컴퓨터정보공학부 2019021725 안성현

도서 관리 앱

<2021/03/04>

PT

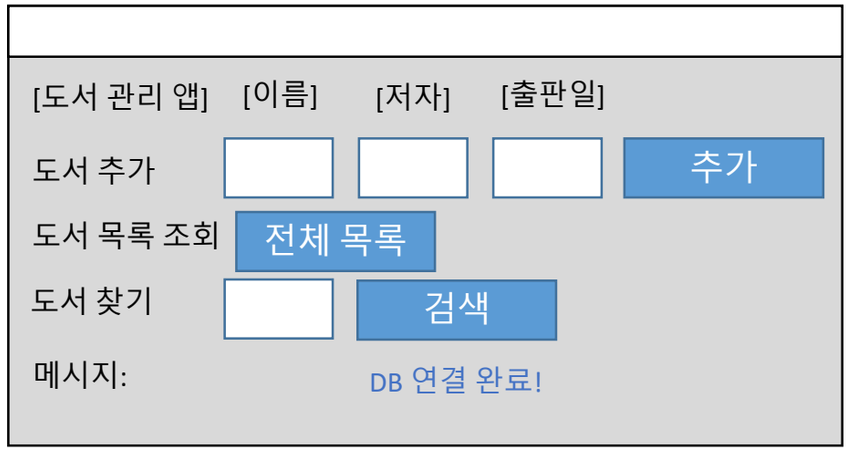
1. 설계

2. 구현

3. 실행 정보

설계

- 도서 관리 앱
  - 도서 관리 데스크톱 애플리케이션으로 (도서 추가, 도서 목록 조회, 도서 찾기) 기능을 제공한다.
  - 예상 GUI는 아래와 같다.



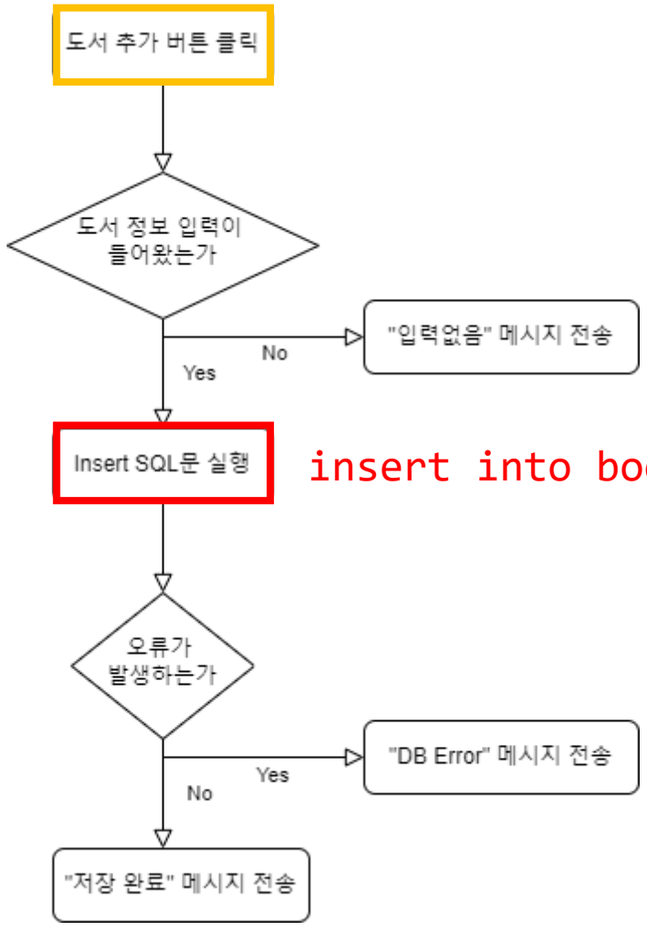
메인 윈도우 (저장, 전체 조회, 검색)

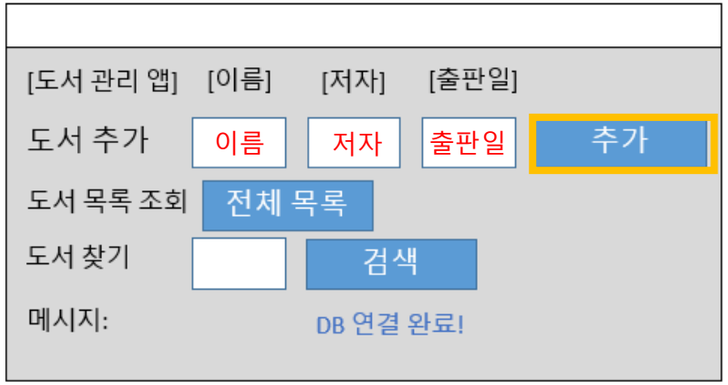
이름	저자	출판일
웹프로그래밍	천인국	2019.10.21
선형대수	김대수	2017.02.14

[전체 목록], [검색] -> 서브 윈도우 생성
(표 형태로 결과 보여줌)

설계

- 도서 관리 앱 (도서 추가)
  - 도서를 추가하는 알고리즘을 순서도(flowchart)로 간단하게 나타내면 다음과 같다.

