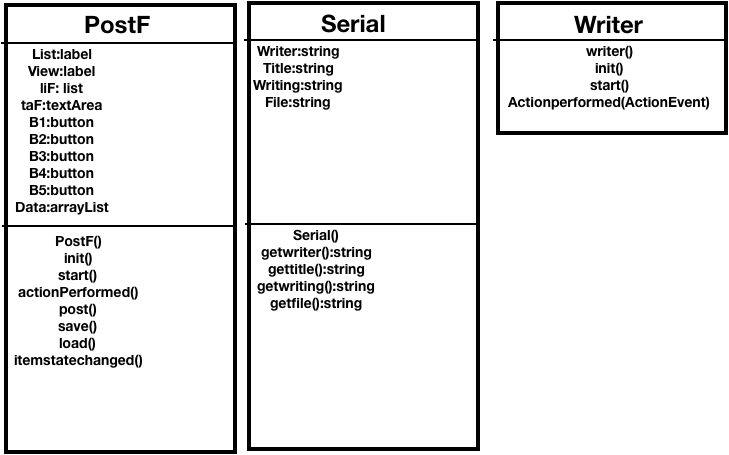
iterator패턴을 사용하면 컬렉션 구현 방법을 노출시키지 않고도 그 집합체에 있는 모든 항목에 접근할 수 있다. 인터페이스 및 구현이 간단해진다.



