

== REPORT ==

< 고속도로 휴게소 통합 리뷰 >

“과제 수행자의 개인정보를 삭제한 후 과제 결과물을 활용하는 것을 허락합니다.”

과 목	데이터베이스
담당 교수	강영명 교수
제출 일자	2021.11.28
전공	컴퓨터공학
팀원	20200001 강요한, 20200837 박종민,
	20180817 김성민, 20180847 윤승한, 20160125 김정우
팀구성	3 조

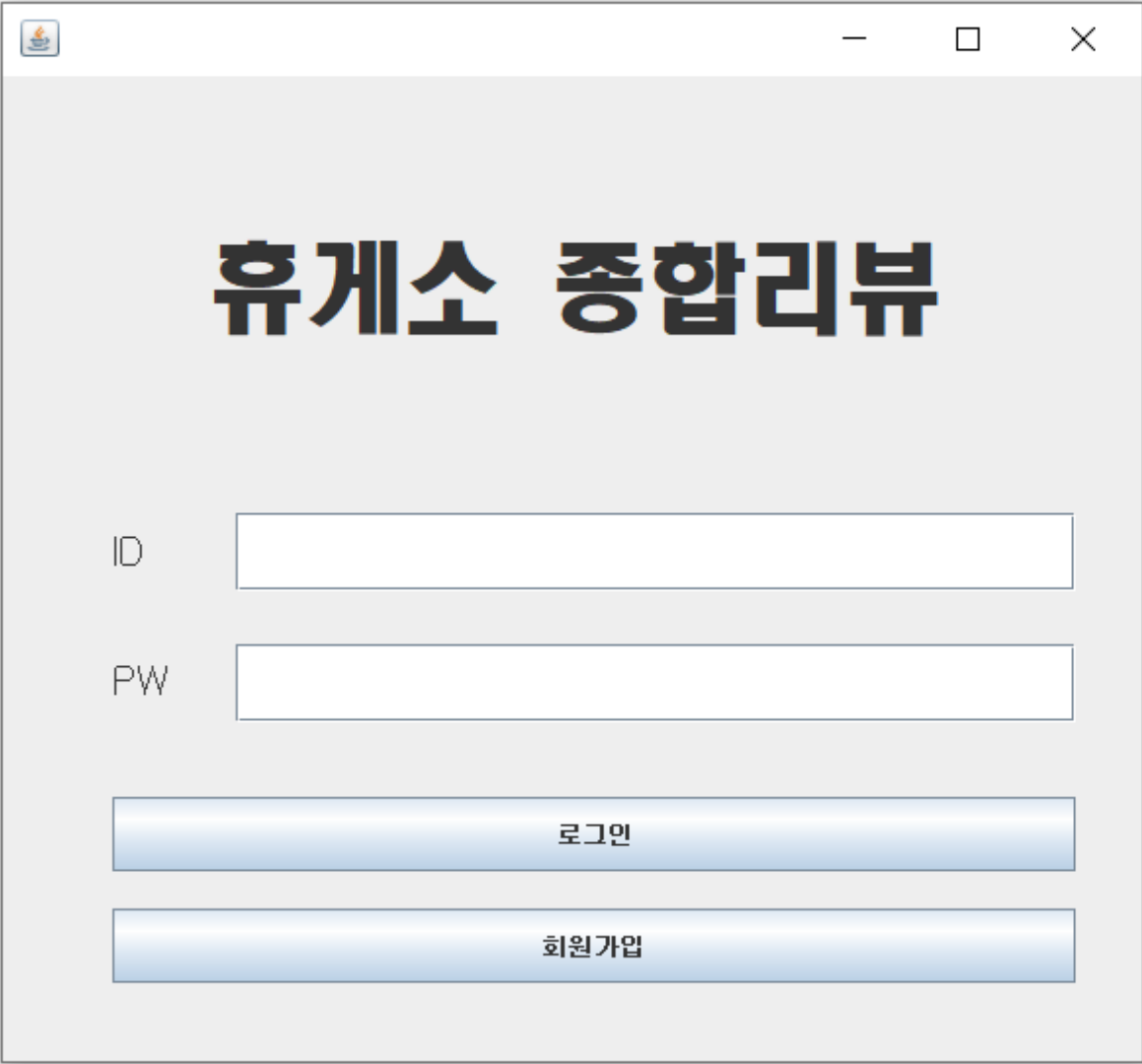
- 목 차 -

I. 주요 화면 캡처	
- 로그인 & 회원가입	
- 휴게소 목록	
- 휴게소 상세정보	
II. 화면 생성에 사용된 SQL문 설명	
- 로그인 & 회원가입	
- 휴게소 목록	
- 휴게소 상세정보	
III. 애플리케이션 특이점	
IV. 보완점	
V. 팀원별 작성 내용	

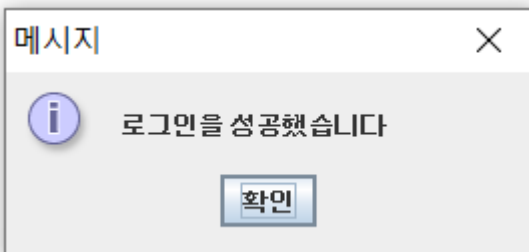
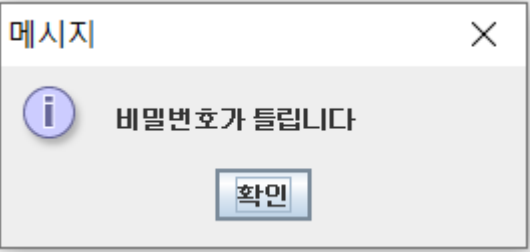
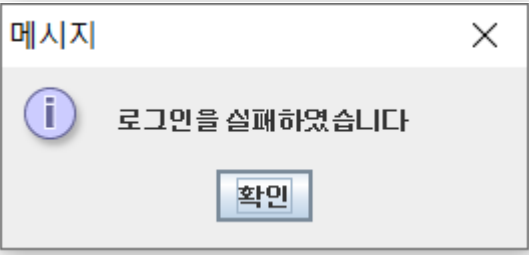
I. 주요 화면 캡처

고속도로 휴게소 통합리뷰 어플은 '로그인&회원가입', '휴게소 목록', '휴게소 상세정보'화면으로 구성되어 있습니다.

1) 로그인 & 회원가입 화면




APP을 구동하게 되면 처음 보이는 화면입니다
로그인, 회원가입 버튼 클릭과 ID와PW를 적고 로그인 할 수 있습니다.

 <p>메시지</p> <p>로그인을 성공했습니다</p> <p>확인</p>	 <p>메시지</p> <p>비밀번호가 틀립니다</p> <p>확인</p>
