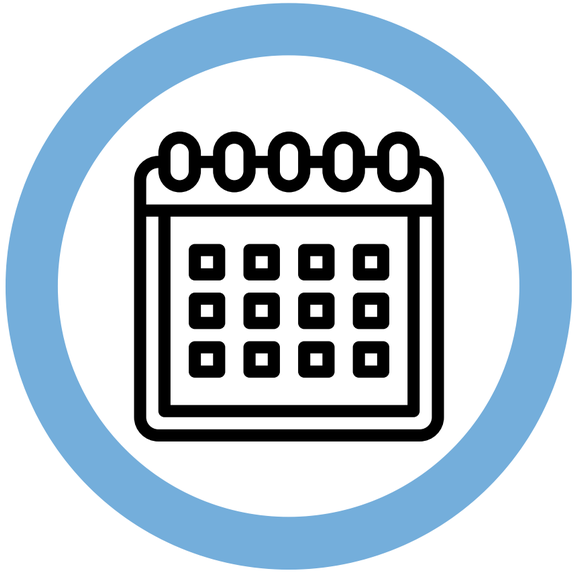
JAVA PROJECT. 241014 ~ 241024
TEAM : 박진우

원격으로 빠른 진료 예약을 할 수 있는 병원 예약관리 시스템

목차

1. 프로젝트 소개

1-1. 개요 (1/2)



팀	기간
(1인) 박진우	2024.10.14 ~ 2024.10.24

Java Project GitHub Link

https://github.com/J1NU2/Hospital_Reservation_Management

1. 프로젝트 소개

1-1. 개요 (2/2)

Why?

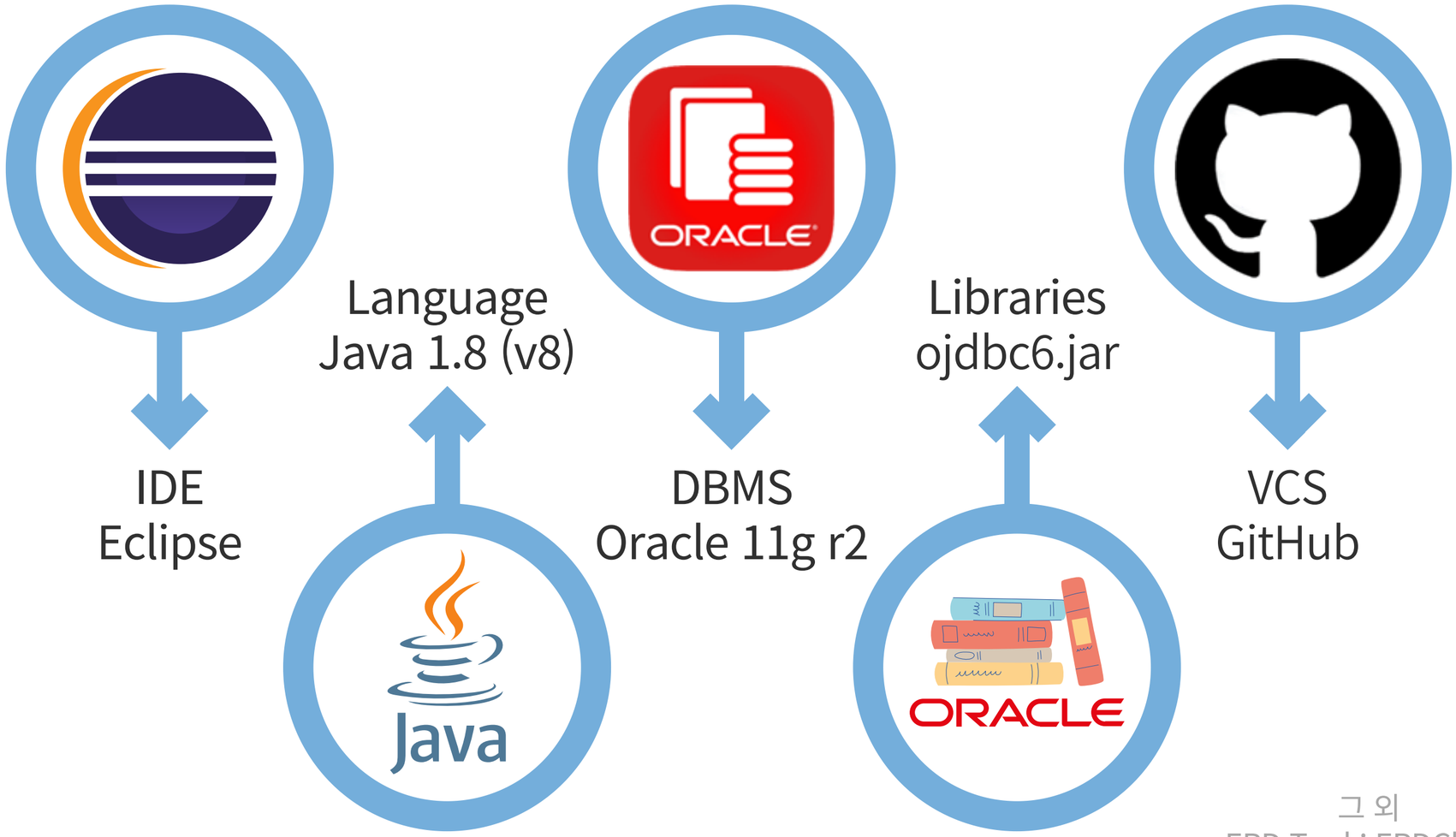
직접 병원에 갈 시간이 부족하거나, 전화로 진료 예약을 하기 힘든 사람들을 위한 프로그램을 만들어보자.

프로젝트 소개

- 사용자가 원하는 시간대에 미리 병원 진료를 예약하고, 관계자가 예약 목록을 확인할 수 있는 프로그램
- Java AWT 및 Swing을 이용한 GUI 구성
- Java DTO/DAO를 이용한 클래스 설계
- Oracle DBMS를 이용한 데이터 관리

1. 프로젝트 소개

1-2. 개발 환경



그 외
ERD Tool : ERDCloud
DBMS Tool : SQL Plus

1. 프로젝트 소개

1-3. 개발 일정

병원 예약관리 시스템												
Task	Completion (완/미완/진행중)	Schedule										
		10/14	10/15	10/16	10/17	10/18	10/19	10/20	10/21	10/22	10/23	10/24
		월	화	수	목	금	토	일	월	화	수	목
0. WBS 작성	완											
1. 요구사항 분석												
1-1. 시나리오 작성	완											
1-2. 유스케이스 다이어그램 작성	완											
1-3. Mockup 작성	완											
2. 기능별 설계												
2-1. DB 설계												
2-1-1. ERD 작성	완											
2-1-2. 테이블 명세서 작성	완											
2-1-3. 테이블 설계	완											
2-1-4. 클래스 명세서 작성	완											
2-1-5. 클래스 설계	완											
2-2. DB 기능 구현												
2-2-1. DAO → DB 데이터 전달 구성	완											
2-3. Java 기능 설계												
2-3-1. GUI 구성	완											
2-3-2. 코드 구현	완											
3. 테스트												
3-1. 초기 데이터값 INSERT	완											
3-2. 테스트	완											
3-3. 버그 수정 및 마무리	완											
4. PPT 작성	-											
5. 발표	-											

2. 프로젝트 분석

2-1. 요구사항 정의 (1/9)

1. 회원가입 기능 (1/3)

요구사항 명		회원가입 기능
요구사항	정의	회원제 서비스를 제공받기 위한 회원 등록 정의
	상세 설명	일반회원(환자) 또는 관계자(의사)의 정보를 입력받아 회원 등록을 해야 한다. - 일반회원(환자) 정보 : 아이디, 비밀번호, 이름, 주민등록번호, 나이, 성별, 휴대전화번호, 가입날짜 - 가입날짜는 입력하는 것이 아닌 등록 날짜를 기준으로 저장된다. - 관계자(의사) 정보 : 의사번호, 비밀번호, 이름, 전공 - 전공은 입력하는 것이 아닌 정해진 리스트 내에서 선택한다. - 전공 : 내과, 일반외과, 정형외과, 소아과, 안과, 이비인후과, 피부과, 비뇨기과, 정신과, 성형외과
분류		기능
기술스택		Java Swing/AWT, DAO, DTO, DBMS(Oracle)

요구사항 명		회원가입 중복검사 기능
요구사항	정의	회원 정보 등록 시 검사를 통한 중복방지 정의
	상세 설명	회원 등록 시 아이디 중복 생성을 방지하기 위해 일반회원(환자)는 아이디, 주민번호를 관계자(의사)는 의사번호를 기준으로 중복방지 검사를 진행한다. - 일반회원(환자) : 아이디, 주민번호 입력값을 가지고 중복여부를 판단한다. - 관계자(의사) : 의사번호 입력값을 가지고 중복여부를 판단한다. - 해당 정보가 중복되면 true, 중복되지 않으면 false
분류		기능
기술스택		Java Swing/AWT, DAO, DBMS(Oracle)

2. 프로젝트 분석

2-1. 요구사항 정의 (2/9)

1. 회원가입 기능 (2/3)

요구사항 명		비밀번호 재확인 기능
요구사항	정의	비밀번호 등록 시 비밀번호 재확인을 통한 일치 검사 정의
	상세 설명	회원 등록 시 오타 및 잘못 입력되는 상황을 방지하기 비밀번호 확인을 진행한다. - 비밀번호와 비밀번호 확인 입력값을 비교하여 일치하도록 한다. - 일치하면 true, 일치하지 않으면 false
분류		기능
기술스택		Java Swing/AWT, Method

요구사항 명		회원가입 정보 미입력 확인 기능
요구사항	정의	회원 등록 시 정보 미입력에 따른 처리 정의
	상세 설명	회원 등록 시 모든 정보는 필수로 입력한다. - 입력된 정보가 하나라도 빠져있다면 false, 그렇지 않다면 true
분류		기능
기술스택		Java Swing/AWT, Method

요구사항 명		정보 입력 제한 기능
요구사항	정의	회원 등록 시 정보 입력 유형 정의
	상세 설명	회원 등록 시 입력값은 입력 유형에 포함되어야 입력이 가능하다. - 아이디 : 영어 대소문자, 숫자, 8글자 제한 - 비밀번호 : 영어 대소문자, 숫자, 20글자 제한 - 이름 : 영어 대소문자, 한글, 8글자 제한 - 주민번호 : 숫자, 앞자리6글자 / 뒷자리7글자 제한 - 전화번호 : 숫자, 첫번째3글자 / 두번째3-4글자 / 세번째4글자 제한 - 의사번호 : 영어 대문자, 숫자, 6글자 제한
분류		기능
기술스택		Java Swing/AWT, Method

2. 프로젝트 분석

2-1. 요구사항 정의 (3/9)

1. 회원가입 기능 (3/3)

요구사항 명		나이, 성별 자동 입력 기능
요구사항	정의	주민번호 입력 시 나이, 성별 자동 입력 정의
	상세 설명	회원 등록 시 주민번호를 입력하면 나이, 성별이 자동으로 입력된다. - 나이는 주민번호 앞자리와 현재 날짜를 비교하여 만나이로 저장한다. - 성별은 주민번호 뒷자리의 첫번째 글자를 기준으로 홀수면 "M", 짝수면 "F"를 저장한다.
분류		기능
기술스택		Java Swing/AWT, Method

2. 프로젝트 분석

2-1. 요구사항 정의 (4/9)