적용한 부분

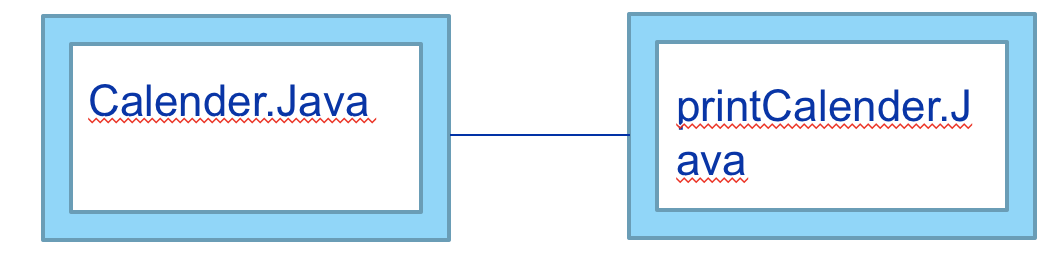


Class Diagram



1. 달력

Class Diagram



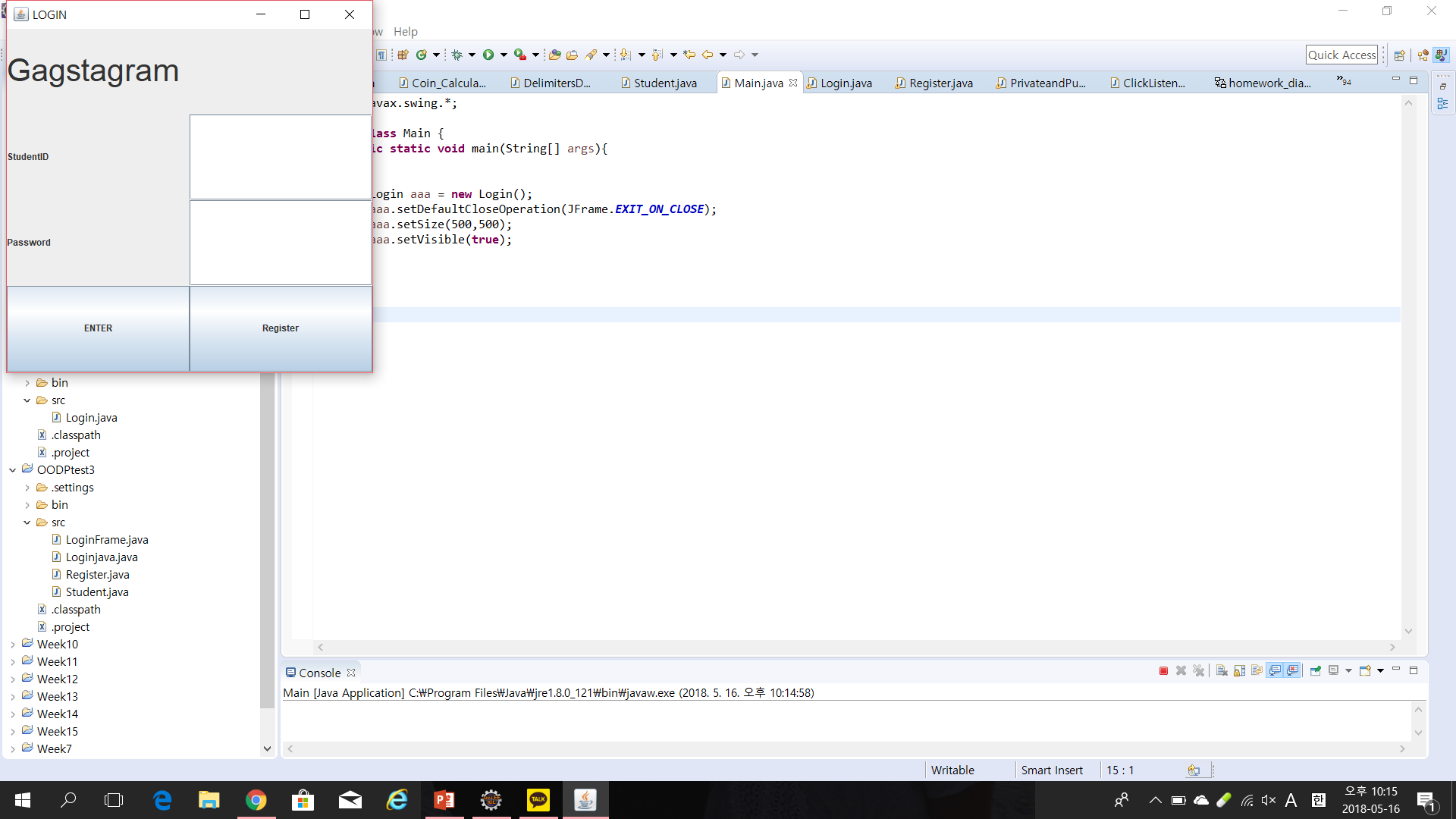
총 구현 기능

* 사진 저장 및 슬라이드쇼
* 텍스트 포스팅
* 회원가입 및 로그인