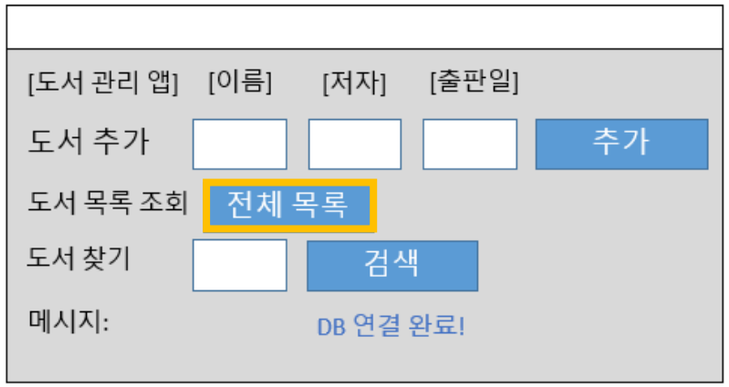




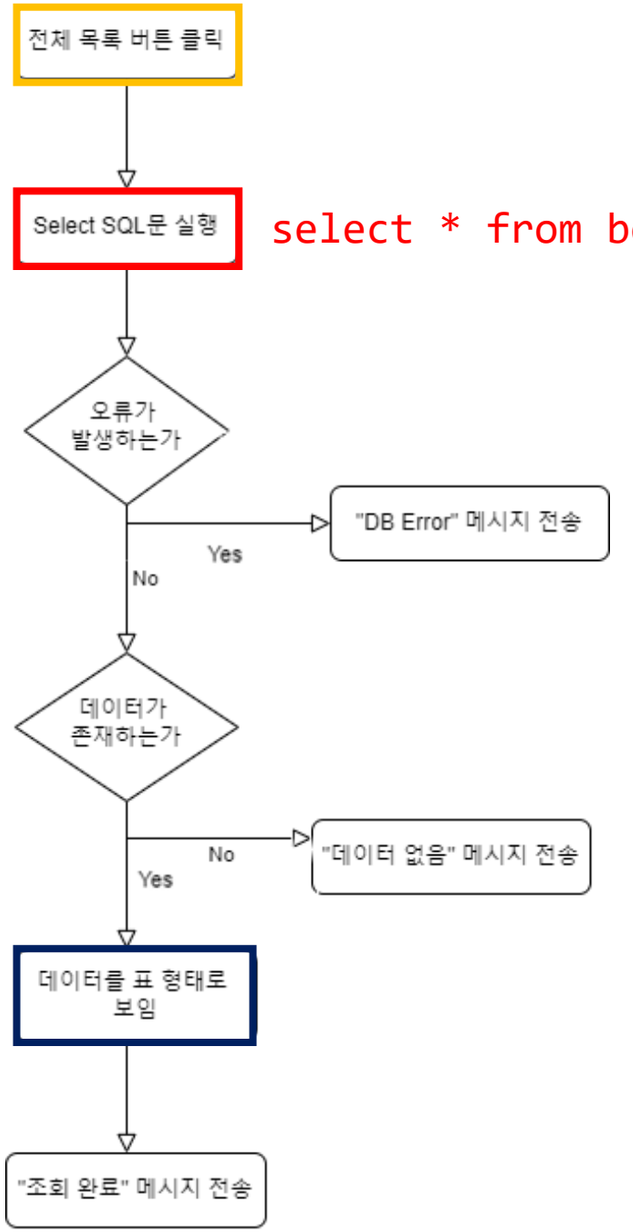

```
insert into book values(이름, 저자, 출판일);
```

설계

- 도서 관리 앱 (전체 조회)
  - 도서 목록 전체를 조회하는 알고리즘을 순서도(flowchart)로 간단하게 나타내면 다음과 같다.