 <p>메시지</p> <p>로그인을 실패하였습니다</p> <p>확인</p>	

-로그인 성공 메시지 : 로그인 DB에 정상적으로 연동이 되어 회원가입 후 올바른 ID와 PW를 기입한 후 로그인 버튼을 클릭하면 볼 수 있습니다.

-비밀번호 불일치 메시지 : ID에 대한 PW가 일치하지 않은 상태로 로그인 버튼을 클릭한다면 볼 수 있습니다.

-로그인 실패 메시지 : DB연동이 제대로 되지 않은 상태에서 로그인 버튼을 클릭한다면 볼 수 있습니다.



—

□

×

회원가입

아이디 :

비밀번호 :

이름 :

이메일 :

회원가입

취소

로그인 화면에서 회원가입 버튼을 클릭하면 볼 수 있습니다.
ID, PW, 이름, 이메일을 기입하고 회원가입을 할 수 있습니다.
로그인DB와의 연동이 필수입니다.

2) 휴게소 목록

휴게소 종합

휴게소명

휴게소 목록

서울만남(부산 방향)
죽전(서울 방향)
기흥(부산 방향)
안성(서울 방향)
안성(부산 방향)
망향(부산 방향)
옥산(부산 방향)
천안삼거리(서울 방향)
천안호두(부산 방향)
죽암(서울 방향)
죽암(부산 방향)
금강(부산 방향)
옥천(서울 방향)
황간(서울 방향)
황간(부산 방향)
입장거봉포도(서울 방향)
추풍령(서울 방향)
추풍령(부산 방향)
칠곡(서울 방향)
경산(서울 방향)
평사(부산 방향)
칠곡(부산 방향)
건천(서울 방향)
건천(부산 방향)
연양(서울 방향)

로그인에 성공하게 된다면 볼 수 있습니다.
DB에 존재하는 휴게소 목록을 전부 볼 수 있습니다.
검색기능으로 데이터를 검색할 수 있습니다.