* 달력 출력 및 선택

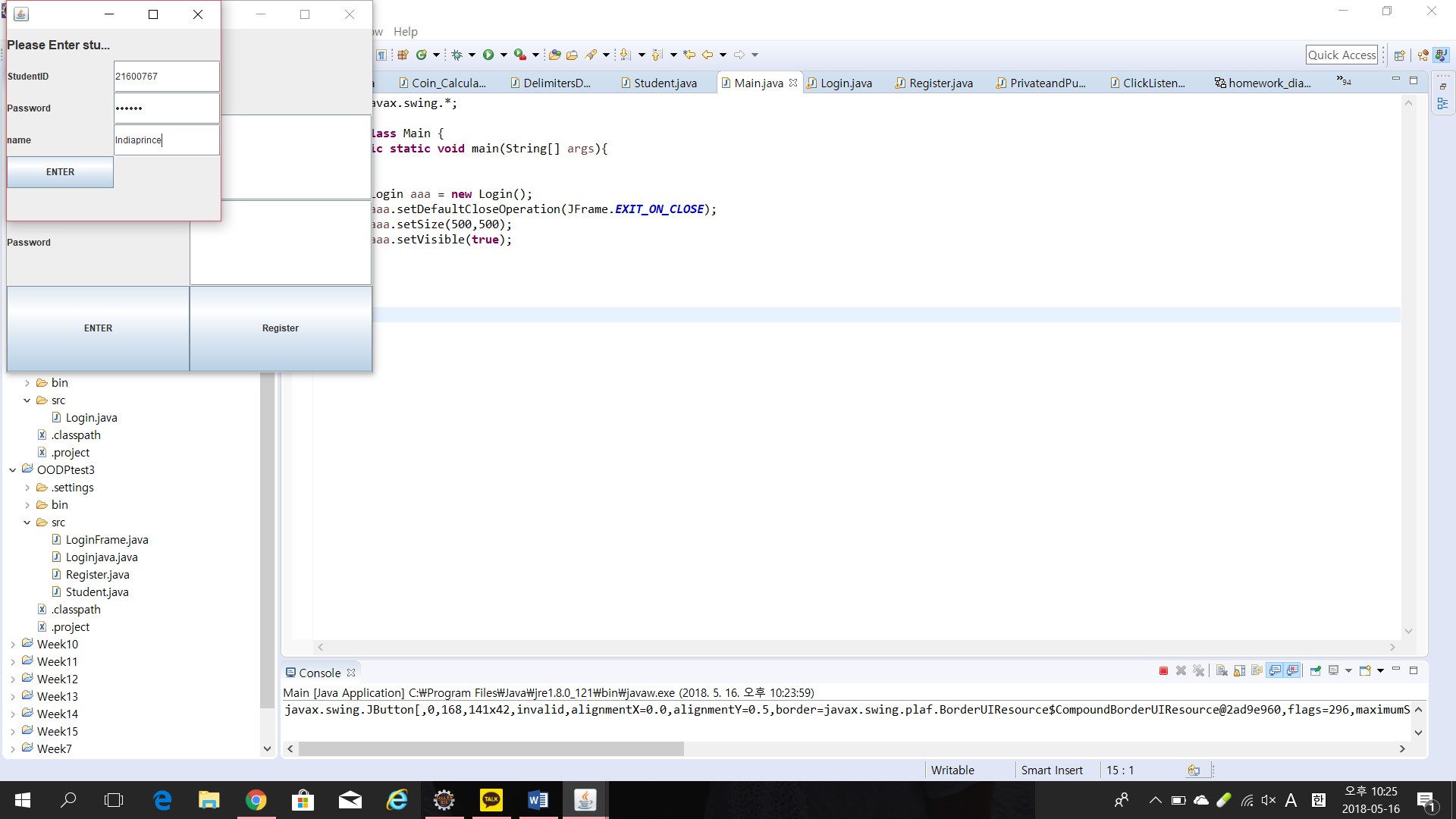
실행

* 로그인

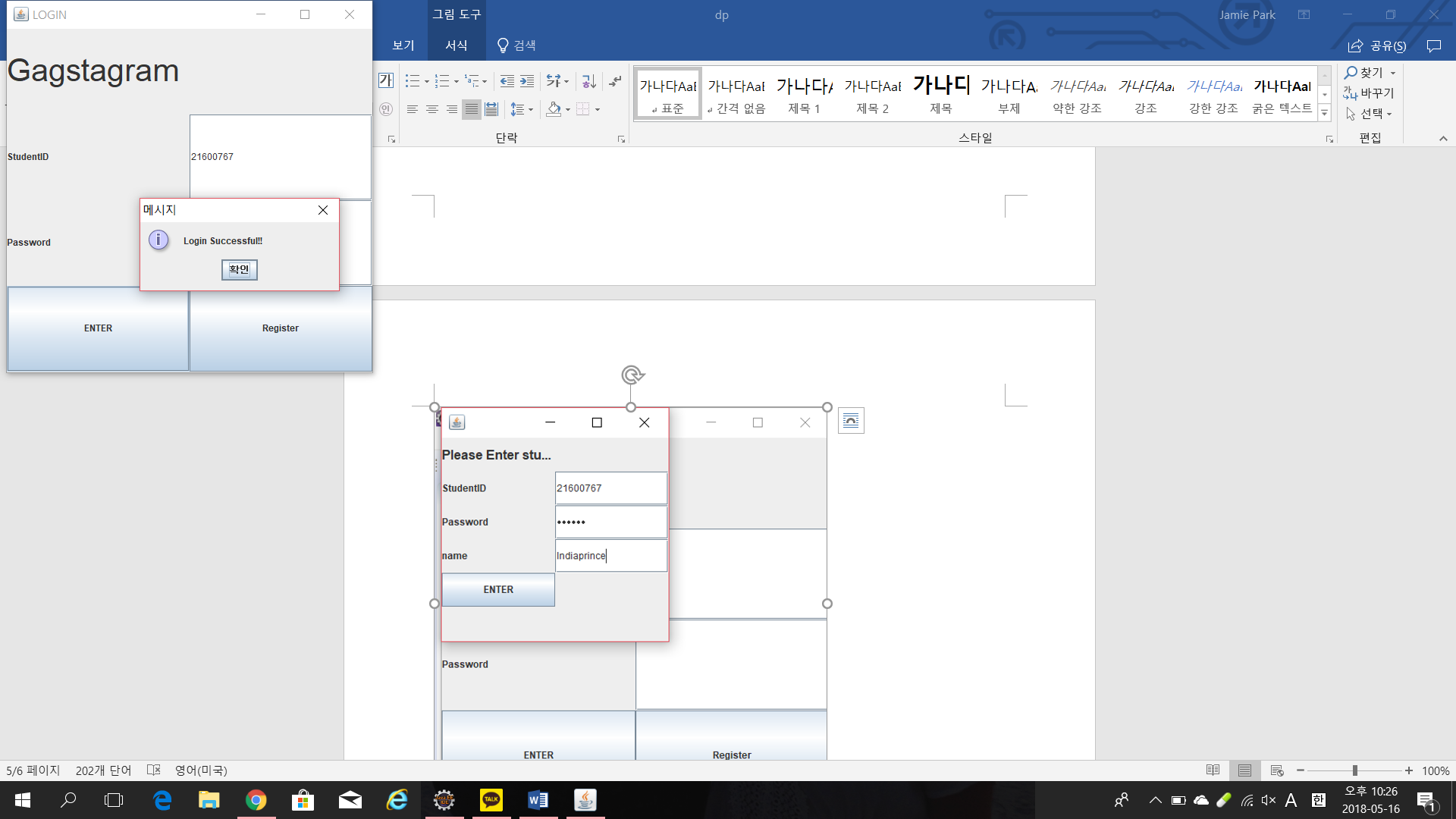
< 첫실행 화면>



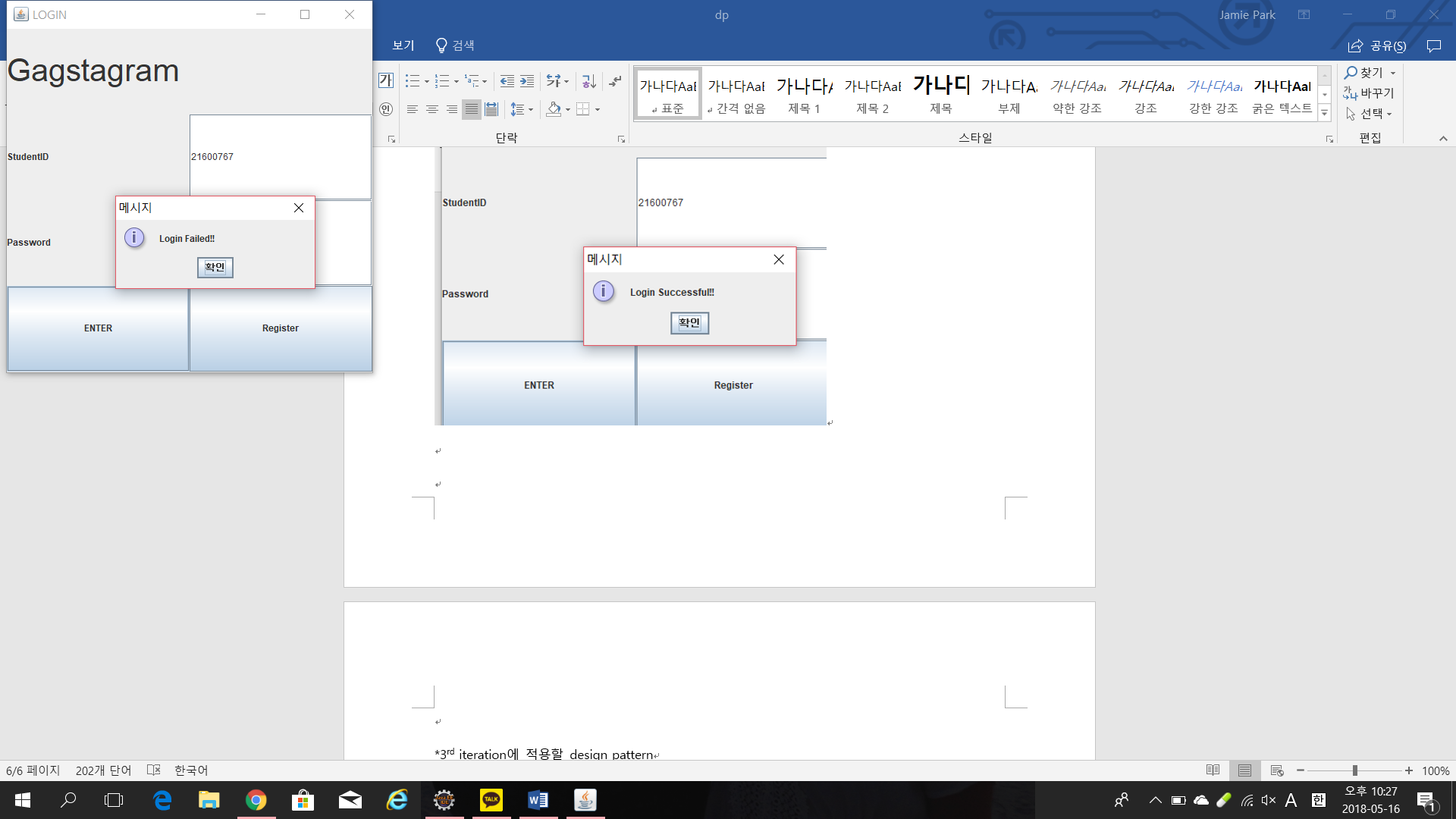
<회원가입 화면>



<로그인 성공>



<로그인 실패>