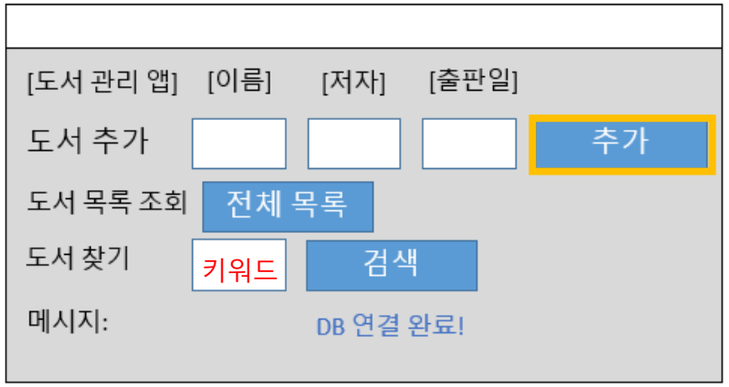


이름	저자	출판일
웹프로그래밍	천인국	2019.10.21
선형대수	김대수	2017.02.14



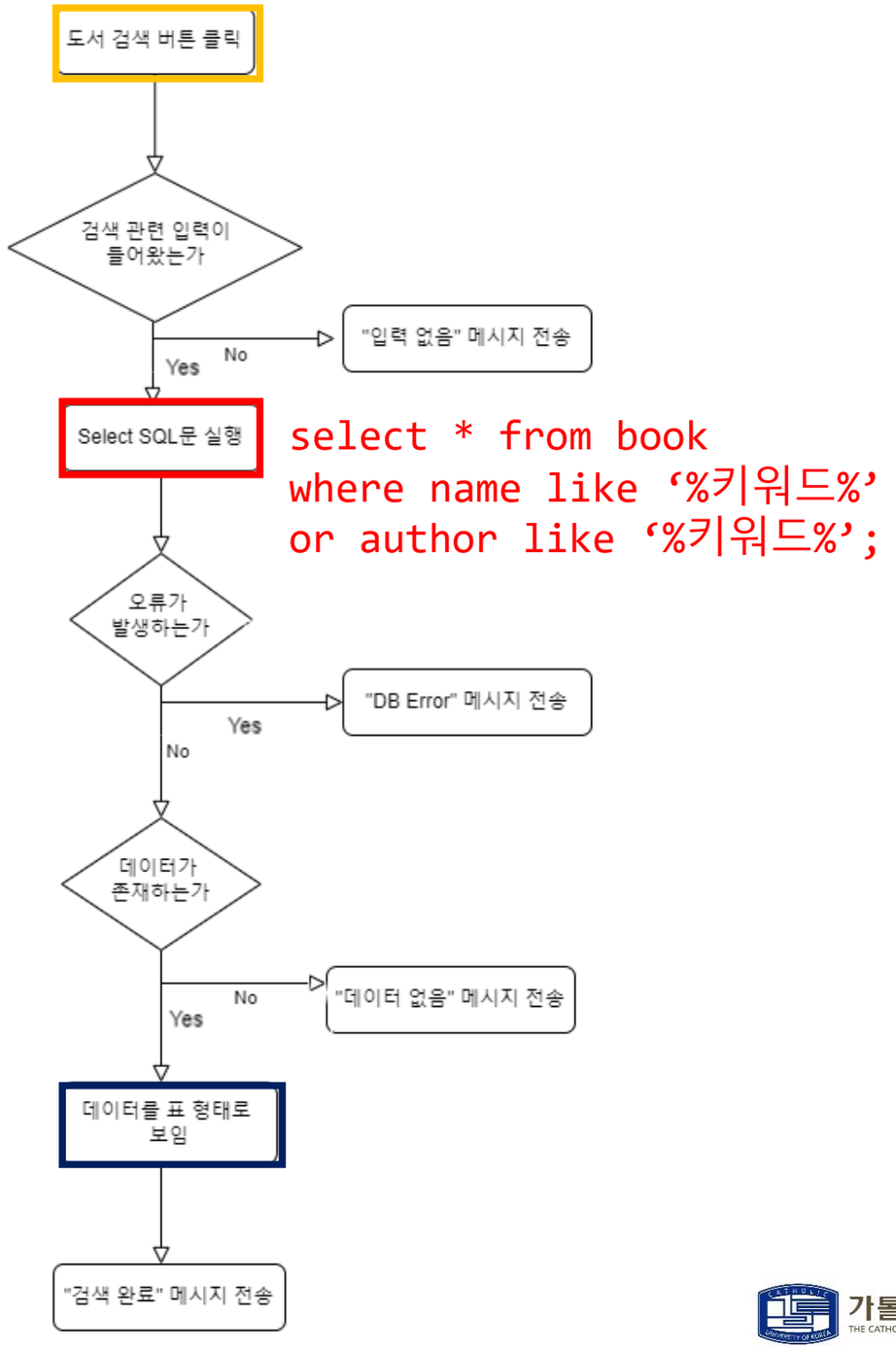
설계

- 도서 관리 앱 (도서 찾기)
  - 도서를 검색하는 알고리즘을 순서도(flowchart)로 간단하게 나타내면 다음과 같다.



이름	저자	출판일
웹프로그래밍	천인국	2019.10.21

If) 키워드='웹'




```
select * from book
where name like '%키워드%'
or author like '%키워드%';
```

구현

- MySQL 기본 설정
  - Java에서 MySQL 데이터베이스를 접근하기 위해서는 기본 설정이 필요하다.

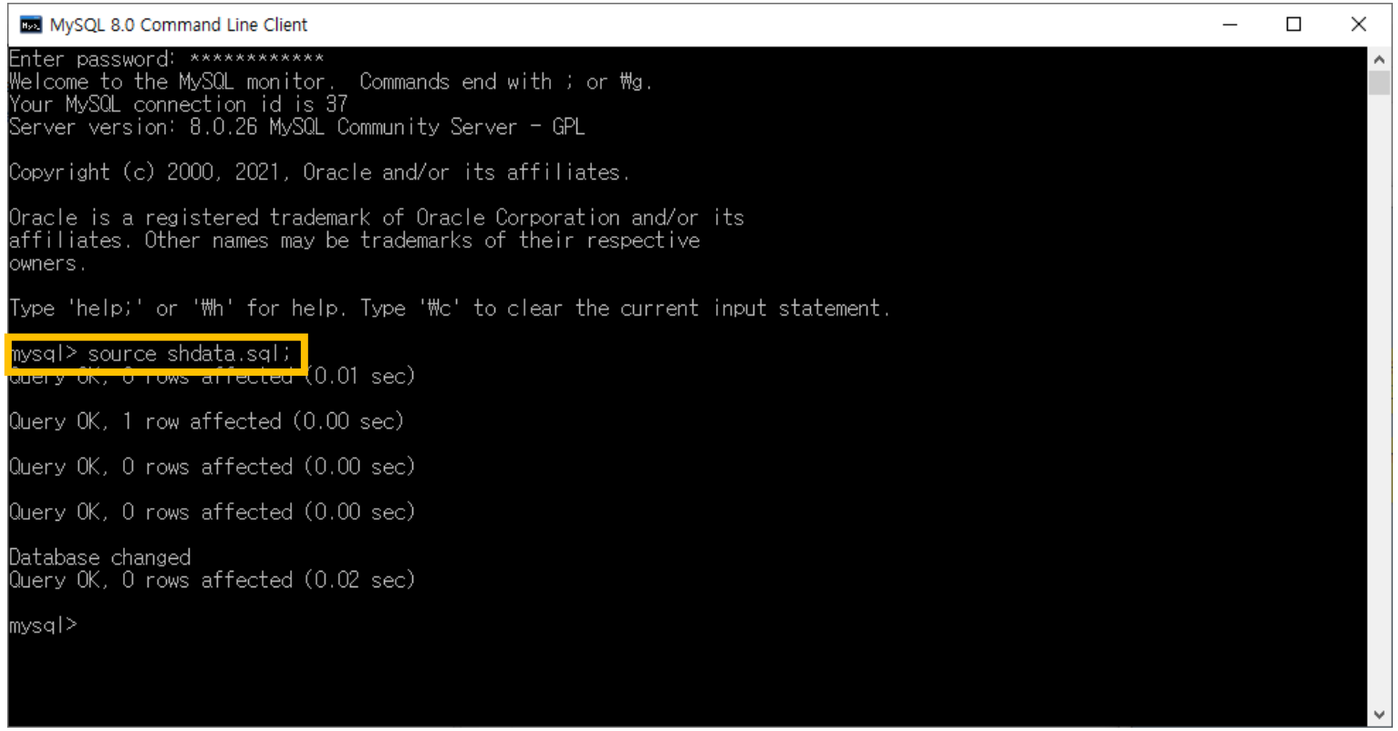
```
create user javaman@localhost identified by '0000';
create database dbjava;
grant all privileges on dbjava.* to javaman@localhost;
flush privileges;
use dbjava;
create table book(name varchar(30), author varchar(30), date varchar(30));
```

shdata.sql (initialize sql file)