2. 로그인 기능

요구사항 명		로그인 기능
요구사항	정의	회원제 서비스를 이용하기 위한 회원 로그인 정의
	상세 설명	일반회원(환자) 또는 관계자(의사)의 정보를 입력받아 로그인을 진행한다. - 입력값은 아이디와 비밀번호를 입력한다. - 입력값이 회원 등록이 된 회원의 정보와 일치하는지 판단하여 로그인 여부를 결정한다.
분류		기능
기술스택		Java Swing/AWT, DAO, DBMS(Oracle)

요구사항 명		관계자 인증 절차
요구사항	정의	관계자 로그인 서비스를 이용하기 위한 관계자 인증 절차 정의
	상세 설명	미리 정의된 인증번호를 입력받아 관계자 로그인 UI로 넘어간다. - 인증번호가 맞다면 로그인 UI로 넘어가고, 틀리면 인증번호 재확인을 요청받는다.
분류		인터페이스
기술스택		Java Swing/AWT

3. 회원가입/로그인 공통

요구사항 명		비밀번호 표시 은닉
요구사항	정의	비밀번호 입력 정의
	상세 설명	비밀번호 입력 시 입력한 내용이 보여지지 않는다. - 입력값은 바뀌지 않고, 표시만 ' · ' 으로 표시된다.
분류		인터페이스
기술스택		Java Swing/AWT

2. 프로젝트 분석

2-1. 요구사항 정의 (5/9)

4. 진료 예약 (1/4)

요구사항 명		예약하기 기능
요구사항	정의	회원의 진료 예약 서비스 이용 정의
	상세 설명	일반회원(환자)는 담당 의사, 예약날짜/시간 정보를 선택하여 진료 예약을 한다. - 예약 시 다음 정보들을 저장한다. : 일반회원(환자) : 이름, 주민번호 : 선택한 담당 의사 : 이름, 의사번호, 전공 : 선택한 예약날짜(YYYY-MM-DD) : 선택한 예약시간(HH:MM)
분류		기능
기술스택		Java Swing/AWT, DAO, DTO, DBMS(Oracle)

요구사항 명		담당 의사 정보 표시 기능
요구사항	정의	담당 의사 선택 시 정보 표시 정의
	상세 설명	진료 예약 시 담당 의사를 선택하면 선택한 의사의 정보를 표시한다. - 선택된 담당 의사의 정보(의사번호, 전공)가 입력된다.
분류		기능
기술스택		Java Swing/AWT, DAO, DBMS(Oracle)

요구사항 명		예약날짜 리스트 재배치 기능
요구사항	정의	월 선택 시 일 선택 리스트 재배치 정의
	상세 설명	사용자가 예약날짜 선택 시 월을 선택하면 해당 월에 맞게 일이 재배치된다. - 선택된 월에 따라 그에 맞는 일이 재배치되어 리스트에 보여지도록 한다.
분류		기능
기술스택		Java Swing/AWT, Method

2. 프로젝트 분석

2-1. 요구사항 정의 (6/9)

4. 진료 예약 (2/4)

요구사항 명		진료 예약 정보 미선택 확인
요구사항	정의	진료 예약 시 정보 미선택에 따른 처리 정의
	상세 설명	사용자는 진료 예약 시 모든 정보는 필수로 선택한다. - 선택된 정보가 하나라도 빠져있다면 false, 그렇지 않다면 true
분류		기능
기술스택		Java Swing/AWT, Method

요구사항 명		진료 예약 제한 기능
요구사항	정의	진료 예약 시 검사를 통한 예약 제한 정의
	상세 설명	진료 예약 시 무분별한 예약으로 인한 메모리 낭비와 예약날짜 일치를 방지하기 위해 담당 의사의 정보와 예약날짜/시간을 기준으로 제한 검사를 진행한다. - 현재 로그인한 일반회원(환자)이 예약한 내역이 있다면 진료 완료가 이뤄지기 전까지 예약이 불가능하도록 한다. - 다른 회원이 한명의 의사에게 예약한 날짜/시간이 있을 때, 해당 담당 의사, 예약날짜/시간이 겹치지 않도록 예약을 제한한다. - 해당 예약 정보가 있다면 true, 없다면 false
분류		기능
기술스택		Java Swing/AWT, DAO, DBMS(Oracle)

2. 프로젝트 분석

2-1. 요구사항 정의 (7/9)

4. 진료 예약 (3/4)

요구사항 명		진료 예약 내역 확인 기능
요구사항	정의	진료 예약 시 검사를 통한 예약 제한 정의
	상세 설명	진료 예약 후 일반회원(환자)과 관계자(의사)는 진료 예약 내역을 확인할 수 있다. - 로그인한 일반회원(환자)은 자신이 예약한 현재 예약 내역을 확인할 수 있다. - 로그인한 일반회원(환자)은 자신이 예약했던 모든 예약 내역을 확인할 수 있다. - 로그인한 관계자(의사)는 자신에게 예약한 모든 일반회원(환자)의 예약 내역을 확인할 수 있다. - 예약 내역은 예약 정보를 포함한 테이블 형태로 표시된다.
분류		기능
기술스택		Java Swing/AWT, DAO, DBMS(Oracle)

2. 프로젝트 분석

2-1. 요구사항 정의 (8/9)

4. 진료 예약 (4/4)

요구사항 명		진료 예약 취소 기능
요구사항	정의	예약된 진료 내역 취소 정의
	상세 설명	진료 예약 후 일반회원(환자)과 관계자(의사)는 예약된 진료 내역을 취소할 수 있다. - 로그인한 일반회원(환자)은 자신이 예약한 현재 예약 내역을 취소할 수 있다. - 로그인한 관계자(의사)는 자신에게 예약한 일반회원(환자)의 예약 내역을 취소할 수 있다. - 관계자(의사)는 진료 예약 취소 시 취소 사유를 입력해야 한다.
분류		기능
기술스택		Java Swing/AWT, DAO, DBMS(Oracle)

요구사항 명		진료 확인 기능
요구사항	정의	진료 확인에 대한 정의
	상세 설명	관계자(의사)는 일반회원(환자)의 진료를 마친 뒤 진료 내역을 선택하여 증상메모를 수정한다. - 관계자(의사)는 진료를 마치면 진료를 마친 일반회원(환자)의 예약 내역을 선택하여 진료 여부를 변경한다. - 진료 여부 변경과 동시에 증상 메모를 입력한다.
분류		기능
기술스택		Java Swing/AWT, DAO, DBMS(Oracle)

요구사항 명		진료 내역 미선택 확인
요구사항	정의	진료 내역 정보 미선택에 따른 처리 정의
	상세 설명	관계자(의사)는 진료 후 처리 시 처리하고자 하는 행을 필수로 선택한다. - 선택된 행이 없으면 false, 있으면 true
분류		기능
기술스택		Java Swing/AWT, Method

2. 프로젝트 분석

2-1. 요구사항 정의 (9/9)

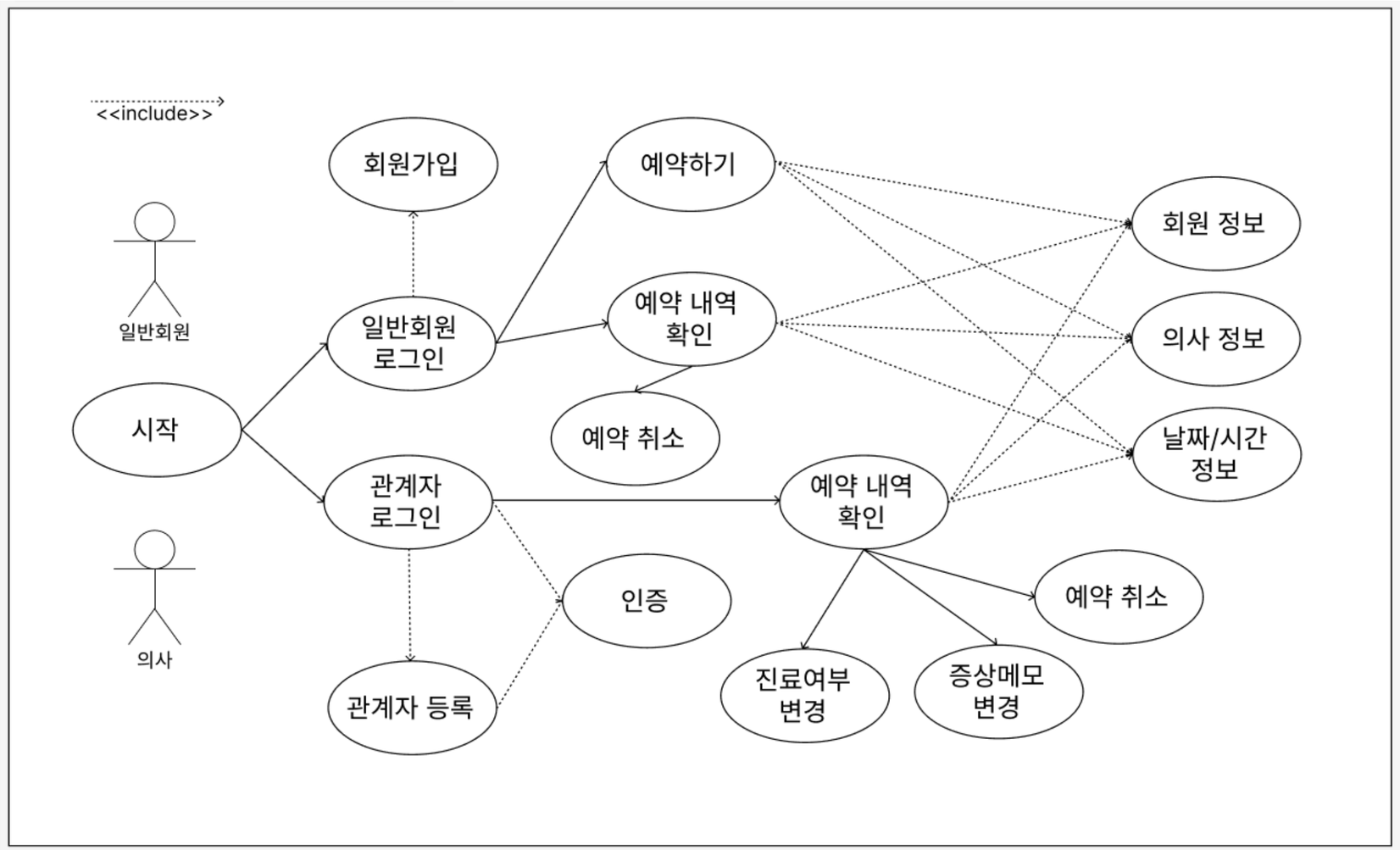
5. 인터페이스(UI)

요구사항 명		일반회원(환자) 인터페이스 구성
요구사항	정의	일반회원(환자) 인터페이스 구성 정의
	상세 설명	- 사용자는 회원가입 및 로그인 UI를 보고 이용할 수 있어야 한다. - 회원가입 정보 입력 시 미입력 사항에 대한 오류 메시지를 확인할 수 있어야 한다. - 회원가입 기능 이용에 있어 아이디, 주민번호 중복 검사 시 경고 텍스트를 확인할 수 있어야 한다. - 사용자는 로그인 후 진료 예약을 진행할 수 있어야 한다. - 사용자는 자신의 진료 예약 목록을 확인하고 취소할 수 있어야 한다. - 사용자가 현재 예약한 진료 내역을 취소할 때, 현재 진료 내역이 없다면 취소할 수 없다는 경고 텍스트를 확인할 수 있어야 한다.
분류		인터페이스
기술스택		Java Swing/AWT