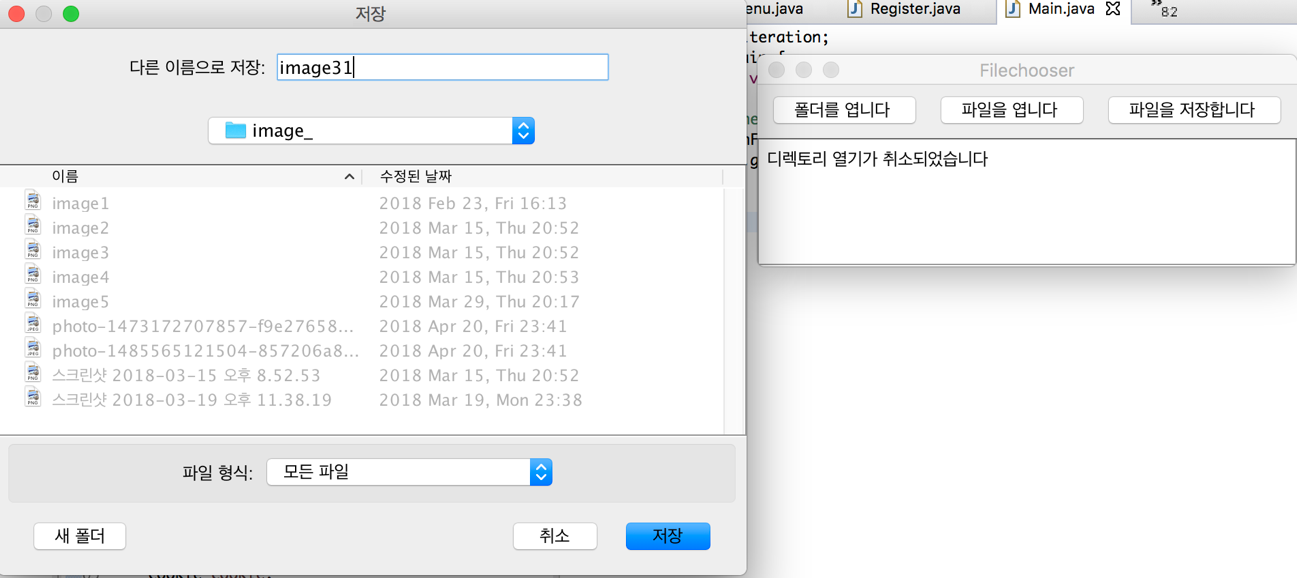


* 슬라이드 쇼

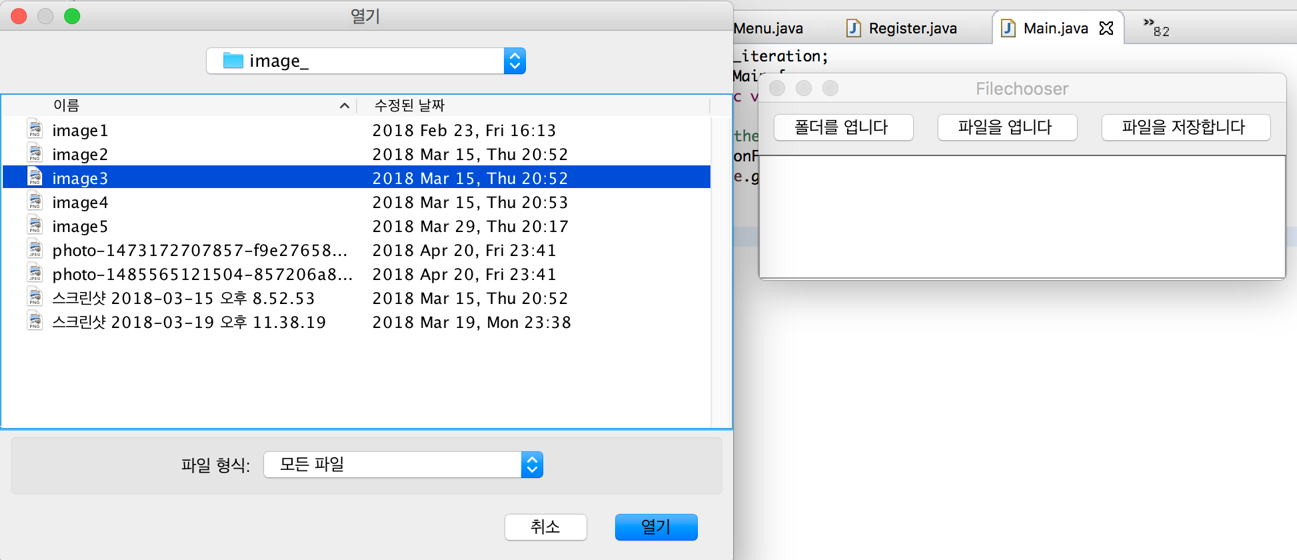
<첫실행 화면>



<’폴더를 엽니다’를 눌렀을 때>



<’파일을 엽니다’를 눌렀을때>



* 텍스트

<첫실행화면>

스크린샷이(가) 표시된 사진