휴게소 종합

휴게소명 인천

검색

휴게소 목록

문막(인천 방향)
횡성(인천 방향)

검색기능을 이용하여 "인천"을 검색한 화면입니다.

검색 결과가 존재할 때 휴게소 목록에 나타나게 되고

결과가 존재하지 않으면 목록은 빈 화면만 보이게 됩니다.

3) 휴게소 상세정보

상세정보

서울만남(부산 방향)

대표 음식: 말죽거리소고기국밥

고속도로: 경부선

화물 휴게소: X

전화번호: 02-578-3372

좋아요: 37

싫어요: 16

좋아요

싫어요

휴게소 목록에 있는 휴게소를 클릭하게 되면 보이는 화면입니다.

휴게소의 상세정보를 확인할 수 있습니다.

좋아요 나 싫어요 버튼을 한번 클릭할 때 마다 좋아요, 싫어요 가 1씩 늘어나게 됩니다
실시간으로 DB에 업데이트 되며, 보여지게 됩니다.

II. 화면 생성에 사용된 SQL문 설명

1) 로그인 & 회원가입

```

74 String sql = "SELECT user_PASSWORD FROM userlogin WHERE user_ID = ? ";
75 PreparedStatement pstmt = con.prepareStatement(sql); // pstmt 변수에 sql문 넣기
76 pstmt.setString(1, IdText.getText()); // sql문에서 ?에 해당하는 값에 userID를 집어넣음
77 rs = pstmt.executeQuery(); // sql 수행결과를 rs에 저장
78 if (rs.next()) {
79     // ?로 사용자에게 받은 ID와 비밀번호가 일치하는 경우
80     if (rs.getString(1).contentEquals(PwField.getText())) {
81         // 로그인 성공
82         JOptionPane.showMessageDialog(null, "로그인을 성공했습니다");
83         new Layout3(new RestareaDao().selectName(null));
84         frame.dispose();
85         pstmt.close();
86         con.close();
87     } else {
88         JOptionPane.showMessageDialog(null, "비밀번호가 틀립니다");
89     }
90 }
91 } catch (Exception e1) {
92     e1.printStackTrace();
93     JOptionPane.showMessageDialog(null, "로그인을 실패하였습니다");
94 }
95 }
96
97 });
98

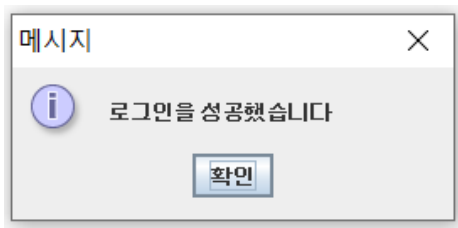
```

```
String sql = "SELECT user_PASSWORD FROM userlogin WHERE user_ID = ? ";
```

user_password(필드) userlogin(테이블)를 통해 조회됩니다. 조건식 WHERE로 uesr_ID= ? 로 넣어 정확한 데이터를 찾습니다.

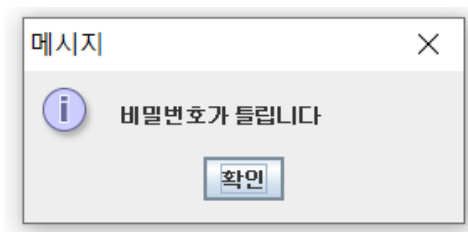
```
PreparedStatement pstmt = con.prepareStatement(sql); // pstmt 변수에 sql문 넣기
```

이 문장을 추가로 설명드리면 preparedStatement는 쿼리문이 실행되기전까지 대기하라는 뜻입니다.