요구사항 명		관계자(의사) 인터페이스 구성
요구사항	정의	관계자(의사) 인터페이스 구성 정의
	상세 설명	- 관계자는 회원가입 및 로그인 UI를 보고 이용할 수 있어야 한다. - 회원가입 정보 입력 시 미입력 사항에 대한 오류 메시지를 확인할 수 있어야 한다. - 회원가입 기능 이용에 있어 의사번호 중복 검사 시 경고 텍스트를 확인할 수 있어야 한다. - 관계자는 자신에게 예약된 모든 진료 예약 목록을 확인할 수 있어야 한다. - 관계자는 진료 예약 목록을 확인하고 선택할 수 있어야 한다. - 관계자는 선택된 진료 예약 목록에 대해 진료여부/증상메모를 변경하고 취소할 수 있어야 한다. - 선택된 진료 예약 목록이 없을 때, 진료여부/증상메모 변경이나 취소를 진행한다면 경고 텍스트를 확인할 수 있어야 한다.
분류		인터페이스
기술스택		Java Swing/AWT

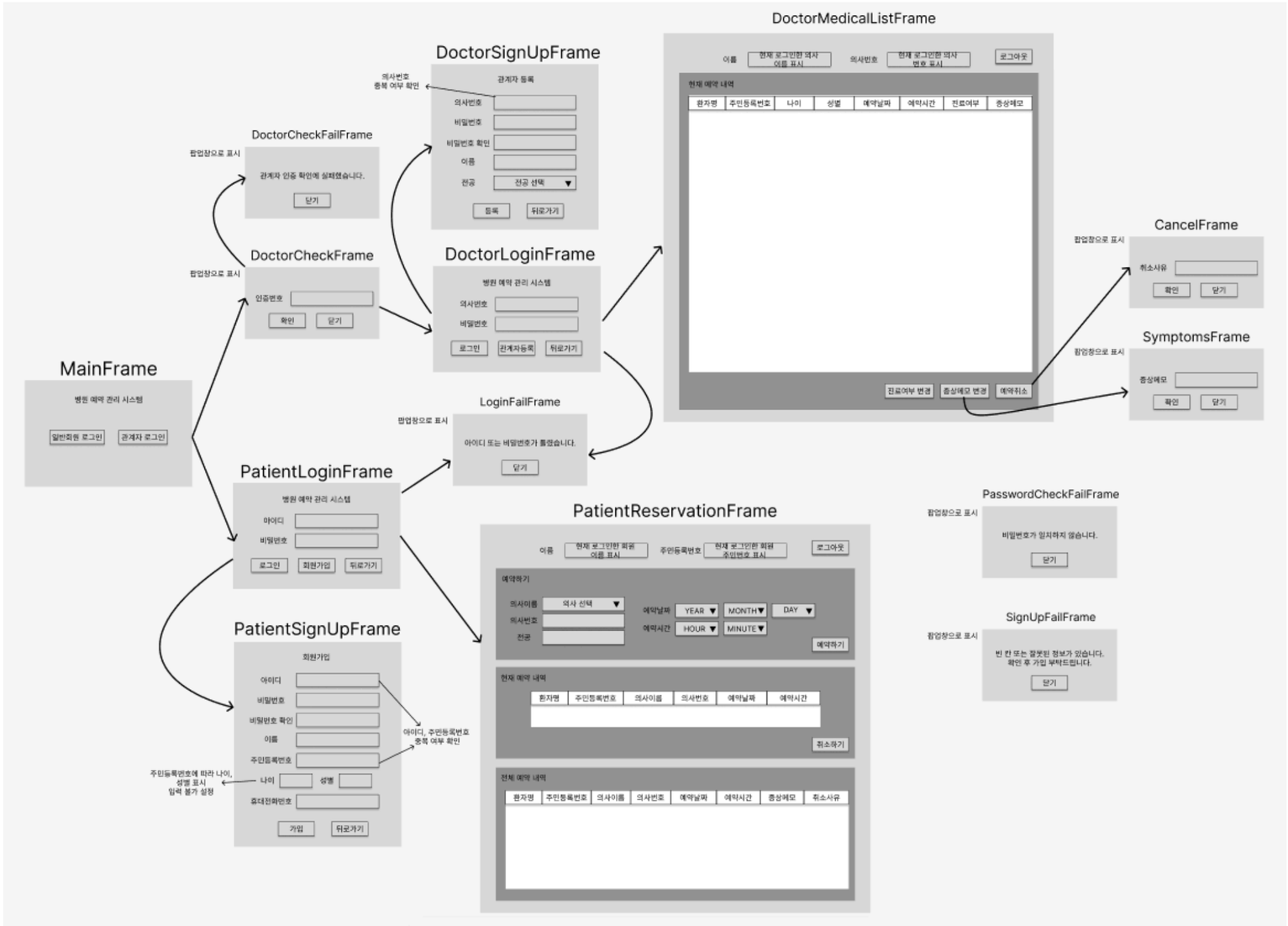
2. 프로젝트 분석

2-2. Usecase Diagram



2. 프로젝트 분석

2-3. UI Mockup Design



3. 프로젝트 설계

3-1. ERD

Patient — 일반회원

🔑	주민등록번호	identity_num	VARCHAR2(14)	NOT NULL	PRIMARY KEY
	아이디	patient_id	VARCHAR2(8)	NOT NULL	UNIQUE
	비밀번호	patient_pwd	VARCHAR2(20)	NOT NULL	Comment
	이름	patient_name	VARCHAR2(8 CHAR)	NOT NULL	Comment
	나이	patient_age	NUMBER(3)	NOT NULL	Comment
	성별	patient_gender	VARCHAR2(1)	NOT NULL	CHECK IN('M','F')
	휴대전화번호	patient_phone	VARCHAR2(13)	NOT NULL	Comment
	가입날짜	created_at	VARCHAR2(19)	NOT NULL	YYYY-MM-DD HH24:MI:SS

Reservation — 예약

🔑	예약날짜	reserv_date	VARCHAR2(10)	NOT NULL	PRIMARY KEY YYYY.MM.DD
🔑	예약시간	reserv_time	VARCHAR2(5)	NOT NULL	PRIMARY KEY HH:MM
🔑	주민등록번호	identity_num	VARCHAR2(14)	NOT NULL	FOREIGN KEY
🔑	의사번호	doctor_num	VARCHAR2(6)	NOT NULL	PRIMARY KEY FOREIGN KEY
	진료여부	medical_check	VARCHAR2(1)	NOT NULL	CHECK IN('Y','N') DEFAULT 'N'
	증상메모	symptoms_memo	VARCHAR2(50)	NULL	DEFAULT ''
	취소사유	cancel_reason	VARCHAR2(50)	NULL	DEFAULT ''

Doctor — 의사

🔑	의사번호	doctor_num	VARCHAR2(6)	NOT NULL	PRIMARY KEY
	비밀번호	doctor_pwd	VARCHAR2(20)	NOT NULL	Comment
	이름	doctor_name	VARCHAR2(8 CHAR)	NOT NULL	Comment
	전공	doctor_major	VARCHAR2(5 CHAR)	NOT NULL	Comment

3. 프로젝트 설계

3-2. 테이블 설계 (1/3)

Table		일반회원 Patient					
No	Column		Data Type	Key		Null	Constraint	비고
	컬럼 설명	컬럼명	데이터 타입	기본키	외래키		제약 조건	
1	주민등록번호	identity_num	VARCHAR2(14)	PK		NOT NULL		
2	아이디	patient_id	VARCHAR2(8)			NOT NULL	UNIQUE	
3	비밀번호	patient_pwd	VARCHAR2(20)			NOT NULL		
4	이름	patient_name	VARCHAR2(8 CHAR)			NOT NULL		
5	나이	patient_age	NUMBER(3)			NOT NULL		
6	성별	patient_gender	VARCHAR2(1)			NOT NULL	CHECK	IN('M', 'F')
7	휴대전화번호	patient_phone	VARCHAR2(13)			NOT NULL		
8	가입날짜	created_at	VARCHAR2(19)			NOT NULL		SYSDATE, 'YYYY-MM-DD HH24:MI:SS'

SQL Plus를 사용하여
Patient 테이블 생성

```
SQL> CREATE TABLE patient (
  2  identity_num VARCHAR2(14) PRIMARY KEY,
  3  patient_id VARCHAR2(8) UNIQUE NOT NULL,
  4  patient_pwd VARCHAR2(20) NOT NULL,
  5  patient_name VARCHAR2(8 CHAR) NOT NULL,
  6  patient_age NUMBER(3) NOT NULL,
  7  patient_gender VARCHAR2(1) NOT NULL CHECK (patient_gender IN('M','F')),
  8  patient_phone VARCHAR2(13) NOT NULL,
  9  created_at VARCHAR2(19) NOT NULL);

테이블이 생성되었습니다.

SQL> DESC patient;
 이름                                      널?      유형
 ----------------------------------------- -------- ----------------------------
 IDENTITY_NUM                              NOT NULL VARCHAR2(14)
 PATIENT_ID                                NOT NULL VARCHAR2(8)
 PATIENT_PWD                               NOT NULL VARCHAR2(20)
 PATIENT_NAME                              NOT NULL VARCHAR2(8 CHAR)
 PATIENT_AGE                               NOT NULL NUMBER(3)
 PATIENT_GENDER                            NOT NULL VARCHAR2(1)
 PATIENT_PHONE                             NOT NULL VARCHAR2(13)
 CREATED_AT                                NOT NULL VARCHAR2(19)
```

3. 프로젝트 설계

3-2. 테이블 설계 (2/3)

Table	의사 Doctor						
No	Column		Data Type	Key		Null	Constraint	비고
	컬럼 설명	컬럼명	데이터 타입	기본키	외래키		제약 조건	
1	의사번호	doctor_num	VARCHAR2(6)	PK		NOT NULL		
2	비밀번호	doctor_pwd	VARCHAR2(20)			NOT NULL		
3	이름	doctor_name	VARCHAR2(8 CHAR)			NOT NULL		
4	전공	doctor_major	VARCHAR2(5 CHAR)			NOT NULL		내과, 일반외과, 정형외과, 소아과 안과, 이비인후과, 피부과 비뇨기과, 정신과, 성형외과

SQL Plus를 사용하여
Doctor 테이블 생성

```
SQL> CREATE TABLE doctor (
  2  doctor_num VARCHAR2(6) PRIMARY KEY,
  3  doctor_pwd VARCHAR2(20) NOT NULL,
  4  doctor_name VARCHAR2(8 CHAR) NOT NULL,
  5  doctor_major VARCHAR2(5 CHAR) NOT NULL);

테이블이 생성되었습니다.

SQL> DESC doctor;
 이름                                      널?      유형
 ----------------------------------------- -------- ----------------------------
 DOCTOR_NUM                                NOT NULL VARCHAR2(6)
 DOCTOR_PWD                                NOT NULL VARCHAR2(20)
 DOCTOR_NAME                               NOT NULL VARCHAR2(8 CHAR)
 DOCTOR_MAJOR                              NOT NULL VARCHAR2(5 CHAR)
```

3. 프로젝트 설계

3-2. 테이블 설계 (3/3)