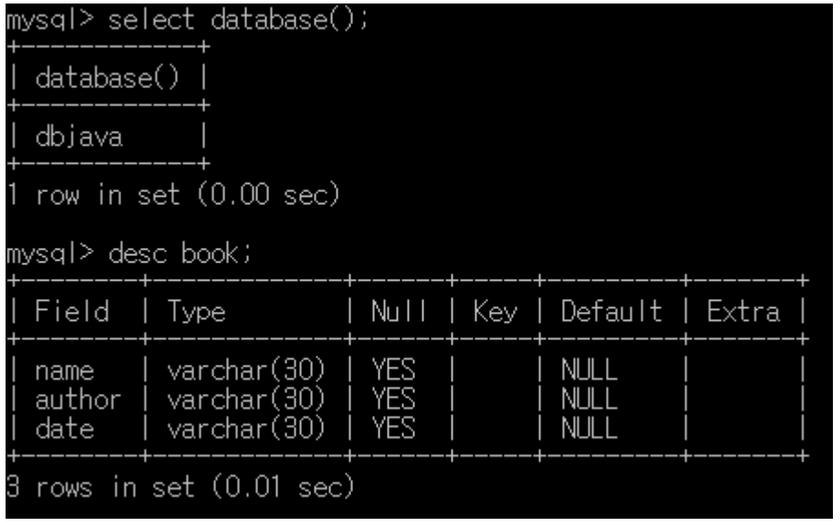
1. MySQL에 'javaman'이라는 사용자를 추가한다. 사용자의 비밀번호는 '0000'이다.
2. dbjava라는 데이터베이스를 생성한다.
3. javaman에게 dbjava의 모든 테이블에 대한 권한을 부여한다.
4. 권한을 실제로 적용한다.
5. dbjava 데이터베이스에 접속한다.
6. (name,author,date)를 속성으로 갖는 book이라는 테이블을 생성한다.

구현

- MySQL 기본 설정
  - MySQL 8.0 Command Line Client에 접속해서 shdata.sql 파일로 SQL문을 실행한다.
  - shdata.sql의 경로: C:\Program Files\MySQL\MySQL Server 8.0\bin

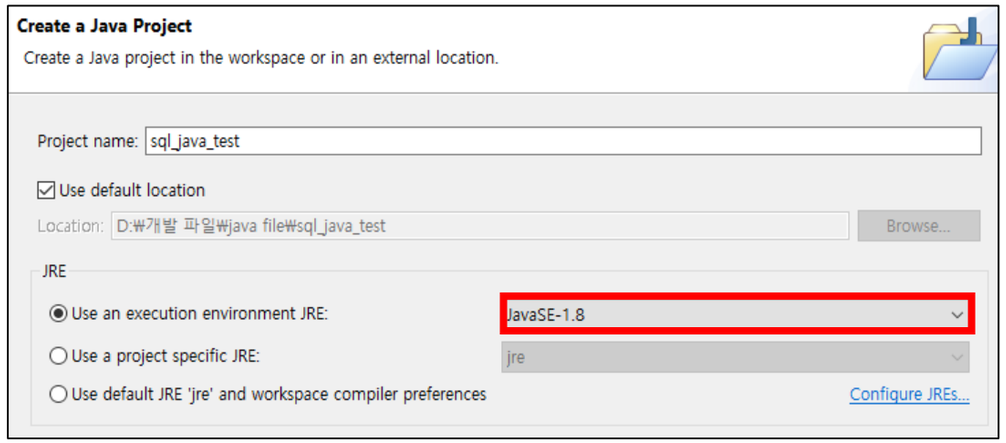


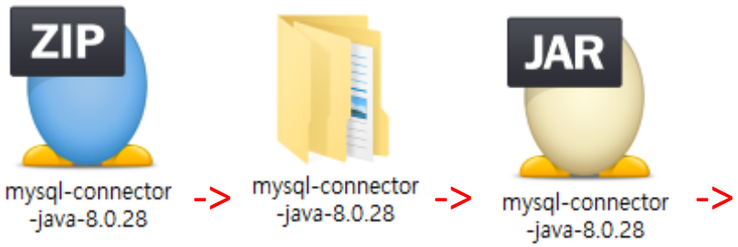
->



구현

- Eclipse 기본 설정 I
  - Project를 생성할 때 JAVA 버전은 1.8로 설정한다.
  - https://dev.mysql.com/downloads/connector/j/ 에서 **'mysql-connector.zip'**을 다운로드하고 생성한 Project에 jar파일을 적용한다.