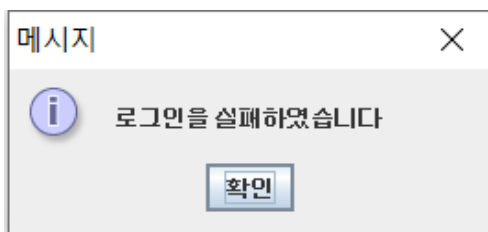
```
if (rs.next()) {
    // ?로 사용자에게 받은 ID와 비밀번호가 일치하는 경우
    if (rs.getString(1).contentEquals(PwField.getText())) {
        // 로그인 성공
        JOptionPane.showMessageDialog(null, "로그인을 성공했습니다");
        new Layout3(new RestareaDao().selectName(null));
        frame.dispose();
        pstmt.close();
        con.close();
    }
}
```

?로 사용자에게 받은 ID와 비밀번호가 일치할 경우 로그인을 성공했습니다. 라는 메시지가 뜬다.



```
} else {
    JOptionPane.showMessageDialog(null, "비밀번호가 틀립니다");
}
```


아닐 경우 비밀번호가 틀렸습니다. 가 화면에 출력됩니다.



```
} catch (Exception e1) {
    e1.printStackTrace();
    JOptionPane.showMessageDialog(null, "로그인을 실패하였습니다");
}
```

PrintStackTrace()는 예외발생시 메서드정보를 화면에 출력합니다.

예외발생시 로그인을 실패하였습니다. 화면에 출력합니다.

— □ ×

회원가입

아이디 :

비밀번호 :

이름 :

이메일 :

회원가입

취소

```
// 회원가입 클릭 시
JoinBtn.addActionListener(new ActionListener() {

    @SuppressWarnings("deprecation")
    @Override
    public void actionPerformed(ActionEvent e) {

        try {
            String sql = "INSERT INTO userlogin VALUES (?, ?, ?, ?)";
            PreparedStatement pstmt = con.prepareStatement(sql);
            pstmt.setString(1, IdText.getText());
            pstmt.setString(2, NameText.getText());
            pstmt.setString(3, PwText.getText());
            pstmt.setString(4, MailText.getText());
            pstmt.executeUpdate();
            JOptionPane.showMessageDialog
            (null, "회원가입이 완료되었습니다");
            frame.dispose();
            pstmt.close();
            con.close();
        } catch (SQLException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
            JOptionPane.showMessageDialog
            (null, "회원가입을 실패하였습니다");
            frame.dispose();
        }
    }
});
```

```
String sql = "INSERT INTO userlogin VALUES (?, ?, ?, ?)";
PreparedStatement pstmt = con.prepareStatement(sql);
pstmt.setString(1, IdText.getText());
pstmt.setString(2, NameText.getText());
pstmt.setString(3, PwText.getText());
pstmt.setString(4, MailText.getText());
pstmt.executeUpdate();
```

Userlogin테이블에 id, 이름, 비밀번호, 메일을 저장합니다

```
pstmt.executeUpdate();
JOptionPane.showMessageDialog
(null, "회원가입이 완료되었습니다");
frame.dispose();
pstmt.close();
con.close();
```