Table	예약 Reservation						
No	Column		Data Type	Key		Null	Constraint	비고
	컬럼 설명	컬럼명	데이터 타입	기본키	외래키		제약 조건	
1	예약날짜	reserv_date	VARCHAR2(10)	PK		NOT NULL		YYYY.MM.DD
2	예약시간	reserv_time	VARCHAR2(5)	PK		NOT NULL		HH:MM
3	주민등록번호	identity_num	VARCHAR2(14)		FK	NOT NULL		REFERENCES patient(identity_num)
4	의사번호	doctor_num	VARCHAR2(6)	PK	FK	NOT NULL		REFERENCES doctor(doctor_num)
5	진료여부	medical_check	VARCHAR2(1)			NOT NULL	CHECK	IN('Y', 'N') DEFAULT 'N'
6	증상메모	symptoms_memo	VARCHAR2(50)					DEFAULT ''
7	취소사유	cancel_reason	VARCHAR2(50)					DEFAULT ''

SQL Plus를 사용하여
Reservation 테이블 생성

```
SQL> CREATE TABLE reservation (
  2  reserv_date VARCHAR2(10),
  3  reserv_time VARCHAR2(5),
  4  identity_num VARCHAR2(14) NOT NULL,
  5  doctor_num VARCHAR2(6) NOT NULL,
  6  medical_check VARCHAR2(1) DEFAULT 'N' CHECK (medical_check IN('Y','N')) NOT NULL,
  7  symptoms_memo VARCHAR2(50) DEFAULT '',
  8  cancel_reason VARCHAR2(50) DEFAULT '',
  9  CONSTRAINT reservation_pk PRIMARY KEY(reserv_date, reserv_time, doctor_num),
 10  CONSTRAINT reservation_iden_fk FOREIGN KEY(identity_num) REFERENCES patient(identity_num),
 11  CONSTRAINT reservation_docnum_fk FOREIGN KEY(doctor_num) REFERENCES doctor(doctor_num));

테이블이 생성되었습니다.

SQL> DESC reservation;
 이름                                      널?      유형
 ----------------------------------------- -------- ----------------------------
 RESERV_DATE                               NOT NULL VARCHAR2(10)
 RESERV_TIME                               NOT NULL VARCHAR2(5)
 IDENTITY_NUM                              NOT NULL VARCHAR2(14)
 DOCTOR_NUM                                NOT NULL VARCHAR2(6)
 MEDICAL_CHECK                             NOT NULL VARCHAR2(1)
 SYMPTOMS_MEMO                                      VARCHAR2(50)
 CANCEL_REASON                                      VARCHAR2(50)
```

3. 프로젝트 설계

3-3. 클래스 설계 (1/3)

0. main 패키지

패키지	main		
클래스	Main		
변수	변수명	변수타입	설명
	mainframe	MainFrame	MainFrame 객체 주소 (메인화면 Frame 생성)

1. dto 패키지

패키지	dto			
클래스	PatientDTO			
멤버 변수	변수명	접근제어자	변수타입	설명
	identityNum	private	String	주민등록번호
	id	private	String	아이디
	pwd	private	String	비밀번호
	name	private	String	이름
	age	private	int	나이
	gender	private	String	성별
	phone	private	String	휴대전화번호
	createdAt	private	String	가입날짜
메서드	메서드명	접근제어자	리턴타입	매개변수 정
	getIdentityNum	public	String	
	setIdentityNum	public	void	String identity
	getId	public	String	
	setId	public	void	String id
	getPwd	public	String	
	setPwd	public	void	String pw
	getName	public	String	
	setName	public	void	String nam
	getAge	public	int	
	setAge	public	void	int age
	getGender	public	String	
	setGender	public	void	String genc
	getPhone	public	String	
	setPhone	public	void	String pho
	getCreatedAt	public	String	
	setCreatedAt	public	void	String create
	toString	public	String	

패키지	dto			
클래스	ReservationDTO			
멤버 변수	변수명	접근제어자	변수타입	설명
	date	private	String	예약날짜
	time	private	String	예약시간
	identityNum	private	String	예약자 주민번호
	doctorNum	private	String	의사번호
	medicalCheck	private	String	진료여부
	symptomsMemo	private	String	증상메모
	cancelReason	private	String	취소사유
	메서드명	접근제어자	리턴타입	매개변수 정의
	getDate	public	String	
	setDate	public	void	String date
	getTime	public	String	
	setTime	public	void	String time
	getIdentityNum	public	String	
	setIdentityNum	public	void	String identityNum
	getDoctorNum	public	String	
	setDoctorNum	public	void	String doctorNum
	getMedicalCheck	public	String	
	setMedicalCheck	public	void	String medicalCheck
	getSymptomsMemo	public	String	
	setSymptomsMemo	public	void	String symptomsMemo
	getCancelReason	public	String	
	setCancelReason	public	void	String cancelReason
	toString	public	String	

패키지	dto				
클래스	DoctorDTO				
멤버 변수	변수명	접근제어자	변수타입	설명	
	num	private	String	의사번호	
	pwd	private	String	비밀번호	
	name	private	String	이름	
	major	private	String	전공	
메서드	메서드명	접근제어자	리턴타입	매개변수 정의	설명
	getNum	public	String		의사번호 Getter
	setNum	public	void	String num	의사번호 Setter
	getPwd	public	String		비밀번호 Getter
	setPwd	public	void	String pwd	비밀번호 Setter
	getName	public	String		이름 Getter
	setName	public	void	String name	이름 Setter
	getMajor	public	String		전공 Getter
	setMajor	public	void	String major	전공 Setter
	toString	public	String		의사 정보 리턴

예약날짜 Getter
예약날짜 Setter
예약시간 Getter
예약시간 Setter
예약자 주민번호 Getter
예약자 주민번호 Setter
의사번호 Getter
의사번호 Setter
진료여부 Getter
진료여부 Setter
증상메모 Getter
증상메모 Setter
취소사유 Getter
취소사유 Setter
예약 정보 리턴

3. 프로젝트 설계

3-3. 클래스 설계 (2/3)

2. dao 패키지

인터페이스

패키지	dao				
인터페이스	DBdao				
	메서드명	접근제어자	리턴타입	매개변수 정의	설명
추상 메서드	patientSignUp	public	void	PatientDTO patientdto	일반회원(환자) 회원가입
	patientLogin	public	PatientDTO	String findId String findPwd	일반회원(환자) 로그인
	patientIdCheck	public	boolean	String findId	회원가입 시 아이디 중복 체크
	patientIdenCheck	public	boolean	String findIden	회원가입 시 주민번호 중복 체크
	doctorSignUp	public	void	DoctorDTO doctordto	관계자(의사) 등록
	doctorLogin	public	DoctorDTO	String findNum String findPwd	관계자(의사) 로그인
	doctorNumCheck	public	boolean	String findNum	관계자(의사) 등록 시 의사번호 중복 체크
	doctorAll	public	ArrayList<DoctorDTO>		예약 시 모든 의사 목록 조회
	patientOne	public	PatientDTO	String findIden	일반회원 한명의 정보 조회
	doctorOne	public	DoctorDTO	String findNum	의사 한명의 정보 조회
	reservationAdd	public	void	ReservationDTO reservdto	예약하기
	reservationCurrent	public	ReservationDTO	String findIden	일반회원(환자)이 현재 예약한 내역 조회
	reservationCurrentDoctor	public	ReservationDTO	String findDate String findTime String findNum	선택한 의사에게 예약하려는 날짜/시간이 겹치는 내역 조회
	reservationDel	public	void	String findIden	현재 예약한 내역 취소
	reservationMod	public	void	ReservationDTO reservdto String modMemo	진료여부, 증상메모 변경
	reservationCancel	public	void	ReservationDTO reservdto String modReason	취소사유 변경
	reservationPatientAll	public	ArrayList<ReservationDTO>	String findIden	로그인한 일반회원(환자)이 예약한 모든 내역 조회
	reservationDoctorAll	public	ArrayList<ReservationDTO>	String findNum	로그인한 관계자(의사)에게 예약한 모든 내역 조회

implements

DAO

패키지	dao				
클래스	HospitalDAO			인터페이스 implements	DBdao
멤버 변수	변수명	접근제어자	변수타입	설명	
	dbDriver	private	String	DB 드라이버 이름	
	dbName	private	String	DB ID	
	dbPwd	private	String	DB Password	
	dbURL	private	String	DB URL	
	conn	private	Connection	DB 연결 자원	
	hospitalDAO	public	HospitalDAO	싱글톤 작업을 위한 자기 자신의 객체 주소	
메서드	메서드명	접근제어자	리턴타입	매개변수 정의	설명
	HospitalDAO	public			HospitalDAO 생성자
	getInstance	public	HospitalDAO		HospitalDAO 싱글톤 작업
	init	private	void		DB 드라이버 로드
	conn	private	boolean		DB 연결 로드
	patientSignUp	public	void	PatientDTO patientdto	일반회원(환자) DB 등록
	patientLogin	public	PatientDTO	String findId String findPwd	일반회원(환자) 로그인 시 해당 정보가 DB에 있는지 조회
	patientIdCheck	public	boolean	String findId	일반회원(환자) 등록 시 ID가 DB에 있는지 조회 (중복체크)
	patientIdenCheck	public	boolean	String findIden	일반회원(환자) 등록 시 주민번호가 DB에 있는지 조회 (중복체크)
	doctorSignUp	public	void	DoctorDTO doctordto	관계자(의사) DB 등록
	doctorLogin	public	DoctorDTO	String findNum String findPwd	관계자(의사) 로그인 시 해당 정보가 DB에 있는지 조회
	doctorNumCheck	public	boolean	String findNum	관계자(의사) 등록 시 의사번호가 DB에 있는지 조회 (중복체크)
	doctorAll	public	ArrayList<DoctorDTO>		예약 시 모든 의사 정보 전체 조회
	patientOne	public	PatientDTO	String findIden	일반회원(환자) 한명의 정보 조회
	doctorOne	public	DoctorDTO	String findNum	관계자(의사) 한명의 정보 조회
	reservationAdd	public	void	ReservationDTO reservdto	예약 DB 등록
	reservationCurrent	public	ReservationDTO	String findIden	일반회원(환자)이 현재 예약한 예약 내역 조회
	reservationCurrentDoctor	public	ReservationDTO	String findDate String findTime String findNum	선택한 의사에게 예약하려는 날짜/시간이 겹치는지 조회
	reservationDel	public	void	String findIden	일반회원(환자)이 현재 예약한 예약 취소
	reservationMod	public	void	ReservationDTO reservdto String modMemo	예약 내역의 진료여부, 증상메모 변경
	reservationCancel	public	void	ReservationDTO reservdto String modReason	예약 내역의 취소사유 변경
	reservationPatientAll	public	ArrayList<ReservationDTO>	String findIden	로그인한 일반회원(환자)이 예약한 모든 내역 조회
	reservationDoctorAll	public	ArrayList<ReservationDTO>	String findNum	로그인한 관계자(의사)에게 예약한 모든 내역 조회

3. 프로젝트 설계

3-3. 클래스 설계 (3/3)