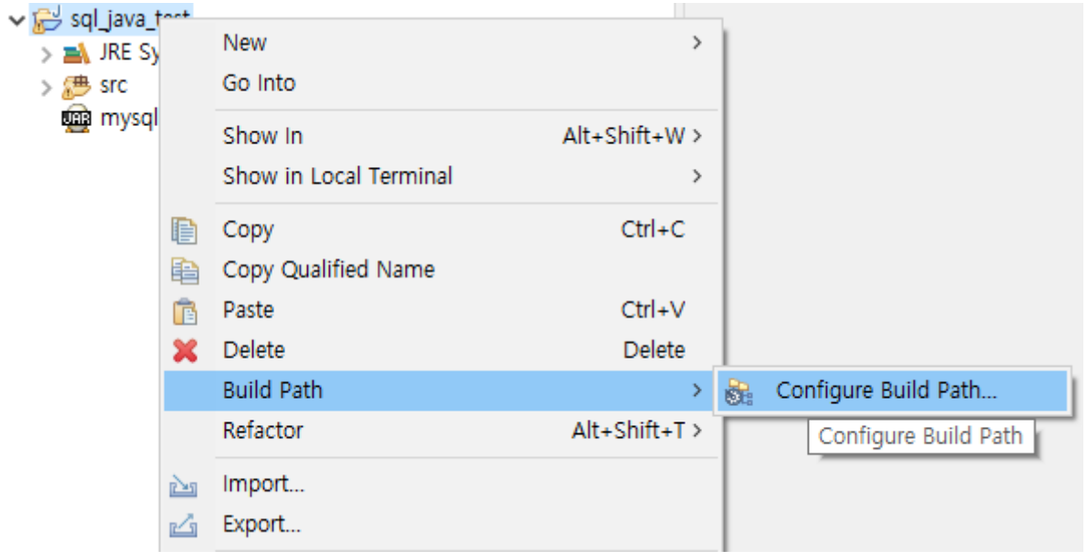


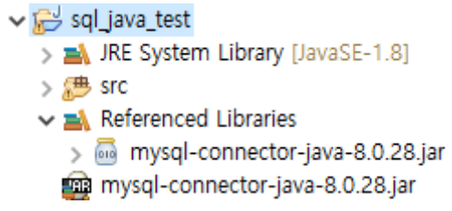


다운로드받은 jar파일을 Eclipse 프로젝트에 붙여넣기

구현

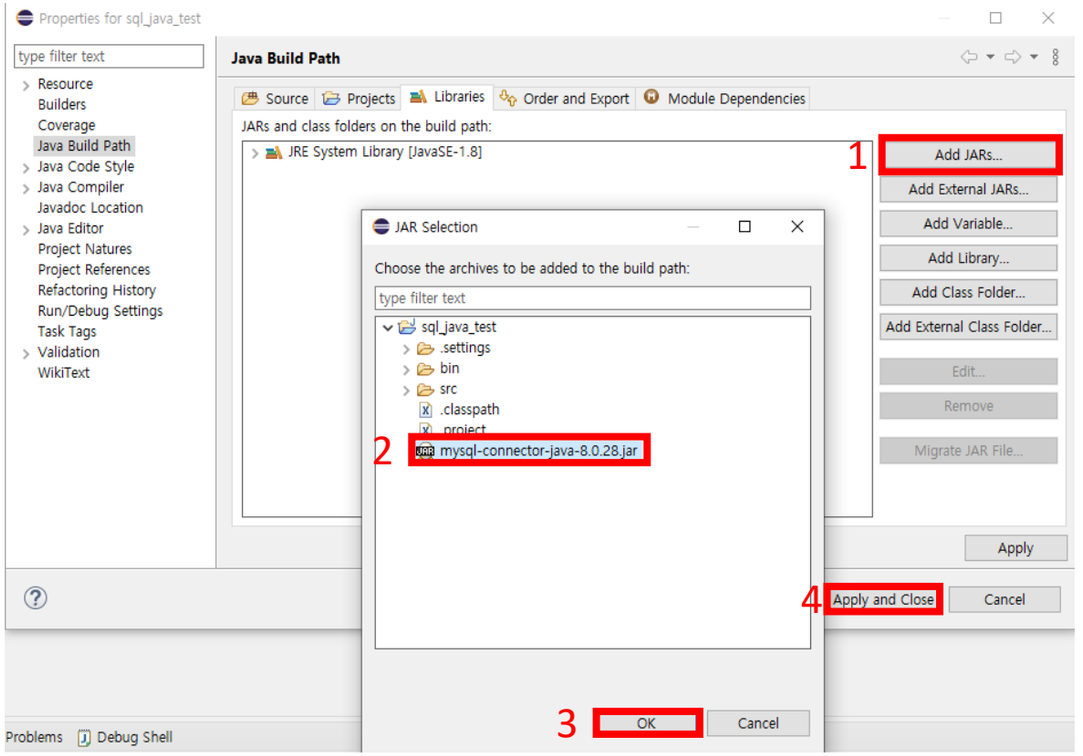
- Eclipse 기본 설정 2
  - Project -> Build Path -> Configure Build Path -> Libraries 접속 -> Add JARs -> Apply and Close





<- 정상적으로 추가되면 이런 형태

->



구현

- MySQL 연결
  - DB Connection을 위한 코드이다. ‘MySQL 기본 설정’에서 생성한 사용자(javaman)로 접속하였다.

```
// class for DB Connection
public class test {
    Connection conn;
    Statement stmt;
    ResultSet rs;

    public test() {
        // DB Connection
        try {
            // Connection을 위해 필요한 DB Driver 로드
            Class.forName("com.mysql.jdbc.Driver");
            System.out.println("DB Driver Loading OK!");

            // DB Connection
            conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/dbjava","javaman","0000");
            System.out.println("DB Connection OK!");

            // Statement 객체 생성 (Java -> DB로 SQL문 전송하는 역할, DB -> JAVA 처리 결과 전송할 때 받는 역할)
            stmt = conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);
        }
        catch(ClassNotFoundException e) {
            e.printStackTrace();
            System.out.println("DB Driver Error!");
        }
        catch(SQLException se) {
            se.printStackTrace();
            System.out.println("DB Error!");
        }
    }
```

1. DB Driver 로드

2. DB 로컬 주소와 사용자 정보를 이용해서 접속

3. SQL문을 전송하고 결과를 받기 위해 Statement객체 생성

구현

- 메인 윈도우
  - 메인 윈도우 생성과 관련된 클래스이다. Java Swing을 사용하였다. JFrame에 Component를 추가하는 방식이다.

```
// Class for Window
class frame extends JFrame{
    JLabel error_label;

    public frame(test t) {
        // 윈도우 창 설정
        setSize(500,200);
        setLocation(700,300);
        setTitle("도서관");
        setLayout(new GridLayout(5,1));
        setDefaultCloseOperation(EXIT_ON_CLOSE);

        // panel 객체 배열 생성
        JPanel pns[]=new JPanel[5];
        for(int i=0; i<4; i++) {
            pns[i]=new JPanel();
            pns[i].setLayout(new GridLayout(0,5));
        }
        pns[4]=new JPanel();
        pns[4].setLayout(new GridLayout(1,2));

        // panel 0 (소개 관련)
        JLabel[] lbs=new JLabel[4]; // Labels
        String[] strs= {"[도서 관리 앱]","[이름]","[저자]","[출판일]"};
        for(int i=0; i<4; i++) {
            lbs[i]=new JLabel(strs[i]);
            pns[0].add(lbs[i]);
        }

        // panel 1 (도서 추가 관련)
        JLabel lb1=new JLabel("도서 추가"); // Label
        pns[1].add(lb1);
        JTextField[] fields=new JTextField[3]; // TextFields
        for(int i=0; i<3; i++) {
            fields[i]=new JTextField();
            pns[1].add(fields[i]);
        }
        JButton bt1=new JButton("추가"); // Button
        bt1.addActionListener(new Action(this,t,fields[0],fields[1],fields[2]));
        pns[1].add(bt1);

        // panel 2 (도서 조회 관련)
        JLabel lb2=new JLabel("도서 목록 조회"); // Label
        JButton bt2=new JButton("전체 목록"); // Button
        Cursor cursor = bt2.getCursor();
        bt2.addActionListener(new Action(this,t));
        pns[2].add(lb2);
        pns[2].add(bt2);

        // panel 3 (도서 검색 관련)
        JLabel lb3=new JLabel("도서 찾기"); // Label
        JTextField search_field=new JTextField(); // TextField
        JButton bt3=new JButton("검색"); // Button
        bt3.addActionListener(new Action(this,t,search_field));
        pns[3].add(lb3);
        pns[3].add(search_field);
        pns[3].add(bt3);

        // panel 4 (에러 관련)
        JLabel lb4=new JLabel("메시지:"); // Label
        pns[4].add(lb4);
        error_label=new JLabel("DB 연결 완료!"); // Label
        pns[4].add(error_label);


        // 컴포넌트 배치 및 활성화
        for (int i=0; i<5; i++) {
            add(pns[i]);
        }
        setVisible(true);
    }
}
```