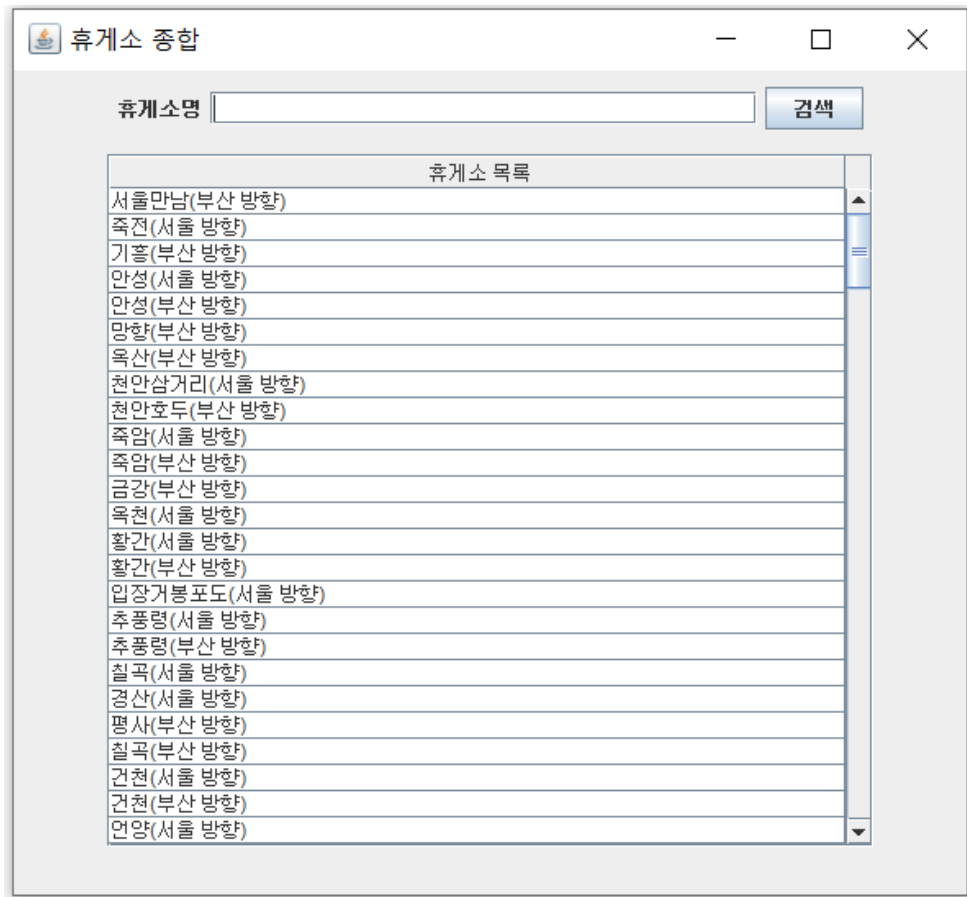
적용이되면 회원가입이 완료되었습니다. 창을 띄웁니다.

```
e1.printStackTrace();
JOptionPane.showMessageDialog
(null, "회원가입을 실패하였습니다");
frame.dispose();
```

예외인 경우 발생시 회원가입을 실패하였습니다. 창을 띄웁니다

```
frame.dispose(); :프로그램이 아닌 현재의 frame을 종료 시킵니다
```

2) 휴게소 목록화면



```

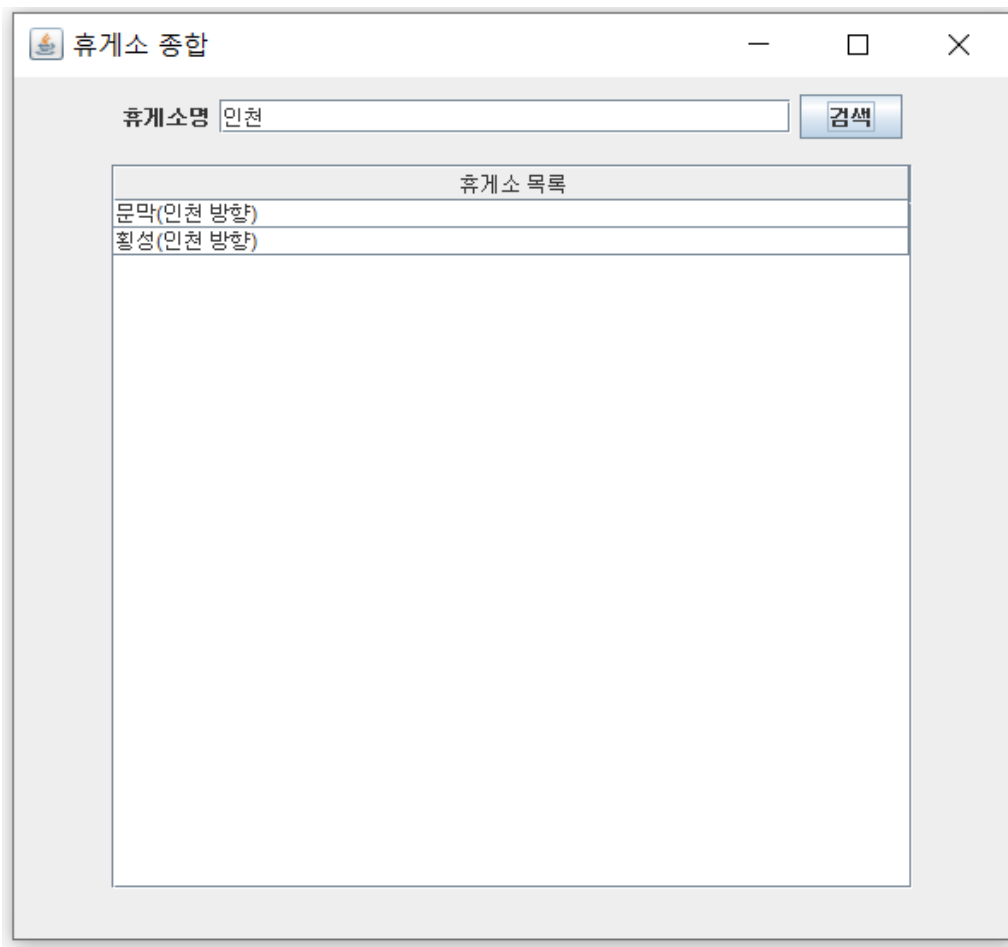
,

private void setTable() { // 테이블 설정
    // TODO Auto-generated method stub
    model = new DefaultTableModel(total, colnames) {
        @Override
        public boolean isCellEditable(int row, int column) {
            return false; // 모든 셀 수정 불가능
        }
    };

    table = new JTable(model);
    // 셀이 한개씩 선택
    table.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
    table.addMouseListener(this); // 마우스리스너 추가
}

public void mouseClicked(MouseEvent me) { // 마우스 클릭 리스너
    int row = table.getSelectedRow(); // 선택된 테이블 열 저장한 변수
    new DetailFlow((String) table.getValueAt(row, 0)); // 상세정보 레이아웃을 여는 코드
}