3. gui 패키지

JFrame extends

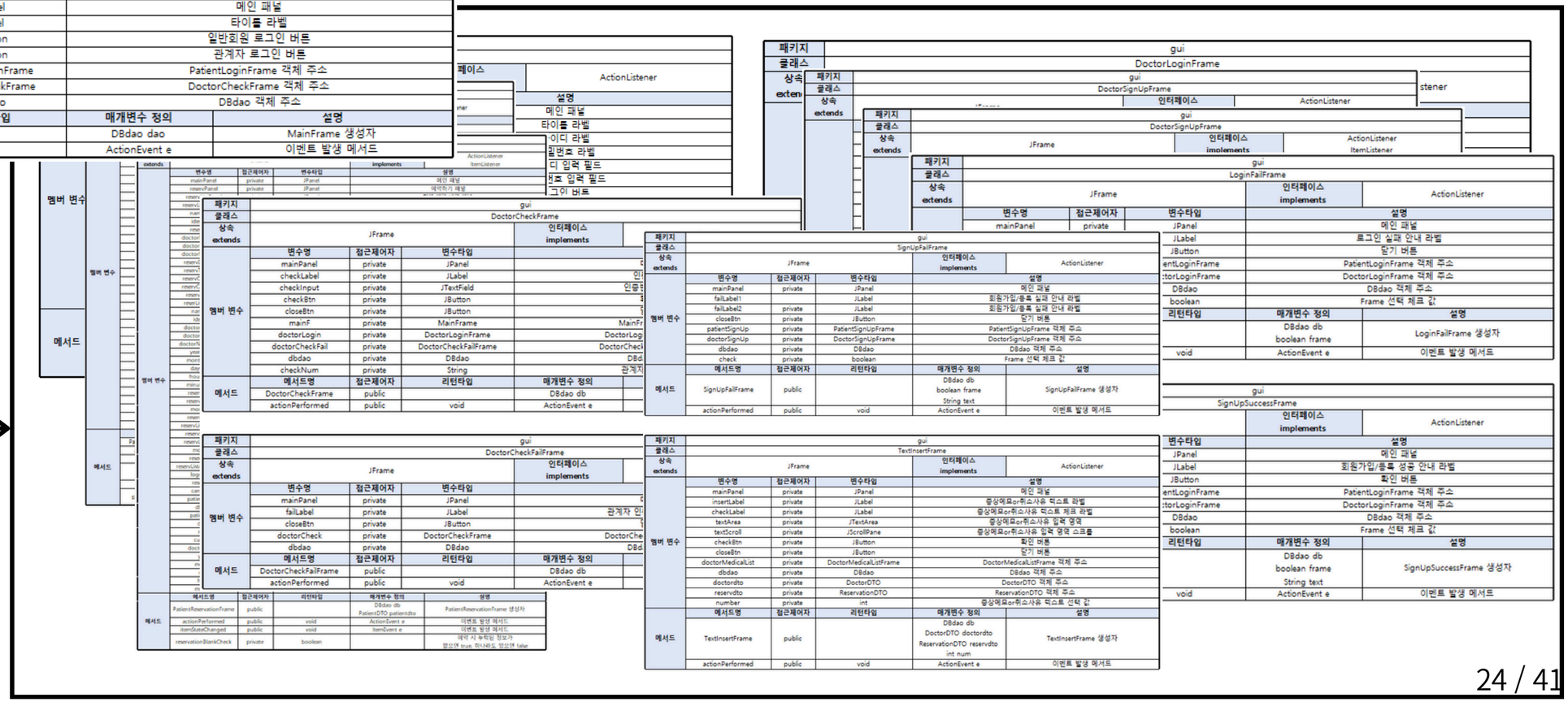
implements

ActionListener, ItemListener

총 13개의 프레임 Class

MainFrame

패키지	gui				
클래스	MainFrame				
상속 extends	JFrame			인터페이스 implements	ActionListener
멤버 변수	변수명	접근제어자	변수타입	설명	
	mainPanel	private	JPanel	메인 패널	
	titleLabel	private	JLabel	타이틀 라벨	
	patientLoginBtn	private	JButton	일반회원 로그인 버튼	
	doctorLoginBtn	private	JButton	관계자 로그인 버튼	
	patientLogin	private	PatientLoginFrame	PatientLoginFrame 객체 주소	
	doctorCheck	private	DoctorCheckFrame	DoctorCheckFrame 객체 주소	
	dbdao	private	DBdao	DBdao 객체 주소	
메서드	메서드명	접근제어자	리턴타입	매개변수 정의	설명
	MainFrame	public		DBdao dao	MainFrame 생성자
	actionPerformed	public	void	ActionEvent e	이벤트 발생 메서드



4. 프로그램 구현

4-1. DTO 클래스 사용

PatientDTO

```
public class PatientDTO {
    private String identityNum = null;  // 주민등록번호
    private String id = null;           // 아이디
    private String pwd = null;          // 비밀번호
    private String name = null;         // 이름
    private int age = 0;                // 나이
    private String gender = null;       // 성별
    private String phone = null;        // 휴대전화번호
    private String createdAt = null;    // 가입날짜

    // 주민등록번호 Getter/Setter
    public String getIdentityNum() {
        return identityNum;
    }
    public void setIdentityNum(String identityNum) {
        this.identityNum = identityNum;
    }
```

DoctorDTO

```
public class DoctorDTO {
    private String num = null;      // 의사번호
    private String pwd = null;      // 비밀번호
    private String name = null;     // 이름
    private String major = null;    // 전공

    // 의사번호 Getter/Setter
    public String getNum() {
        return num;
    }
    public void setNum(String num) {
        this.num = num;
    }
```

ReservationDTO

```
public class ReservationDTO {
    private String date = null;             // 예약날짜
    private String time = null;             // 예약시간
    private String identityNum = null;      // 예약자 주민번호
    private String doctorNum = null;        // 의사번호
    private String medicalCheck = null;     // 진료여부
    private String symptomsMemo = null;     // 증상메모
    private String cancelReason = null;     // 취소사유

    // 예약날짜 Getter/Setter
    public String getDate() {
        return date;
    }
    public void setDate(String date) {
        this.date = date;
    }
```

Java의 다른 클래스에서 각 DTO 객체를 사용할 수 있도록 하고,
DB에 데이터 CRUD 작업을 할 때도
각 DTO 객체의 정보를 가져와 사용할 수 있도록
회원, 의사, 예약 정보가 담긴 데이터를 전송하는 객체
DTO 클래스로 나누어 만들었다.

각 DTO 클래스 내에서 사용하는 변수를 은닉화하기 위해
모든 멤버 변수를 접근제어자 private를 사용하였다.

...

다른 클래스에서 DTO 객체에 접근하려고 할 때,
DTO의 멤버 변수를 호출하여 사용할 수 있도록
Getter, Setter 메서드를 만들어 다른 클래스에서도
접근이 가능하도록 만들었다.

4. 프로그램 구현

4-2. DB DAO Interface 사용 (1/2)

```java
package dao;

import java.util.ArrayList;

public interface DBdao {
    // 추상메서드
    public void patientSignUp(PatientDTO patientdto);
```

```java
package dao;

import java.sql.Connection;

public class HospitalDAO implements DBdao {
    // 일반회원(환자) 등록
    @Override
    public void patientSignUp(PatientDTO patientdto) {
        if (conn()) {
            System.out.println("★ 데이터베이스 연결 성공");
            try {
                // 주민등록번호,아이디,비밀번호,이름,나이,성별,휴대전화번호,가입날짜
                String sql = "INSERT INTO patient VALUES(?,?,?,?,?,?,?,"
                        + "TO_CHAR(SYSDATE, 'YYYY-MM-DD HH24:MI:SS'))";
                PreparedStatement pstmt = conn.prepareStatement(sql);
                pstmt.setString(1, patientdto.getIdentityNum());
                pstmt.setString(2, patientdto.getId());
                pstmt.setString(3, patientdto.getPwd());
                pstmt.setString(4, patientdto.getName());
                pstmt.setInt(5, patientdto.getAge());
                pstmt.setString(6, patientdto.getGender());
                pstmt.setString(7, patientdto.getPhone());

                int resultInt = pstmt.executeUpdate();
                if (resultInt > 0) {
                    conn.commit();
                } else {
                    conn.rollback();
```

DB 작업을 위한 DAO를
DBdao 인터페이스로 생성하고,
HospitalDAO에서 구현한 인터페이스를 사용
→ HospitalDAO implements DBdao

4. 프로그램 구현

4-2. DB DAO Interface 사용 (2/2)

```java
package dao;

import java.util.ArrayList;

public interface DBdao {
    // 추상메서드
    public void patientSignUp(PatientDTO patientdto);
```

```java
package dao;

import java.sql.Connection;

public class HospitalDAO implements DBdao {
    // 일반회원(환자) 등록
    @Override
    public void patientSignUp(PatientDTO patientdto) {
        if (conn()) {
            System.out.println("★ 데이터베이스 연결 성공");
            try {
                // 주민등록번호,아이디,비밀번호,이름,나이,성별,휴대전화번호,가입날짜
                String sql = "INSERT INTO patient VALUES(?,?,?,?,?,?,?,"
                        + "TO_CHAR(SYSDATE, 'YYYY-MM-DD HH24:MI:SS'))";
                PreparedStatement pstmt = conn.prepareStatement(sql);
                pstmt.setString(1, patientdto.getIdentityNum());
                pstmt.setString(2, patientdto.getId());
                pstmt.setString(3, patientdto.getPwd());
                pstmt.setString(4, patientdto.getName());
                pstmt.setInt(5, patientdto.getAge());
                pstmt.setString(6, patientdto.getGender());
                pstmt.setString(7, patientdto.getPhone());

                int resultInt = pstmt.executeUpdate();
                if (resultInt > 0) {
                    conn.commit();
                } else {
                    conn.rollback();
```

DBdao 인터페이스에서 추상 메서드를 생성, HospitalDAO에서 DBdao의 추상 메서드를 재정의(Override)하여 사용하였다.

4. 프로그램 구현

4-3. Frame 분리

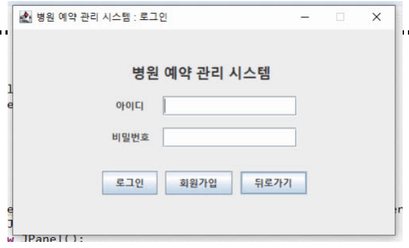
gui 패키지의 모든 Frame은 JFrame을 상속받는다.

...

gui 패키지에서 여러개의 Frame으로 프레임 분리한다.
이후 각 프레임에서 ActionListener 인터페이스를 사용하고,
ActionListener가 연결된 버튼 클릭 시 setVisible 메서드를 이용해
현재 Frame은 감추고, 다른 Frame을 표시한다.

