**2020학년도 정보보호시스템 7조 팀 프로젝트 설계서**

|  |
| --- |
| 벨 라파둘라 모델 |

(Bell-LaPadula Model)

|  |  |
| --- | --- |
| 7조 | |
| 20164269 | 이현호 |
| 20184387 | 정유진 |
| 20164310 | 정정현 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 목차 | | | | |
| 1 |  | 개요 |  | 3p |
|  | .1 | 프로젝트의 목표 |  | 4p |
|  | .2 | 개발환경 |  | 4p |
|  | .3 | 소프트웨어의 주요기능 |  | 4p |
|  | .3.1 | 관리자 기능 |  | 4p |
|  | .3.2 | 사용자 기능 |  | 4p |
|  | .4 | 설계상 제약사항 |  | 5p |
| 2 |  | 사용자 인터페이스 설계 |  | 5p |
| 3 |  | 프로그램 설계 |  | 7p |
|  | .1 | 클래스 설계 |  | 8p |
|  | .2 | 의사코드 |  | 9p |

**1. 개요**

|  |  |  |
| --- | --- | --- |
| 팀 | 7조 | |
| 프로젝트명 | 국문 | 벨 라파둘라 모델 |
| 영문 | Bell-LaPadula Model |
| 프로젝트 요약 | 본 프로젝트는 정보보호의 3요소 중 기밀성과 관련하여 정보의 상하 관계가 구분된 정보를 보호하기 위한 접근 통제 모델인 벨 라파둘라를 약식으로 구현함으로써 정보보안에 대한 이해도를 높이기 위해 시작되었다.  본 프로젝트는 작동하는 프로그램을 구현하는데 초점을 맞추고 있다. 따라서 소스코드에서 개선시킬 수 있는 사항들, 예를 들어 static 변수의 사용과 비효율적인 로직은 용인하도록 한다.  이외에도 약식으로 진행되는 프로젝트이지만, 실제 프로젝트 진행처럼 진행 중 발생하는 문제점과 의견들을 최대한 기록하여 진행한다. | |
| 프로젝트 기간 | 2020년 11월 12일 ~ 2020년 12월 3일(주제변경으로 인함) | |
| 프로젝트 인원 | 3명 | |

**1.1 프로젝트의 목표**

본 프로젝트는 다음과 같은 목표를 갖는다.

-벨 라파둘라 모델의 완벽한 이해

-실제 작동하는 프로그램 개발

**1.2 개발환경**

|  |  |  |
| --- | --- | --- |
| 구분 | 항목 | 사양 |
| 하드웨어 | CPU | i5-4210M |
| MEMORY | 8GB |
| SSD | 256GB |
| 소프트웨어 | OS | Windows 10 Pro |
| COMPILER | Eclipse 2020-09 (4.17.0) |

**1.3 소프트웨어의 주요기능**

1.3.1 관리자 기능

-사용자 목록 조회 : 회원가입한 사용자의 목록을 조회 가능하도록 한다.

-권한부여 : 회원가입한 사용자의 권한을 수정할 수 있도록 한다.

1.3.2 사용자 기능

-회원가입 - 게시판 사용을 위한 계정을 생성할 수 있도록 한다.

-로그인 - 게시판 작성을 위해 등록 되어있는 계정으로 로그인할 수 있도록 한다.

-사용자와 게시판의 레벨을 비교해 게시판의 글을 수정, 삭제, 작성할 수 있거나 없도록 한다.

**1.4 설계상 제약사항**

-이벤트 처리를 하기 위해 프로그래밍 언어로 JAVA를 사용한다.

-프로그램을 재실행해도 게시판의 내용과 회원정보를 유지하기 위해 텍스트파일을 입출력 한다.

-회원정보는 암호화되지 않는다.

**2. 사용자 인터페이스 설계**

|  |  |
| --- | --- |
| **[그림 1] 로그인 화면** | **[그림 2] 접속 초기화면** |
| **[그림 3] 게스트 계정 생성** | **[그림 4] 생성한 계정에 Admin 계정으로 등급 설정** |
| **[그림 5] 게시판 초기 모습** | **[그림 6] 읽기가 허용된 모습** |
| **[그림 7] 쓰기가 허용된 모습** | **[그림 8] no read up** |
| **[그림 9] no write down** |  |

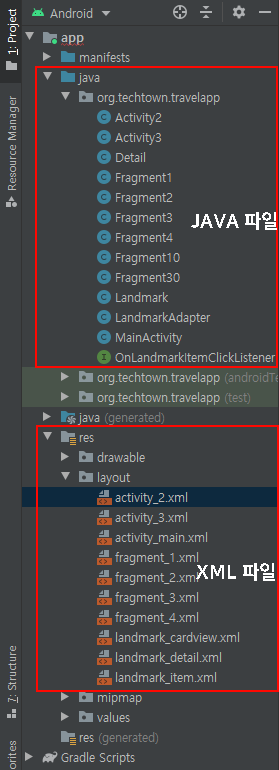
위 그림들은 분석단계에서 설계한 그림이다.

-[그림 5]는 [그림2]와 유사하기 때문에 별도로 구현하지 않는다.