```



```

public ArrayList<String> selectName(String a) { // 휴게소종합정보 테이블에서 휴게소명 속성 레코드를 가져오는 메소드
    // 레코드를 저장하는 ArrayList배열
    ArrayList<String> ra_arr = new ArrayList<>();

    String[][] total = new String[199][1]; //마지막에 반환할 2차원 배열
    String sql;

    if(a==null) { //값이 null이면 전체 다 갖고오기
        sql = "select * from 휴게소종합정보";
    }
    else { //String 값이 있으면 휴게소명에 'a'가 있는 레코드만 가져오기
        sql="select * from 휴게소종합정보 where 휴게소명 like '%" + a + "%'";
    }


    pstmt = null;
    rs = null;

    try {
        pstmt = con.prepareStatement(sql);
        rs = pstmt.executeQuery();

        while (rs.next()) {
            ra_arr.add(rs.getString("휴게소명")); // 배열에 가져온 레코드 추가
        }
        for (int i = 0; i < ra_arr.size(); i++) {
            total[i][0] = ra_arr.get(i);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return ra_arr;
}

```

3) 휴게소 상세정보

 상세정보

서울만남(부산 방향)

대표 음식 :

말죽거리소고기국밥

고속도로 :

경부선

화물 휴게소 :

X

전화번호 :

02-578-3372

좋아요 : 37

싫어요 : 16

좋아요

싫어요

최종적으로 확인할 휴게소의 상세정보 화면입니다.

표시된 모든 정보들은 구축한 DataBase로부터 불러온 결과 입니다.

맨 하단의 “좋아요”, “싫어요” 버튼을 한번 클릭하면 구축한 데이터베이스에 각각+1씩 Update되고, 상세정보 화면에 실시간으로 나타납니다.

화면에 사용된SQL 문을 아래에서 설명하겠습니다.

1. 휴게소종합정보 테이블의 데이터가 들어간 부분입니다. 각 코드들과 SQL쿼리문을 살펴보겠습니다.

```
private ResultSet loadRestAreaData() {
    try {
        restAreaDataConnection = DriverManager.getConnection(restAreaDataHost, username, pw);
        Statement stmt = restAreaDataConnection.createStatement();
        String query = "SELECT * FROM 휴게소종합정보 WHERE 휴게소명='"+restAreaName+"'";
        ResultSet result = stmt.executeQuery(query);

        return result;
    } catch (SQLException e) {
        e.printStackTrace();
        return null;
    }
}
```

`restAreaDataConnection = DriverManager.getConnection(restAreaDataHost, username, pw);` 부분을 보겠습니다.