MainFrame

```
public class MainFrame extends JFrame implements ActionListener {
    // 이벤트 발생 메서드
    @Override
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == patientLoginBtn) {
            System.out.println("메인 화면 -> 일반회원 로그인 화면");
            if (patientLogin == null) {
                patientLogin = new PatientLoginFrame(dbdao);
            }
            this.setVisible(false);
            patientLogin.setVisible(true);
        } else if (e.getSource() == doctorLoginBtn) {
            System.out.println("메인 화면 -> 관계자 인증 화면");
            if (doctorCheck == null) {
                doctorCheck = new DoctorCheckFrame(dbdao);
            }
            this.setVisible(false);
            doctorCheck.setVisible(true);
        }
    }
```



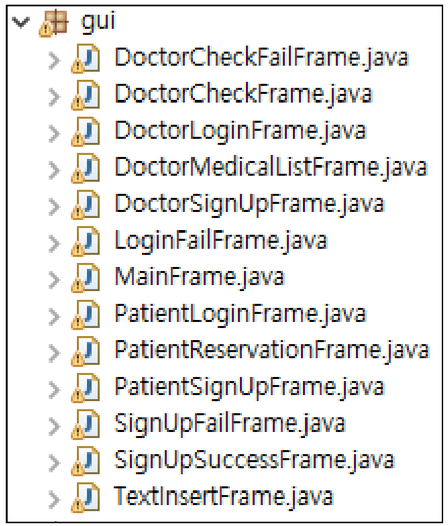
PatientLoginFrame

```
public class PatientLoginFrame extends JFrame implements ActionListener {
    private JPanel mainPanel = new JPanel();

    private JLabel titleLabel = new JLabel("병원 예약 관리 시스템");
    private JLabel idLabel = new JLabel("아이디");
    private JLabel pwdLabel = new JLabel("비밀번호");

    private JTextField idInput = new JTextField();
    private JPasswordField pwdInput = new JPasswordField();

    private JButton loginBtn = new JButton("로그인");
    private JButton signupBtn = new JButton("회원가입");
    private JButton backBtn = new JButton("뒤로가기");

    private MainFrame mainF = null;
    private PatientSignUpFrame patientSignUp = null;
    private PatientReservationFrame patientReservation = null;
    private LoginFailFrame loginFail = null;

    private DBdao dbdao = null;
    private PatientDTO patientdto = null;

    public PatientLoginFrame(DBdao db) {

    // 이벤트 발생 메서드
    public void actionPerformed(ActionEvent e) {

    // 로그인 시 DB에 회원 정보가 있다면 true, 없다면 false
    private boolean LoginCheck() {
}
```

DoctorCheckFrame

```
public class DoctorCheckFrame extends JFrame implements ActionListener {
    private JPanel mainPanel = new JPanel();

    private JLabel checkLabel = new JLabel("인증번호");

    private JPasswordField checkInput = new JPasswordField();

    private JButton checkBtn = new JButton("확인");
    private JButton closeBtn = new JButton("닫기");

    private MainFrame mainF = null;
    private DoctorLoginFrame doctorLogin = null;
    private DoctorCheckFailFrame doctorCheckFail = null;

    private DBdao dbdao = null;

    private String checkNum = "486";            // 관계자 인증 번호

    public DoctorCheckFrame(DBdao db) {

    // 이벤트 발생 메서드
    public void actionPerformed(ActionEvent e) {
}
```

13개의 프레임으로 분리



4. 프로그램 구현

4-4. 회원가입 입력 제한 (1/2)

```
idInput.setBounds(200, 73, 110, 25);
idInput.addKeyListener(new KeyAdapter() {
    public void keyTyped(KeyEvent e) {
        JTextField leng = (JTextField) e.getSource();
        char text = e.getKeyChar();
        if (leng.getText().length() >= 8) e.consume();
        // 아스키코드값
        // 48 : 0 ~ 57 : 9
        // 65 : A ~ 90 : Z
        // 97 : a ~ 122 : z
        if (!((text >= 48 && text <= 57) || (text >= 65 && text <= 90) ||
                (text >= 97 && text <= 122))) {
            e.consume();
        }
    }
});
mainPanel.add(idInput);
pwdInput1.setBounds(200, 118, 170, 25);
pwdInput1.addKeyListener(new KeyAdapter() {
    public void keyTyped(KeyEvent e) {
        JPasswordField leng = (JPasswordField) e.getSource();
        char text = e.getKeyChar();
        String password = new String(leng.getPassword());
        if (password.length() >= 20) e.consume();
        // 아스키코드값
        // 48 : 0 ~ 57 : 9
        // 65 : A ~ 90 : Z
        // 97 : a ~ 122 : z
        if (!((text >= 48 && text <= 57) || (text >= 65 && text <= 90) ||
                (text >= 97 && text <= 122))) {
            e.consume();
        }
    }
```

회원가입 시 입력 제한을 위해
KeyListener 인터페이스의 KeyAdapter 클래스를 이용하여 키보드 내 문자를 눌렀을 때 호출되는 KeyTyped 메서드를 정의하였다.

Why? KeyListener가 아닌 KeyAdapter를 사용한 이유

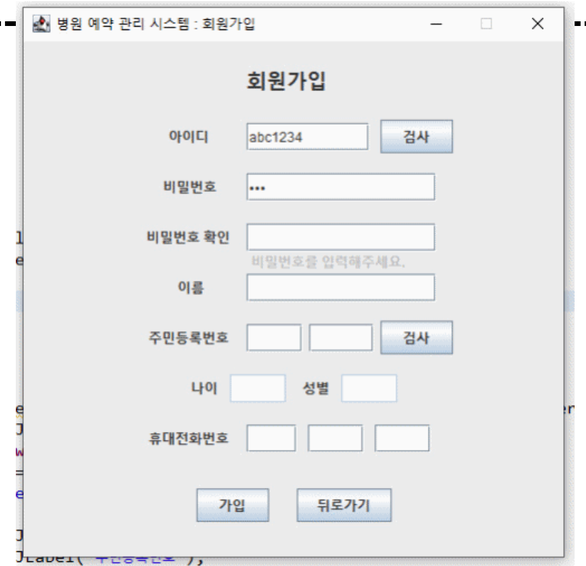
KeyListener 인터페이스를 사용하면 Key관련 메서드를 모두 재정의(Override) 해야 한다.
필요한 것은 문자만 눌렸을 때(KeyTyped 메서드)의 이벤트만 필요하므로 추상 클래스인 KeyAdapter를 사용하였다.

4. 프로그램 구현

4-4. 회원가입 입력 제한 (2/2)

```
idInput.setBounds(200, 73, 110, 25);
idInput.addKeyListener(new KeyAdapter() {
    public void keyTyped(KeyEvent e) {
        JTextField leng = (JTextField) e.getSource();
        char text = e.getKeyChar();
        if (leng.getText().length() >= 8) e.consume();
        // 아스키코드값
        // 48 : 0 ~ 57 : 9
        // 65 : A ~ 90 : Z
        // 97 : a ~ 122 : z
        if (!((text >= 48 && text <= 57) || (text >= 65 && text <= 90) ||
                (text >= 97 && text <= 122))) {
            e.consume();
        }
    }
});
mainPanel.add(idInput);
pwdInput1.setBounds(200, 118, 170, 25);
pwdInput1.addKeyListener(new KeyAdapter() {
    public void keyTyped(KeyEvent e) {
        JPasswordField leng = (JPasswordField) e.getSource();
        char text = e.getKeyChar();
        String password = new String(leng.getPassword());
        if (password.length() >= 20) e.consume();
        // 아스키코드값
        // 48 : 0 ~ 57 : 9
        // 65 : A ~ 90 : Z
        // 97 : a ~ 122 : z
        if (!((text >= 48 && text <= 57) || (text >= 65 && text <= 90) ||
                (text >= 97 && text <= 122))) {
            e.consume();
        }
    }
```

1. 입력받은 Object를 각 Field에 맞게
형변환(Casting)하여 저장한다.
2. getKeyChar 메서드를 호출하여 반환받은 문자를 저장한다.
...
1에 저장된 Object의 String 타입의 문자열을 getText 메서드로 호출하여
반환하고, 반환된 문자열의 길이를 사용
2에 저장된 char 타입의 문자를 사용
→ 1과 2의 값을 비교연산자로 참/거짓을 구분하고
조건문이 true라면 다음 코드를 실행한다.
↓
입력된 문자를 consume 메서드로 소멸시킨다.



4. 프로그램 구현

4-5. 회원가입 시 나이/성별 자동 입력 (1/3)

```
else if (e.getSource() == idenCheckBtn) { // 주민번호 중복 검사 버튼 클릭
  System.out.println("주민번호 중복 검사");
  if (idenInput1.getText().isEmpty() && idenInput2.getText().isEmpty()) {
      idenCheck = true;
      idenCheckLabel.setText("주민번호를 입력해주세요.");
      idenCheckLabel.setForeground(Color.RED);
      return;
  } else if (!(idenInput1.getText().length() == 6 && idenInput2.getText().le
      idenCheck = true;
      idenCheckLabel.setText("올바른 주민번호가 아닙니다.");
      idenCheckLabel.setForeground(Color.RED);
      ageText.setText("");
      genderText.setText("");
      return;
  } else {
      identity = idenInput1.getText() + "-" + idenInput2.getText();
      idenCheck = dbdao.patientIdenCheck(identity);
  }
  if (idenCheck) {
      idenCheckLabel.setText("중복된 주민번호가 있습니다.");
      idenCheckLabel.setForeground(Color.RED);
      ageText.setText("");
      genderText.setText("");
  } else {
      idenCheck = false;
      idenCheckLabel.setText("OK");
      idenCheckLabel.setForeground(Color.GREEN);
      ageText.setText(Integer.toString(identityAge(idenInput1.getText())));
      genderText.setText(identityGender(idenInput2.getText()));
  }
```

주민번호의 JTextField에 값을 입력하고 검사 버튼을 눌렀을 때
정상적으로 주민번호가 입력됐다면
나이/성별이 자동으로 입력될 수 있도록
identityAge 메서드와 identityGender 메서드를 호출한다.

...

호출할 때 identityAge 메서드에는 주민번호 앞자리(생년월일)을 넘겨주고, identityGender 메서드에는 주민번호 뒷자리를 넘겨준다.

```
// 생월 생일을 비교하여 나이를 구하는 메서드
private int identityAge(String identity) {
// 생년을 비교하여 나이를 구하는 메서드
private int birthdayAge(String identity, String year) {
```

```
// 주민번호 뒷자리를 통해 성별을 구하는 메서드
private String identityGender(String identity) {
```

4. 프로그램 구현

4-5. 회원가입 시 나이/성별 자동 입력 (2/3)

```java
// 주민등록번호 입력 시 나이/성별 자동 기입
// 생월 생일을 비교하여 나이를 구하는 메서드
private int identityAge(String identity) {
    int age = 0;
    int myMonth = Integer.parseInt(identity.substring(2,4));
    int myDay = Integer.parseInt(identity.substring(4,6));

    Calendar current = Calendar.getInstance();
    String year = Integer.toString(current.get(Calendar.YEAR));
    int month = current.get(Calendar.MONTH) + 1;
    int day = current.get(Calendar.DAY_OF_MONTH);

    if (birthdayAge(identity, year) == 0) {
        return age;
    }
    if (((myMonth * 100) + myDay) > ((month * 100) + day)) {
        age = birthdayAge(identity, year) - 1;
    } else {
        age = birthdayAge(identity, year);
    }
    return age;
}
// 생년을 비교하여 나이를 구하는 메서드
private int birthdayAge(String identity, String year) {
    int age = 0;
    String myYear = null;
    if (Integer.parseInt(identity.substring(0,2)) == Integer.parseInt(year.substring(2,4))) {
        return age;
    }
    if (Integer.parseInt(identity.substring(0,2)) >= Integer.parseInt(year.substring(2,4))) {
        myYear = "19" + identity.substring(0,2);
        age = Integer.parseInt(year) - Integer.parseInt(myYear);
        return age;
    } else {
        myYear = "20" + identity.substring(0,2);
        age = Integer.parseInt(year) - Integer.parseInt(myYear);
        return age;
    }
}
```

호출

현재 시간의 날짜를 구하기 위해
Calendar 객체를 사용하여 년, 월, 일을 저장한다.
(월은 0~11까지의 값을 반환하므로 +1을 해준다.)

...

생월/생일과 현재월/현재일을 비교하여
birthdayAge 메서드를 호출한다.
호출할 때 identityAge 메서드가 받은
주민번호와 현재 날짜의 년을 넘겨준다.

...

생년과 현재연도를 비교하여
생년이 같으면 19XX년대생, 다르면 20XX년대생으로 저장한다.
현재연도와 생년의 값을 String → int 로 형변환(Casting)하여
현재연도에 생년을 뺀 값을 나이로 계산하여 반환한다.

...