매우 높은 신뢰도로 생성된 설명

<포스팅 창>

스크린샷이(가) 표시된 사진

매우 높은 신뢰도로 생성된 설명

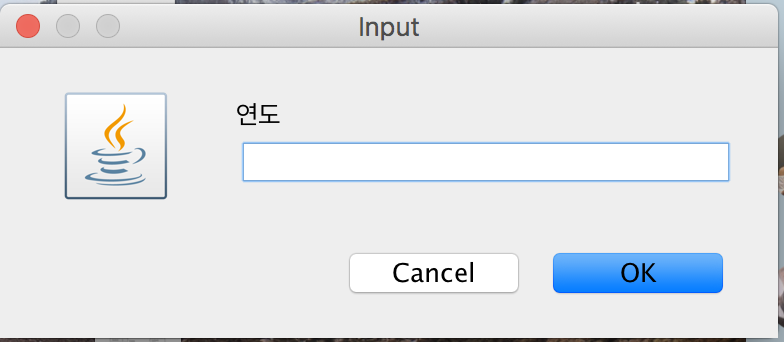
<포스팅 이후>

스크린샷이(가) 표시된 사진

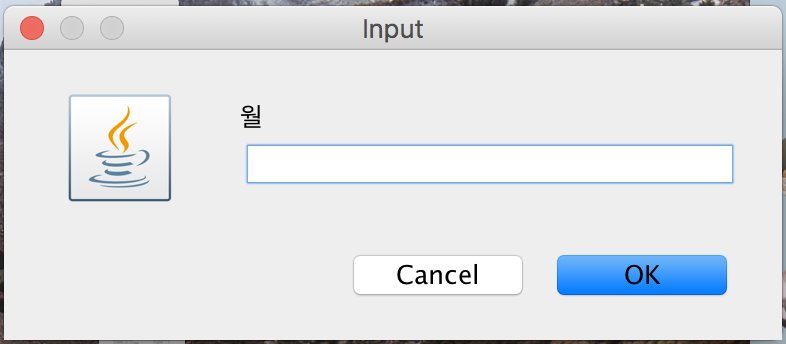
매우 높은 신뢰도로 생성된 설명