구현

- Action Listener 구현
  - 이벤트 처리와 관련된 인터페이스는 ActionListener이다. actionPerformed를 오버라이딩해서 이벤트 기능을 제작하였다.

```
@Override
public void actionPerformed(ActionEvent e) {
    // TODO Auto-generated method stub
    JButton act=(JButton) e.getSource(); //활동하는 버튼
    String actname=act.getText(); // 활동하는 버튼의 텍스트

    switch(actname) {
        case "추가":
            func1(); // 도서 추가 이벤트 처리 핵심 -> 테이블에 입력한 도서 추가
            break;
```

1. 도서 추가 기능

```
        case "전체 목록":
            // mini-window size, title, location
            getContentPane().removeAll(); // 기존 윈도우 삭제되고 새로운 윈도우 생성하는데 기존에 안 사라진게 있을 수 있어서 지움
            setSize(400,200);
            setTitle("전체 목록 조회");
            setLocation(1190,300);
            // table setting
            String[] header= {"이름","저자","출판일"};
            String[][] contents=func2(); // 도서 조회 이벤트 처리 핵심 -> 전체 도서 목록이 담긴 2차원 배열 반환
            if(contents != null) {
                JTable table = new JTable(contents,header);
                JScrollPane scrollpane=new JScrollPane(table);
                add(scrollpane);
                setVisible(true);
            }
            break;
```

2. 모든 도서 조회 기능

```
        case "검색":
            // mini-window size, title, location
            getContentPane().removeAll(); // 기존 윈도우 삭제되고 새로운 윈도우 생성하는데 기존에 안 사라진게 있을 수 있어서 지움
            setSize(400,200);
            setTitle("도서 검색");
            setLocation(1190,500);
            // table setting
            String[] header2= {"이름","저자","출판일"};
            String[][] contents2=func3(); // 도서 검색 이벤트 처리 핵심 -> 도서 검색한 목록이 담긴 2차원 배열 반환
            if(contents2 != null) {
                JTable table2 = new JTable(contents2,header2);
                JScrollPane scrollpane2=new JScrollPane(table2);
                add(scrollpane2);
                setVisible(true);
            }
            break;
}
```

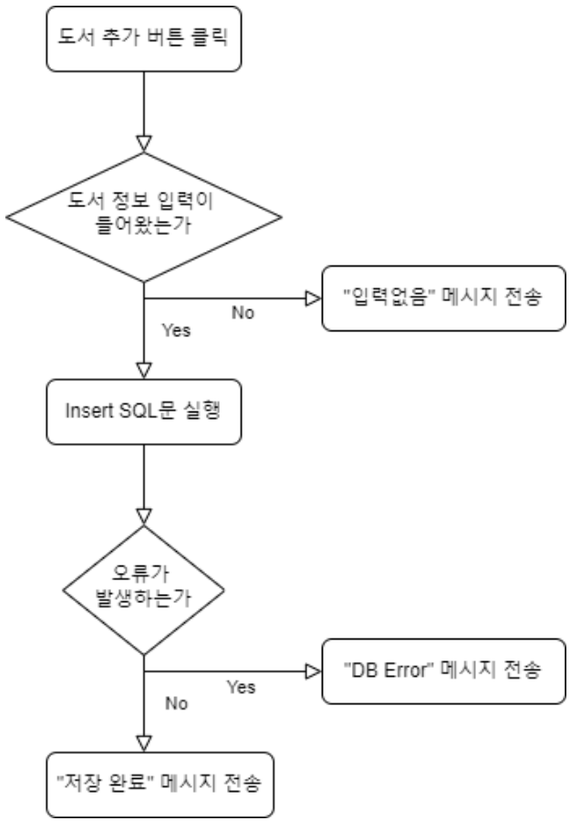
3. 도서 검색 기능

구현

- 이벤트 처리 (도서 추가)
  - [도서 추가] 버튼을 눌렀을 때 발생하는 이벤트를 처리하는 코드이다.

```
// (도서 추가 관련) 이벤트 처리
public void func1() {
    // 조건 만족 시 SQL문 실행
    try {
        String n=name.getText();
        String a=author.getText();
        String d=date.getText();

        // 입력 안 들어옴
        if (n.equals("")||a.equals("")||d.equals("")) {
            f.error_label.setText("입력 없음!");
            f.error_label.setForeground(Color.RED);
        }
        // 입력 들어옴
        else {
            // SQL문 실행
            t.stmt.executeUpdate("insert into book values('"+n+"','"+a+"','"+d+"');");
            f.error_label.setText("저장 완료!!  (전체 목록을 눌러서 확인!!)");
            f.error_label.setForeground(Color.BLUE);
            // 입력 초기화
            name.setText(null);
            author.setText(null);
            date.setText(null);
        }
    }
    // DB Error
    catch (SQLException e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
        f.error_label.setText("DB Error");
        f.error_label.setForeground(Color.RED);
    }
}
```