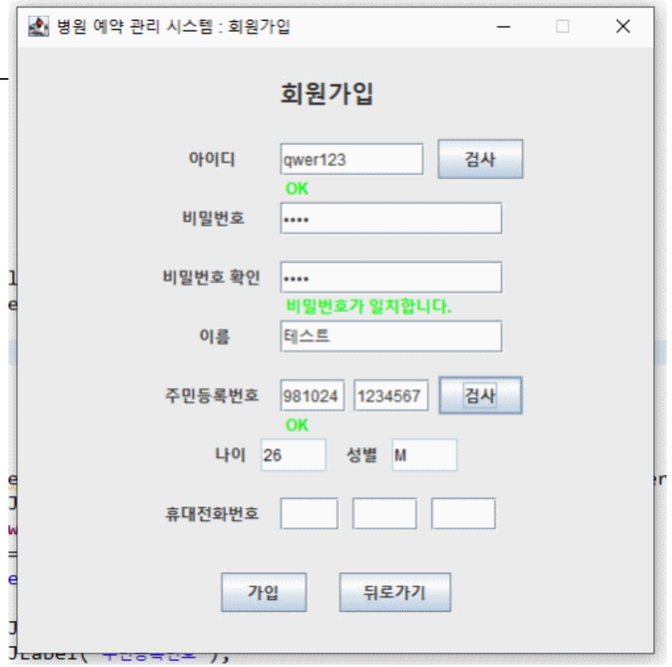
마지막으로 현재월/현재일보다 생월/생일이
크다면 (나이 - 1) 값을 반환하고,
아니라면 나이만 반환한다.

4. 프로그램 구현

4-5. 회원가입 시 나이/성별 자동 입력 (3/3)

```
// 주민번호 뒷자리를 통해 성별을 구하는 메서드
private String identityGender(String identity) {
    if ((Integer.parseInt(identity.substring(0,1)) % 2) != 0) {
        return "M";
    } else {
        return "F";
    }
}
```

주민번호 뒷자리의 0번째 index의 값을
String → int 타입으로 형변환(Casting)한다.
...
형변환된 정수값을 2로 나눠서
홀수면 남자(M), 짝수면 여자(F)를 반환한다.



4. 프로그램 구현

4-6. Java AWT Choice

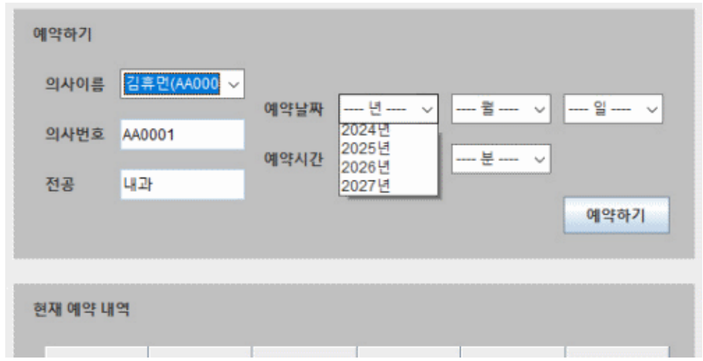
```
// 예약날짜(년)
reservDateLabel.setBounds(200, 55, 50, 50);
reservPanel.add(reservDateLabel);
yearChoice.setBounds(260, 68, 80, 25);
yearChoice.addItem("---- 년 ----");
for (int i=0; i<=3; i++) {
    yearChoice.addItem((current.get(Calendar.YEAR) + i) + "년");
}
yearChoice.addItemListener(this);
reservPanel.add(yearChoice);
// 예약날짜(월)
monthChoice.setBounds(350, 68, 80, 25);
monthChoice.addItem("---- 월 ----");
for (int i=1; i<=12; i++) {
    if (i < 10) {
        monthChoice.addItem("0" + i + "월");
    } else {
        monthChoice.addItem(i + "월");
    }
}
monthChoice.addItemListener(this);
reservPanel.add(monthChoice);
// 예약날짜(일)
dayChoice.setBounds(440, 68, 80, 25);
dayChoice.addItem("---- 일 ----");
dayChoice.addItemListener(this);
reservPanel.add(dayChoice);
```

```
@Override
public void itemStateChanged(ItemEvent e) {
    } else if (e.getSource() == yearChoice) {        // 예약날짜(년) 선택 시
        if (!(e.toString().contains("-"))) {
            int yearIndex = yearChoice.getSelectedItem().indexOf("년");
            year = yearChoice.getSelectedItem().substring(0, yearIndex);

            int yearSelect = Integer.parseInt(year);
            current.set(Calendar.YEAR, yearSelect);

            monthChoice.select(0);
            dayChoice.select(0);
        } else {
            year = null;
        }
    } else if (e.getSource() == monthChoice) {       // 예약날짜(월) 선택 시
        dayChoice.removeAll();
        dayChoice.addItem("---- 일 ----");
        if (!(e.toString().contains("-"))) {
            int monthIndex = monthChoice.getSelectedItem().indexOf("월");
            month = monthChoice.getSelectedItem().substring(0, monthIndex);

            int monthSelect = Integer.parseInt(month);
            current.set(Calendar.MONTH, (monthSelect - 1));

            dayChoice.select(0);

            for (int i=1; i<=current.getActualMaximum(Calendar.DAY_OF_MONTH); i++) {
                if (i < 10) {
                    dayChoice.addItem("0" + i + "일");
                } else {
                    dayChoice.addItem(i + "일");
                }
            }
        } else {
            month = null;
        }
    } else if (e.getSource() == dayChoice) {         // 예약날짜(일) 선택 시
        if (!(e.toString().contains("-"))) {
            int dayIndex = dayChoice.getSelectedItem().indexOf("일");
            day = dayChoice.getSelectedItem().substring(0, dayIndex);
        } else {
            day = null;
        }
    }
```

ItemListener 인터페이스의 itemStateChanged 메서드를 재정의(Override)하여
Java AWT의 Choice 컴포넌트의 아이템이 선택되면
Choice에 대한 값이 수정될 수 있도록 하였다.

...

선택할 수 있는 값의 자료형은 String으로 숫자+문자(년/월/일)로 구성되어 있으므로
숫자만 사용하기 위해 indexOf 메서드를 이용해 년/월/일의 index를 찾아
substring 메서드로 년/월/일 문자를 잘라낸다.



4. 프로그램 구현

4-7. Java Swing JTable (1/3)

```
private String[] reservListCol = {"환자명", "주민등록번호", "의사이름", "의사번호", "예약날짜", "예약시간"};
private String[][] reservListRow = null;
private DefaultTableModel modelOne = new DefaultTableModel(reservListRow, reservListCol) {
    @Override
    public boolean isCellEditable(int row, int column) {
        return false;
    }
};
private JTable reservListOne = new JTable(modelOne);
private JTableHeader reservListColName = reservListOne.getTableHeader();
private String[] reservListAllCol = {"환자명", "주민번호", "의사이름", "의사번호",
                                     "예약날짜", "예약시간", "증상메모", "취소사유"};
private String[][] reservListAllRow = null;
private DefaultTableModel modelAll = new DefaultTableModel(reservListAllRow, reservListAllCol) {
    @Override
    public boolean isCellEditable(int row, int column) {
        return false;
    }
};
private JTable reservListAll = new JTable(modelAll);
private JScrollPane reservListAllScrollPane = new JScrollPane(reservListAll);

private ArrayList<ReservationDTO> rList= null;

public PatientReservationFrame(DBdao db, PatientDTO patientdto) {
    this.rList = dbdao.reservationPatientAll(patientdto.getIdentityNum());
```

Java Swing의 JTable 컴포넌트와
해당 JTable의 테이블 관리 모델 DefaultTableModel 클래스,
테이블의 헤더를 관리하는 JTableHeader 클래스를 사용하였다.

...

테이블의 Column값은 1차원 배열에 지정된 값을 넣어줬고,
Row값은 DB에서 SELECT절을 통해 찾아낸 값을 넣어주기 위해
2차원 배열의 초기값은 null값으로 지정해주었다.

...

테이블의 값을 클릭하여 변경할 수 없도록
DefaultTableModel 객체를 생성하여
isCellEditable 메서드의 반환값을 false로 설정해주었다.

4. 프로그램 구현

4-7. Java Swing JTable (2/3)

```
private String[] reservListCol = {"환자명", "주민등록번호", "의사이름", "의사번호", "예약날짜", "예약시간"};
private String[][] reservListRow = null;
private DefaultTableModel modelOne = new DefaultTableModel(reservListRow, reservListCol) {
    @Override
    public boolean isCellEditable(int row, int column) {
        return false;
    }
};
private JTable reservListOne = new JTable(modelOne);
private JTableHeader reservListColName = reservListOne.getTableHeader();
private String[] reservListAllCol = {"환자명", "주민번호", "의사이름", "의사번호",
                                     "예약날짜", "예약시간", "증상메모", "취소사유"};
private String[][] reservListAllRow = null;
private DefaultTableModel modelAll = new DefaultTableModel(reservListAllRow, reservListAllCol) {
    @Override
    public boolean isCellEditable(int row, int column) {
        return false;
    }
};
private JTable reservListAll = new JTable(modelAll);
private JScrollPane reservListAllScrollPane = new JScrollPane(reservListAll);

private ArrayList<ReservationDTO> rList= null;

public PatientReservationFrame(DBdao db, PatientDTO patientdto) {
    this.rList = dbdao.reservationPatientAll(patientdto.getIdentityNum());
```

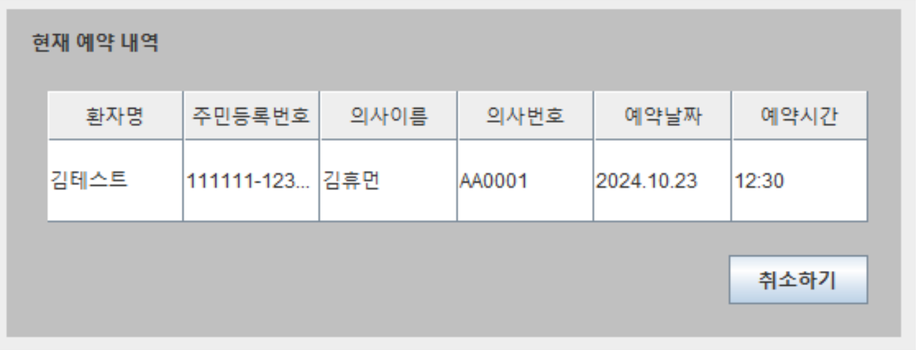
```
// 현재 예약 내역
reservListPanel.setLayout(null);
reservListPanel.setBounds(60, 285, 545, 200);
reservListPanel.setBackground(Color.LIGHT_GRAY);
reservListLabel.setBounds(15, -5, 80, 50);
reservListPanel.add(reservListLabel);
try {
    ReservationDTO reservCurrent = dbdao.reservationCurrent(patientdto.getIdentityNum());
    DoctorDTO reservDoctor = dbdao.doctorOne(reservCurrent.getDoctorNum());
    String[] rowData = new String[6];
    rowData[0] = patientdto.getName();
    rowData[1] = patientdto.getIdentityNum();
    rowData[2] = reservDoctor.getName();
    rowData[3] = reservDoctor.getNum();
    rowData[4] = reservCurrent.getDate();
    rowData[5] = reservCurrent.getTime();
    modelOne.addRow(rowData);
} catch (Exception e) {}
reservListColName.setBounds(25, 50, 500, 30);
reservListPanel.add(reservListColName);
reservListOne.setBounds(25, 80, 500, 50);
reservListOne.setRowHeight(50);
reservListPanel.add(reservListOne);
reservListPanel.add(reservCancelLabel);

cancelBtn.setBounds(440, 150, 85, 30);
cancelBtn.addActionListener(this);
reservListPanel.add(cancelBtn);
mainPanel.add(reservListPanel);
```

현재 예약 내역은 예약 DB에서 WHERE절을 통해 단 하나의 값을 SELECT절으로 찾아내기 때문에 JTableHeader 클래스의 getTableHeader 메서드로 컬럼명을 저장하였다.