`DriverManager`란 미리 로드된 JDBC드라이버를 통해서 `Connection`을 활성화하는 객체입니다.

JDBC란 자바가 제공하는 자바의DB 연결 인터페이스입니다 JDBC4버전 이후로는 `Class.forName`코드를 굳이 작성하지 않아도 자동으로 로드 합니다.

`Connection`이란 데이터 베이스와 연결하는 객체입니다

`DriverManager.getConnection(URL, DB_ID, DB_PW)`으로 Connection 객체를 생성합니다.

URL은 변수 `restAreaDataHost` 로 미리 선언,

DB_ID는 변수 `username`

DB_PW는 변수 `pw` 로 미리 선언하였습니다.

다음은 `Statement stmt = restAreaDataConnection.createStatement();` 부분입니다.-

Statement 는 SQL문을 데이터베이스에 보내기위한 객체입니다.

사용된 SQL문은 다음과 같습니다 `"SELECT * FROM 휴게소종합정보 WHERE 휴게소명='"+restAreaName+"'"` .이 쿼리문을 `query` 라는 변수로 저장했습니다.

이 쿼리문은 휴게소종합정보 테이블의 모든 열중 `휴게소명='"+restAreaName+"'` 조건에 부합하는 데이터를 검색합니다(구축한 DB테이블의 휴게소명과 검색한 휴게소명이 일치되는 조건).

ResultSet 이란 SQL질의에 의해 실행한 결과를 저장하는 객체입니다.

```
private void displayData(ResultSet restAreaData, ResultSet restAreaPreferenceData) {
    try {
        while(restAreaData.next()) {
            RestAreaName.setText(restAreaData.getString("휴게소명"));
            RPfoodName.setText(restAreaData.getString("대표음식"));
            HighwayName.setText(restAreaData.getString("고속도로"));
            TruckStopName.setText(restAreaData.getString("화물 휴게소"));
            CallNum.setText(restAreaData.getString("전화번호"));
        }
    }
}
```

ResultSet 에 저장된 데이터를 2번 이상 불러와야 하므로 while반복문을 사용하여 각각 매칭되는 데이터들을 getString을 사용하여 지정된 위치에 setText했습니다.

2. 휴게소만족도점수 테이블의 데이터가 들어간 부분입니다.

SQL문을 통해 질의하고 데이터를 불러오는 방식은 위와 완전히 같습니다. 그러나 버튼을 누를 때마다 실시간으로 구축한 DB에 값이 Update되는 점이 다릅니다

또한 두 버튼의 동작방식, 쿼리문 또한 변수명을 제외하면 완전히 동일하기 때문에 하나의 예문으로 아래에서 함께 설명하겠습니다.

```
private boolean updateLikeCount(int updateCount) {
    try {
        Statement stmt = restAreaPreferenceConnection.createStatement();
        String query = "UPDATE 휴게소만족도점수 SET user_like = '"+updateCount+"' WHERE 휴게소명='"+restAreaName+"'";
        stmt.executeUpdate(query);

        return true;
    } catch (SQLException e) {
        e.printStackTrace();
        return false;
    }
}
```

```
"UPDATE 휴게소만족도점수 SET user_like = '"+updateCount+"' WHERE 휴게소명='"+restAreaName+"'";
```

휴게소만족도점수 테이블의 `휴게소명='"+restAreaName+"'` 조건에 부합하는 `user_like` 열이

`"'+updateCount+'"` 만큼 플러스(클릭당 +1)되고 Update됩니다.

마찬가지로 2개 이상의 데이터를 불러와야 하므로

```
while(restAreaPreferenceData.next()) {
    LikeCount.setText(String.valueOf(restAreaPreferenceData.getInt("user_like")));
    DislikeCount.setText(String.valueOf(restAreaPreferenceData.getInt("user_dislike")));
}
```

While 문을 사용하여 화면에 setText했습니다.