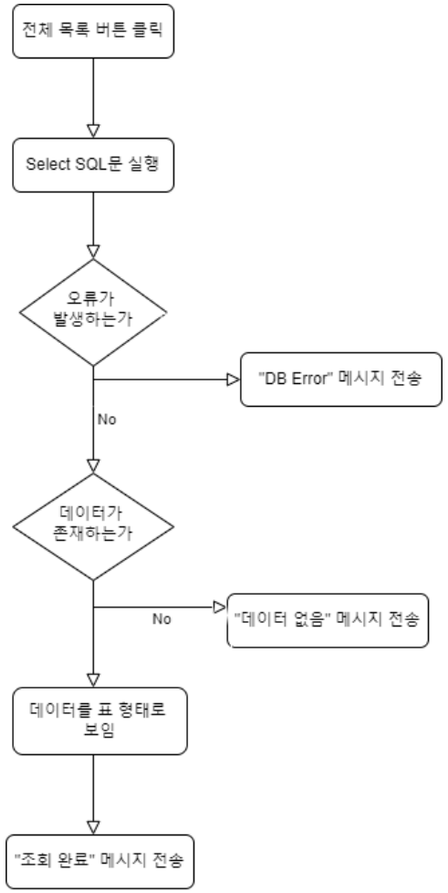
구현

- 이벤트 처리 (전체 조회)
  - [전체 목록] 버튼을 눌렀을 때 발생하는 이벤트를 처리하는 코드이다.

```
// (도서 조회 관련) 이벤트 처리
public String[][] func2() {
    // SQL문 실행
    try {
        ResultSet rs = t.stmt.executeQuery("select * from book;");
        // 데이터 확인
        int rows = 0;
        if (rs.last()) {
            rows = rs.getRow();
            rs.beforeFirst();
        }
        // 데이터 존재
        if(rows>0) {
            // 데이터를 2차원 배열에 담아서 리턴
            String[][] contents=new String[rows][3];
            int idx=0;
            while(rs.next()) {
                contents[idx][0]=rs.getString("name");
                contents[idx][1]=rs.getString("author");
                contents[idx][2]=rs.getString("date");
                idx++;
            }
            f.error_label.setText("조회 완료!!");
            f.error_label.setForeground(Color.BLUE);
            return contents;
        }
        // 데이터 존재 X
        else {
            // 빈 2차원 배열 리턴
            f.error_label.setText("데이터 없음!");
            f.error_label.setForeground(Color.RED);
            String[][] contents=new String[1][3];
            return contents;
        }
    }
```

데이터를 2차원 배열(contents)에 담아서 리턴하고 JTable에 연결함

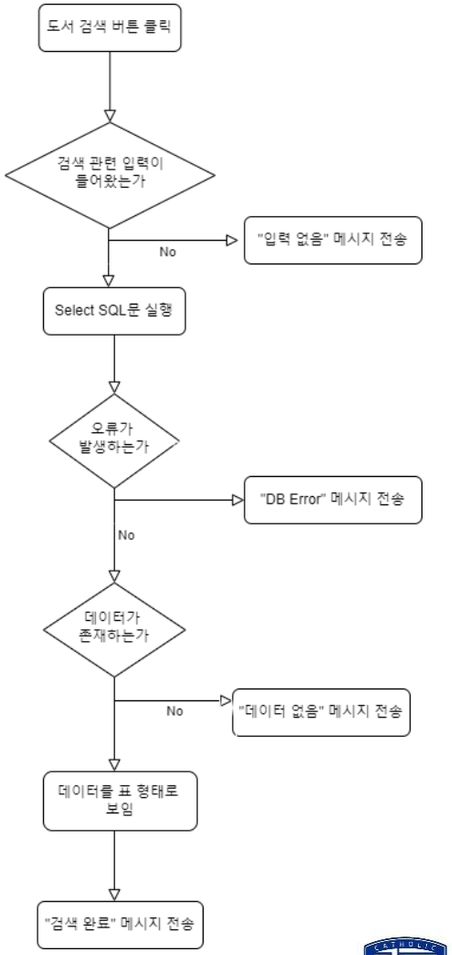
```
    // DB Error
    catch (SQLException e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
        f.error_label.setText("DB Error");
        f.error_label.setForeground(Color.RED);
        return null;
    }
}
```



구현

- 이벤트 처리 (도서 찾기)
  - [검색] 버튼을 눌렀을 때 발생하는 이벤트를 처리하는 코드이다.

```
// (도서 검색 관련) 이벤트 처리
public String[][] func3() {
    // 조건 만족 시 SQL문 실행
    try {
        ResultSet rs;
        String search=search_field.getText();
        // 입력 들어옴
        if (!search.equals("")) {
            rs = t.stmt.executeQuery("select * from book where name like '%"+search+"%' or author like '%"+search+"%';");
            // 입력 초기화
            search_field.setText(null);
            // 데이터 확인
            int rows = 0;
            if (rs.last()) {
                rows = rs.getRow();
                rs.beforeFirst();
            }
            // 데이터 존재
            if(rows>0) {
                // 데이터를 2차원 배열에 담아서 리턴
                String[][] contents=new String[rows][3];
                int idx=0;
                while(rs.next()) {
                    contents[idx][0]=rs.getString("name");
                    contents[idx][1]=rs.getString("author");
                    contents[idx][2]=rs.getString("date");
                    idx++;
                }
                f.error_label.setText("("+search+") 검색 완료!!");
                f.error_label.setForeground(Color.BLUE);
                return contents;
            }
            // 데이터 존재 X
            else {
                // 빈 2차원 배열 리턴
                f.error_label.setText("데이터 없음!");
                f.error_label.setForeground(Color.RED);
                String[][] contents=new String[1][3];
                return contents;
            }
        }
        // 입력 안 들어옴
        else {
            f.error_label.setText("입력 없음!");
            f.error_label.setForeground(Color.RED);
            return null;
        }
    }
    // DB Error
    catch (SQLException e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
        f.error_label.setText("DB Error");
        f.error_label.setForeground(Color.RED);
        return null;
    }
}
```

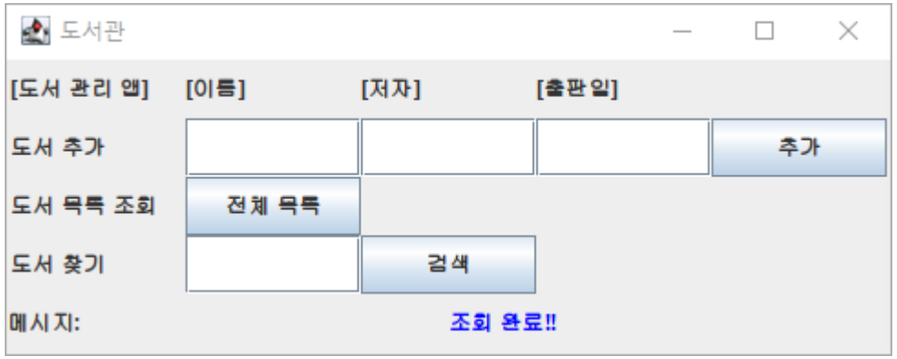


실행정보

- 도서 추가
  - 빈 테이블(book)에 **네 권**의 도서를 추가하였다.