...

SELECT절, WHERE절을 사용하여 찾아낸 DTO 객체의 주소를 저장하여 접근제어자가 public으로 설정된 Getter 메서드를 이용해 RowData 배열에 표시하고자 하는 데이터를 넣어주었다.

이때, DB에서 SELECT절을 통해 찾아낸 값이 없을 경우 예외가 발생하게 되는데, 예외가 발생할 경우를 대비하여 try-catch 예외처리를 사용하였다.



4. 프로그램 구현

4-7. Java Swing JTable (3/3)

```
private String[] reservListCol = {"환자명", "주민등록번호", "의사이름", "의사번호", "예약날짜", "예약시간"};
private String[][] reservListRow = null;
private DefaultTableModel modelOne = new DefaultTableModel(reservListRow, reservListCol) {
    @Override
    public boolean isCellEditable(int row, int column) {
        return false;
    }
};
private JTable reservListOne = new JTable(modelOne);
private JTableHeader reservListColName = reservListOne.getTableHeader();
private String[] reservListAllCol = {"환자명", "주민번호", "의사이름", "의사번호",
                                     "예약날짜", "예약시간", "증상메모", "취소사유"};
private String[][] reservListAllRow = null;
private DefaultTableModel modelAll = new DefaultTableModel(reservListAllRow, reservListAllCol) {
    @Override
    public boolean isCellEditable(int row, int column) {
        return false;
    }
};
private JTable reservListAll = new JTable(modelAll);
private JScrollPane reservListAllScrollPane = new JScrollPane(reservListAll);

private ArrayList<ReservationDTO> rList= null;

public PatientReservationFrame(DBdao db, PatientDTO patientdto) {
    this.rList = dbdao.reservationPatientAll(patientdto.getIdentityNum());
```

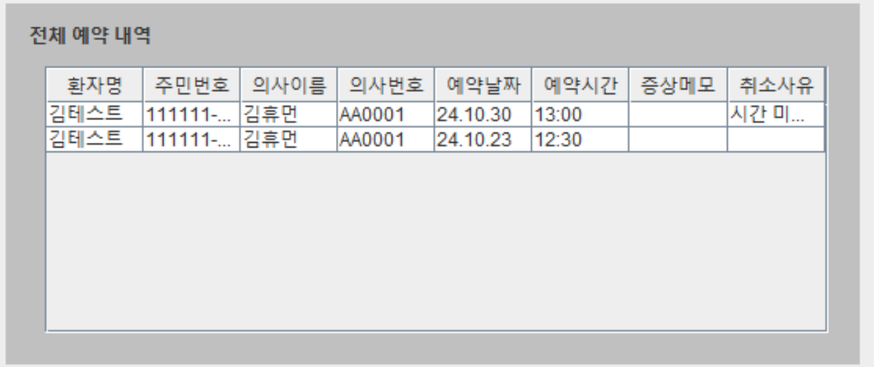
```
// 전체 예약 내역
reservListAllPanel.setLayout(null);
reservListAllPanel.setBounds(60, 505, 545, 230);
reservListAllPanel.setBackground(Color.LIGHT_GRAY);
reservListAllLabel.setBounds(15, -5, 80, 50);
reservListAllPanel.add(reservListAllLabel);
try {
    for (int i=0; i<rList.size(); i++) {
        DoctorDTO reservDoctor = dbdao.doctorOne(rList.get(i).getDoctorNum());
        String[] rowData = new String[8];
        rowData[0] = patientdto.getName();
        rowData[1] = patientdto.getIdentityNum();
        rowData[2] = reservDoctor.getName();
        rowData[3] = reservDoctor.getNum();
        rowData[4] = rList.get(i).getDate().substring(2);
        rowData[5] = rList.get(i).getTime();
        rowData[6] = rList.get(i).getSymptomsMemo();
        rowData[7] = rList.get(i).getCancelReason();
        modelAll.addRow(rowData);
    }
} catch (Exception e) {}
reservListAllScrollPane.setBounds(25, 40, 500, 170);
reservListAllPanel.add(reservListAllScrollPane);
mainPanel.add(reservListAllPanel);
```

전체 예약 내역은 예약 DB에서 WHERE절을 통해 여러 값을 SELECT절으로 찾아내기 때문에
가변길이를 지닌 컬렉션 프레임워크 ArrayList를 사용하여
ReservationDTO 객체의 주소를 저장하였다.
그리고 예약 DB에서 SELECT절로 조회한 데이터가 지정된 테이블의 크기를 넘어가서 표시될 경우를
대비하여 Java Swing의 JScrollPane 컴포넌트를 이용하여 스크롤바를 만들어주었다.

...

SELECT절, WHERE절을 사용하여 찾아낸 DTO 객체의 주소를 저장하여
접근제어자가 public으로 설정된 Getter 메서드를 이용해 RowData 배열에 표시하고자 하는 데이터
를 넣어주었다.
그리고 컬렉션 프레임워크 ArrayList에 저장된 데이터를 모두 추가하기 위해
반복문(for문)을 사용하여 ArrayList의 크기(size 메서드 사용)만큼 반복하도록 하였다.

이때, DB에서 SELECT절을 통해 찾아낸 값이 없을 경우 예외가 발생하게 되는데,
예외가 발생할 경우를 대비하여 try-catch 예외처리를 사용하였다.



5. 개발 테스트

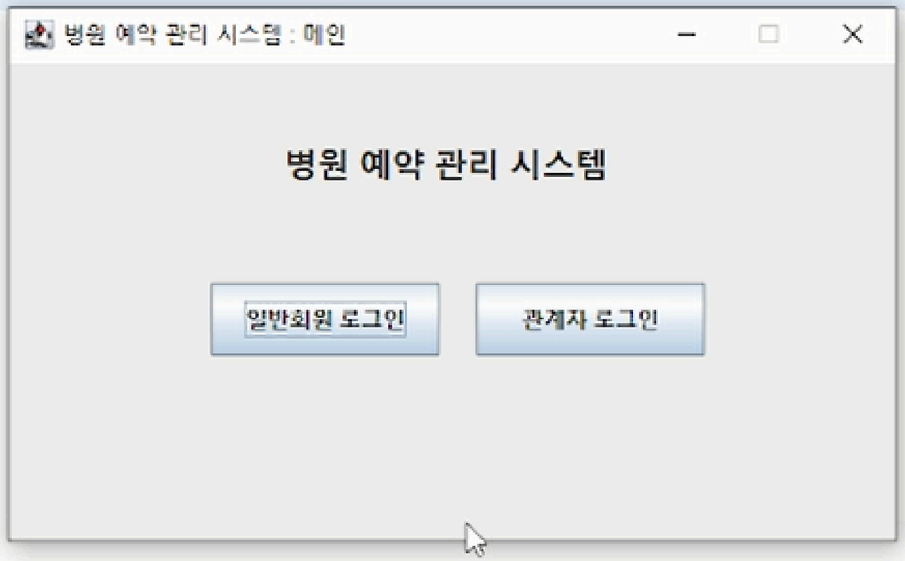
5-1. 테스트 요구사항

기능별		테스트 내용	여부
UI		모든 프레임이 원활하게 잘 넘어가는가?	Y
		인증번호, 비밀번호 입력칸의 은닉이 잘 이뤄지는가?	Y
회원가입	일반회원 (환자)	회원가입이 잘 이뤄지는가?	Y
		아이디 중복 검사가 잘 이뤄지는가?	Y
		비밀번호 확인이 잘 이뤄지는가?	Y
		주민번호 중복 검사가 잘 이뤄지는가?	Y
		주민번호를 통해 나이/성별 입력이 잘 이뤄지는가?	Y
		모든 정보를 입력해야 회원가입이 이뤄지는가?	Y
	관계자 (의사)	관계자 등록이 잘 이뤄지는가?	Y
		의사번호 중복 검사가 잘 이뤄지는가?	Y
		비밀번호 확인이 잘 이뤄지는가?	Y
		모든 정보를 입력해야 관계자 등록이 이뤄지는가?	Y
로그인	ALL	로그인이 잘 이뤄지는가?	Y
	관계자 (의사)	관계자 로그인 프레임으로 넘어갈 때, 인증번호 입력이 잘 이뤄지는가?	Y
예약	ALL	로그아웃이 잘 되는가?	Y
	일반회원 (환자)	예약할 수 있는 모든 의사가 표시되고, 선택할 수 있는가?	Y
		의사를 선택하면 선택한 의사의 정보가 표시되는가?	Y
		예약할 때 모든 정보를 선택해야 예약이 되는가?	Y
		현재 예약한 내역이 있다면 중복 예약을 할 수 없게 되어있는가?	Y
		한명의 의사, 동일한 날짜/시간대에 중복 예약을 할 수 없게 되어있는가?	Y
		현재 예약한 내역을 취소할 수 있는가?	Y
		현재 예약한 내역, 현재까지 예약했던 나의 예약 내역들이 모두 표시되는가?	Y
	관계자 (의사)	내역 중 하나를 선택하여 진료 여부 및 증상 메모 변경이 가능한가?	Y
		내역을 선택하지 않으면 증상 메모 및 취소를 할 수 없게 되어있는가?	Y
		현재 예약된 내역을 취소 사유를 입력하고 취소할 수 있는가?	Y

5. 개발 테스트

5-2. 테스트 영상

```
    public static void main(String[] args) {
        DBdao dao = new HospitalDAO();
        MainFrame mf = new MainFrame(dao);
    }

}
```



6. 마무리

약 11일간의 프로젝트가 끝이 났습니다.
첫 일정의 주제, 팀 선정부터 마지막 일정인 PPT, 발표까지 혼자서 Java 프로젝트를 만들어 나가면서 많은 시행착오가 있었습니다.
처음 주제를 선정하는 것부터 아이디어가 잘 떠오르지 않기도 했고, 프로젝트를 진행하면서 평일에 다 할 수 있을지 걱정됐었는데 주말에도 코드를 구현하면서 시간을 많이 할애했던 것 같습니다.
나이/성별 계산을 위해 사용했던 Calendar 객체나 Choice, JTable 컴포넌트 사용에서 처음 사용해보는 것에 뜻대로 잘 이뤄지지 않아 처음에는 조금 헤맸던 것 같은데, 그래도 시간이 지나면서 정보를 찾아내고 원하는 대로 UI와 코드를 구현했던 것이 기억에 남습니다.
Java의 AWT/Swing을 다뤄보면서 UI를 구성하는 것과, 만든 UI에서 DTO, DAO를 거쳐 데이터베이스까지 데이터가 들어가는 것을 보면서 성취감을 느꼈습니다.
다소 아쉬운 점이 있다면 여유가 생겼을 때 처음 기획했던 프로젝트에서 조금씩 기능을 추가하고 싶었는데 그러지 못했던 점이 다소 아쉬웠습니다.
그래도 예정된 시간 내에 기획했던 프로젝트를 완성할 수 있어서 다행이었다고 생각합니다.

JAVA PROJECT. 241014 ~ 241024
TEAM : 박진우

감사합니다.

GitHub / Gmail

박진우 / J1NU / pj1nu2@gmail.com