Ⅲ. 애플리케이션 특이점

1.로그인 기능

- 기존에 찾아본 유사 애플리케이션은 회원가입과 로그인 기능을 제공하지 않았는데 두 기능들을 도입하여 더 폭넓은 기능을 구현하고자 하였습니다.

IV. 보완점

실용되기엔 부족함이 많은 애플리케이션이지만 보완점을 몇 가지 뽑아보자면

1. 좋아요, 싫어요 버튼 중복클릭

- 버튼을 중복해서 계속해서 클릭을 할 수 있는 점은 보완해야할 점이라고 생각합니다. 회원가입과 로그인이 필요한 어플인 만큼 이 부분(중복클릭 불가)을 구현해야 했다고 생각합니다(기술적, 시간적 문제).

2. 좋아요, 싫어요 취소기능

- 한번 버튼을 누르면 다시는 되돌릴 수 없기 때문에 실수로 클릭하는 것을 방지하는 취소기능을 구현해야 했다고 생각합니다. 위에서 말한 중복클릭을 활용해 볼 수 있겠습니다.

3. 리뷰작성 기능 추가

- 좋아요, 싫어요 외에 추가적으로 리뷰를 작성할 수 있게 했다면 더 유익한 휴게소 정보가 될 것 같다고 생각합니다.

4. 회원가입시 제한 없는 아이디와 비밀번호 설정

- 아이디, 비밀번호의 최소한의 형식 지정(특수문자 불가, 4자리 이상 등)이 있었으면 좋았을 것 같다고 생각합니다.

V. 팀원별 작성 내용

20160125김정우

구축한 DB에 기반한 애플리케이션 코딩 중 "상세보기" 화면에 대한 전체적인 코딩을 담당

해당 화면과 구축된 DB연동

데이터 Update 기능 구현

과제 #3의 보고서 필수요건 중

- 애플리케이션의 특이점
- 주요 화면 캡처 및 화면 생성에 사용된 SQL문 설명
 - 최종 화면인 "상세보기"화면에 대한 SQL문 설명

애플리케이션 제공하는 모든 기능(메시지포함)에 대한 화면 캡처와 간단한 설명 작성

팀 회의를 위한 ZOOM회의방 개설.

모든 팀 회의 참여.

201808217 김성민

- 팀원과의 회의를 통해 구현할 어플리케이션 기초 구조 설계
- 팀원과의 회의를 통해 각자 개발할 파트 분배 및 개발 일정 설정
- 평점 기능을 상세히 하기 위해 DB 수정
 - 휴게소만족도점수 테이블의 '유저평점' 속성을 'user_like', 'user_dislike' 두가지 속성으로 수정
- 어플리케이션 중 '휴게소 목록을 테이블로 출력하는 화면'에 대한 코딩 담당
 - 해당 화면과 구축된 DB 연결 구현
 - Sql문을 사용하여 휴게소명을 이용한 검색 기능 구현
- 각자 개발한 파트를 가지고 각 코드 연결하여 하나의 프로젝트로 취합

20180847 윤승한

- 팀원과의 zoom 회의를 통해 '휴게소 검색 창'의 구현을 맡게 됨
- 같은 파트를 맡은 조원과 함께 sql문을 작성하여 휴게소명을 검색하면 검색 내용이 나오도록 검색 기능 구현
- 발표를 위한 ppt 제작을 맡음

20200001 강요한

과제 3을 수행하기위한 자료 수집 및 레이아웃 보조 ,

보고서 작성에 필요한 레이아웃 3과 로그인 화면 회원가입 화면의 SQL 문 작성

20200837 박종민

- 애플리케이션 중 '로그인 & 회원가입'에 대한 코딩 담당
 - 해당 화면과 구축된 DB 연결 구현
 - SQL문을 사용하여 회원가입 기능 구현
 - Sql문을 사용하여 비밀번호를 이용한 로그인 기능 구현
- 각 팀원들의 조사 자료들을 모아서 취합
- 취합한 내용 검토 후 최종 보고서 작성
- 각자 맡은 파트를 서로 점검 후 피드백을 통해 수정
- 팀원과의 회의를 통한 과제 일정 조율 및 역할분담.