* 달력

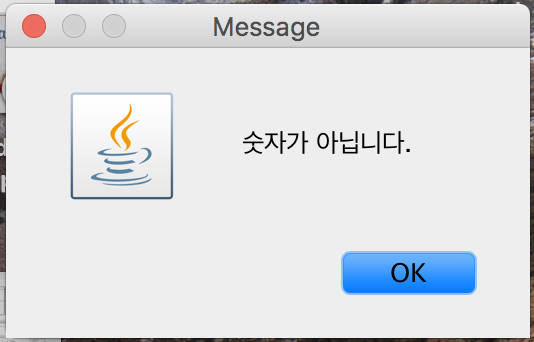
<첫실행 화면>



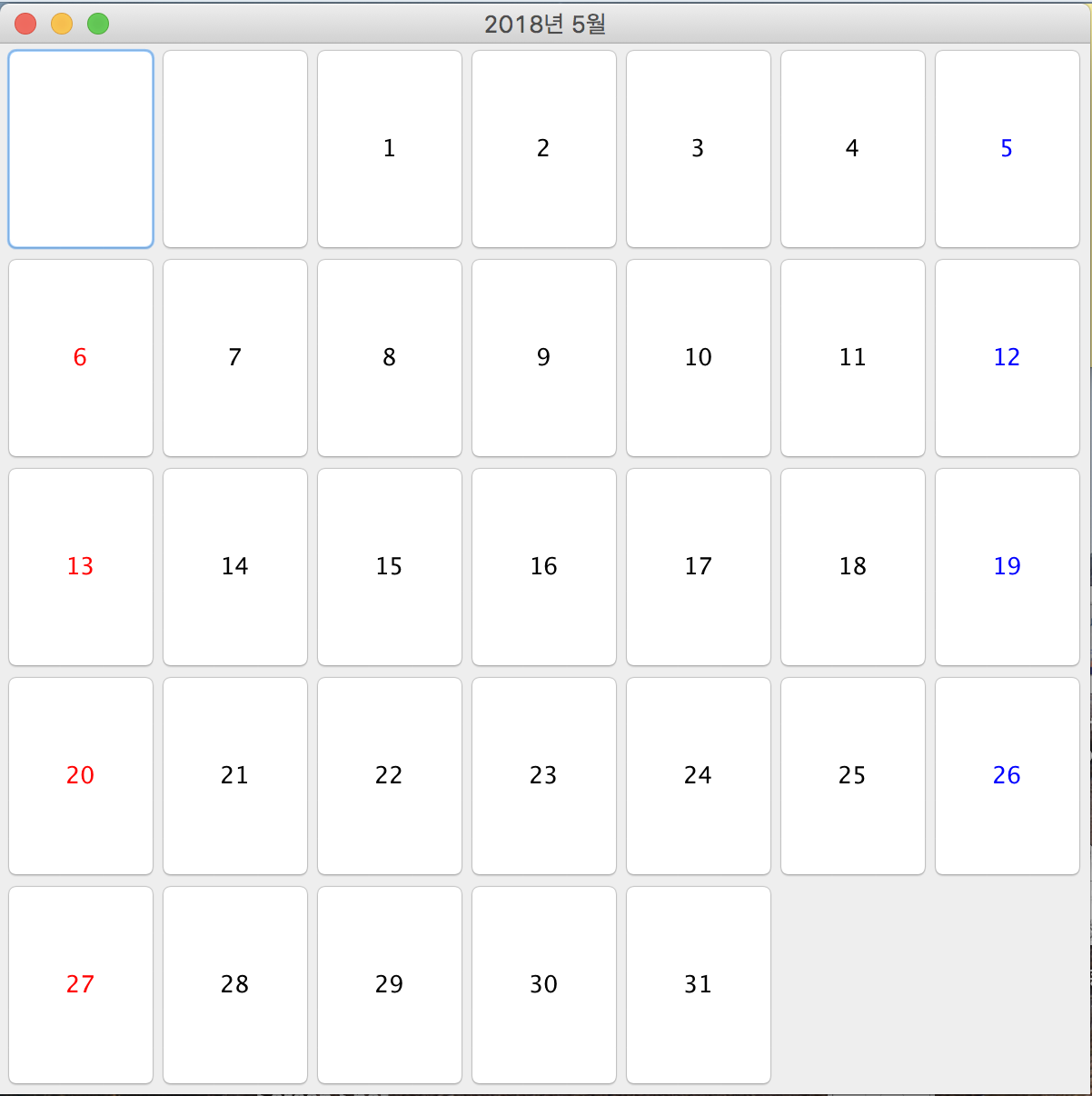
<연도 입력 후>



<숫자가 아닌 다른 수를 입력했을 때>



<정상적인 입력 후>



3rd iteration계획

* 로그인