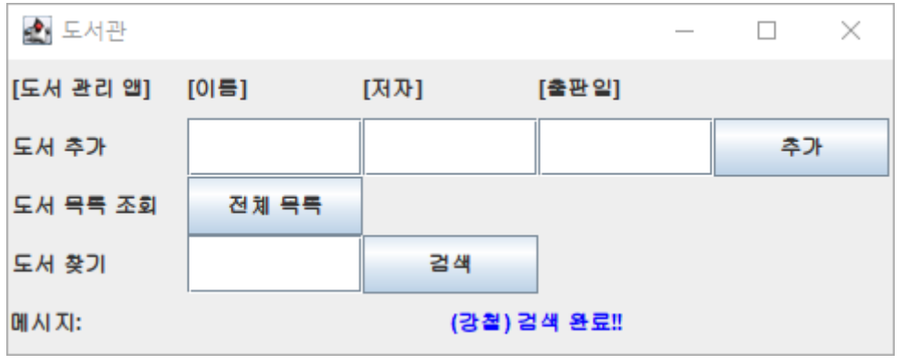
실행정보

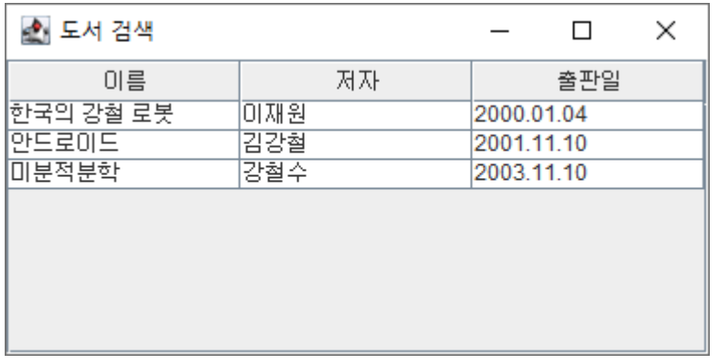
- 전체 조회
  - book 테이블에 저장된 도서 목록을 확인하였다.



실행정보

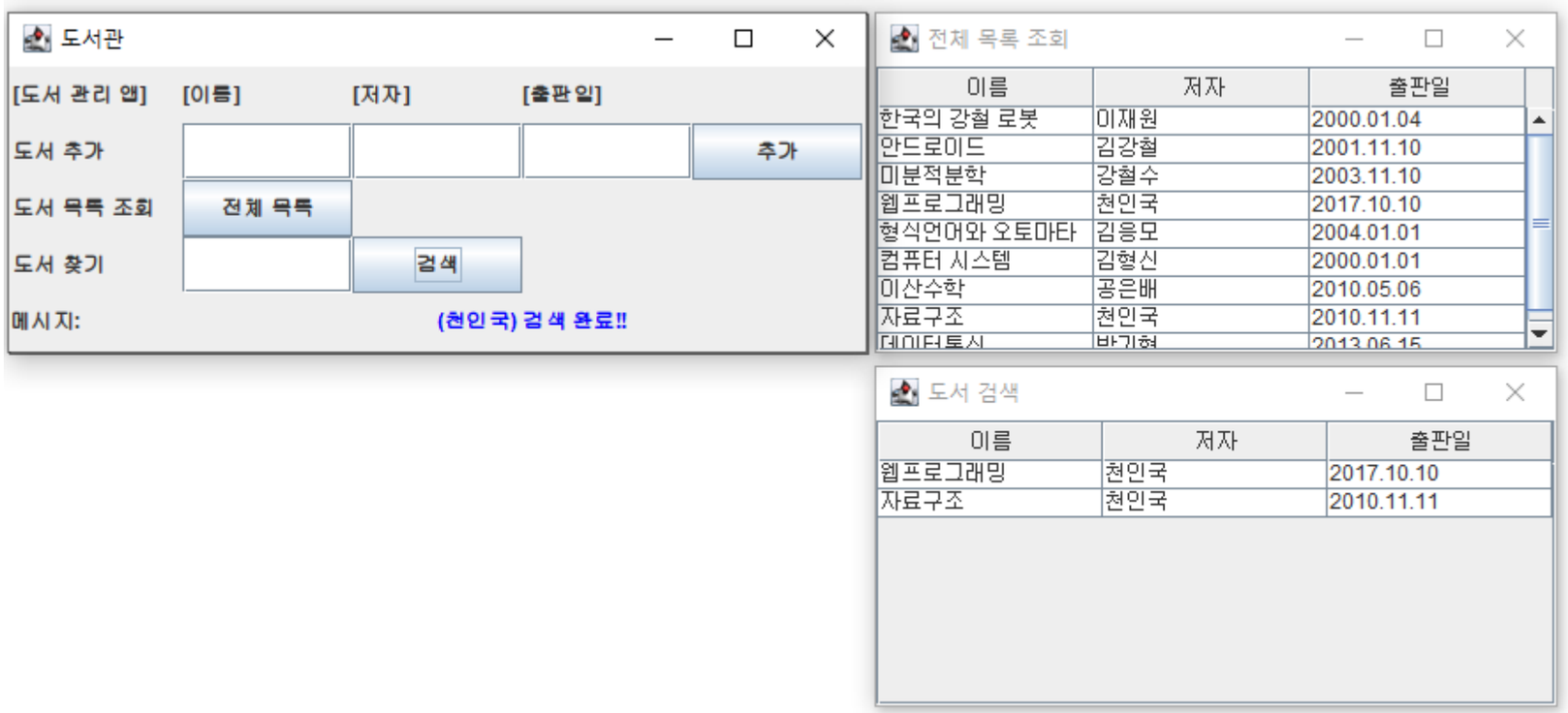
- 도서 찾기
  - “강철” 이라는 키워드로 도서를 검색하였다.
  - 저자가 “김강철”, “강철수” 인 도서를 보여주고, 도서 이름이 “한국의 강철 로봇” 도서를 보여준다.





실행정보

- 전체 구조
  - 도서 목록을 추가한 뒤 전체 구조를 확인해보았다.



컴퓨터정보공학부 2019021725 안성현

감사합니다

<2021/03/04>