-[그림 6]과 [그림 7]는 유사하다고 판단해 두 개의 화면을 따로 만들지 않고 하나의 화면에 [수정/삭제] 버튼의 활성화 여부를 다르게 하도록 한다.

-[그림 8]과 [그림 9] 또한 하나의 화면에서 Dialog의 내용을 다르게 출력하여 구현한다.

**3. 프로그램 설계**



안드로이드 앱에는 화면구성을 담당하는 XML파일과 프로그래밍 구현부분인 JAVA파일이 있다. 이 구성을 인용하여 화면을 구성하는 JAVA파일의 이름은 데이터처리를 하는 JAVA파일의 이름에 ACTIVITY를 붙여 만들 계획이다. 이렇게 하면 각 JAVA파일들의 라인 수를 줄여 가독성을 높일 수 있고, 캡슐화를 통해 정보은닉을 실현할 수 있다.

**3.1 클래스 설계**

[화면구성]

(1) AuthorizationActivity : 관리자가 회원들의 권한을 수정하는 화면

(2)LoginActivity : 사용자 또는 관리자가 로그인하는 화면

(3) RegistrationActivity : 사용자가 가입을 하는 화면

(4) MainActivity : 로그인 직후 게시판에 글을 쓰거나 읽기 위해 파일을 선택하는 화면

(5) ReadingActivity : 게시판의 글을 읽는 화면

(6) WritingActivity : 게시판에 글을 쓰는 화면

[데이터 처리]

(1) Login : 사용자가 입력한 ID와 PW를 텍스트파일과 대조하여 회원인지 판별

(2) Registration : 아이디가 중복인지 검사하고 중복이 아니라면 가입 허용

(3) Reading : 텍스트파일을 읽어들여 JTextArea에 출력

(4) Writing : JTextArea의 내용을 텍스트파일에 저장

[기타]

(1) MyJPanel : 만들어진 화면을 관리한다.

(2) Test : 메인함수가 들어있는 클래스

**3.2 의사코드**

[화면구성]

(1) AuthorizationActivity.java

IDComboBox, LevelComboBox;

IDComboBox <- SELECT ID FROM MemberInfo.txt

procedure authorize

get selected Item from IDComboBox, LevelComboBox;

while !EOF

read MemberInfo.txt

delete line which contains selected Item;

append new line with modified authority;

print “적용 완료”

end

(2) LoginActivity

IDTextField, PWPasswordField, LoginButton, RegisterButton;

String ID, PW;

procedure login

if clicked item == LoginJButton then

ID <- IDTextField, PW <- PWPasswordField;

Login.Login(ID,PW)

if clicked item == RegisterButton then

switch to RegistrationActivity screen

end

(3) RegistrationActivity

IDTextField, PWPasswordField, RegistrationButton, redundancyCheckButton;

if clicked item == RegistrationButton then

if redundancyCheck true

append new memberinfo

else then

print “아이디 중복”

if clicked item == redundancyCheckButton

read MemberInfo.txt

check if IDTextField equals then

return boolean;

(4) MainActivity

BulletinBoardNameComboBox, ActionComboBox;

BulletinBoardNameComboBox <- “Level 1”, “Level 2”, “Level 3”, …;

ActionComboBox <- “작성”, “읽기”;

if BulletinBoardNameComboBox state changed and ActionComboBox state changed then

Bulletinlevel <- BulletinBoardNameComboBox

case “작성”

if authority > Bulletinlevel

switch to WritingActivity screen

else

print “쓰기가 불가능한 계정”

case “읽기”

if authority < Bulletinlevel

switch to ReadingActivity

else

print “읽기가 불가능한 계정”

(5) ReadingActivity

show BulletinBoradTextArea;

(6) WritingActivity

FilenameComboBox, TextArea, SaveButton;

show EmptyTextArea;

if clicked item == SaveButton then

call Writing method;

[데이터 처리]

(1) Login

get ID, PW from LoginActivity;

read a line from MemberInfo.txt;

while !EOF

if line contains ID then

check if ID, PW equals;

return Boolean;

end

(2) Registration

procedure register

append ID, PW, authority to MemberInfo.txt

end

procedure redundancyCheck

read a line from MemberInfo.txt;

while !EOF

if line contains ID then

check if ID equals;

return Boolean;

end

(3) Reading

Read selected text file and append it to BulletinBoardTextArea;

(4) Writing

get filename from WritingActivity

if filename conflict then

print “파일이름 중복”;

else then

set file name to FilenameComboBox;

save text file;

[기타]

(1) MyJPanel

declare every Activity;

initialize every Activity instane to null;

Stack ActivityStack;

procedure change

removeall component;

switch panelName

case “LoginActivity” : show LoginActivity

case “RegistrationActivity” : show RegistrationActivity

case “AuthorizationActivity” : show AuthorizationActivity

case “WritingActivity” : show WritingActivity

case “ReadingActivity” : show ReadingActivity

case “MainActivity” : show MainActivity

end

procedure goback

pop an item from ActivityStack then do MyPanel.change

(2) Test

dynamically allocate a MyJPanel variable;

set panelname to “Bell-LaPadula model”;

show LoginActivity;