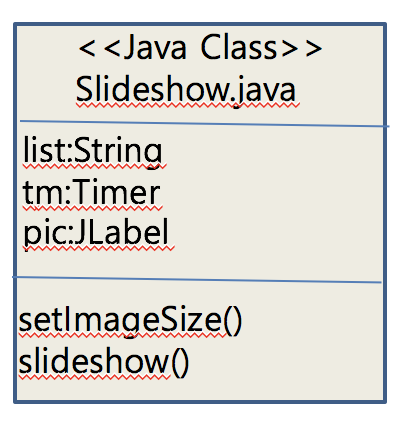
Command pattern

사용하려는 이유 : command pattern에서는 새로운 종류의 요청이 추가 될 때마다 command의 하위 클래스를 매번 생성해야 되기 때문에 template을 사용하는 것이 효율적이지만 template은 요청을 처리하는 객체의 인터페이스가 통일돼야 하기 때문에 command pattern 이 더 나은 방법일 수 있다.

* 슬라이드쇼

abstract factory pattern

슬라이드 쇼의 배경을 바꾸는 것을 abstract factory 디자인 패턴을 사용해서 구현할 예정이다.



* 텍스트 & 달력

텍스트와 달력을 연동하여 일정을 저장할 수 있도록 할 예정이다.