

# 역할분담

## 정은애

- 프로젝트 생성
- 메뉴 생성
- 기능
  - 로그인
  - 탈퇴
  - 게임 순위 조회
- Git 관리

## 김형진

- 단어 게임
- 게임 순위 등록

## 신지훈

- 방명록 등록
- 방명록 삭제

## 정은희

- 방명록 조회

## 진영호

- 회원가입



# 미니프로젝트

- 콘솔 프로그램 -

3조



# 목 차

1. 개발도구
2. DB구성도
3. 소스코드
4. 시 연
5. 느 낀 점

# 개발도구



# DB구성도





# 소스코드 및 설명



# 정은애

## 역 할

- 메뉴 구성
- 로그인
- 탈퇴
- 게임 순위 조회

# 메뉴구성

## ●대분류 메뉴

```
// 대분류 메뉴
public int mainMenu() {
    System.out.println(Font.BACKGROUND_WHITE + Font.FONT_GREEN + "    [ 메뉴를 선택해주세요 ]    " + Font.RESET);
    System.out.println(" 1.로그인 | 2.회원가입 | 0.종료");
    int mainCategory = sc.nextInt();
    return mainCategory;
}
```

## ●중분류 메뉴(로그인 성공 시 나옴)

```
// 로그인 성공 시 나오는 중분류 메뉴
public int middleMenu() {
    System.out.println(" _____[ 메뉴를 선택해주세요 ]_____ ");
    System.out.println(" 1.단어맞추기 | 2.랭킹 | 3.방명록 | 8.로그아웃 | 9.탈퇴 | 0.종료");
    System.out.println(" _____ ");
    int middleMenu = sc.nextInt();
    return middleMenu;
}
```



# 메뉴구성

```
// 단어맞추기 중분류 메뉴
public int gameMenu() {
    System.out.println(Font.BACKGROUND_WHITE + Font.FONT_BLUE
        + "                단어맞추기                " + Font.RESET);
    System.out.println(" ————— [ 메뉴를 선택해주세요 ] ————— ");
    System.out.println("1.색상 | 2.동물 | 3.음식 | 8.로그아웃 | 9.이전메뉴 | 0.종료");
    System.out.println(" ————— ");
    int gameMenu = sc.nextInt(); // 메뉴 번호 입력받기
    return gameMenu;
}
```

●중분류에서 [단어 맞추기]를 선택하면 나오는 메뉴

●중분류에서 [랭킹]를 선택하면 나오는 메뉴

```
// 랭킹 중분류 메뉴
public int rankMenu() {
    System.out.println(Font.BACKGROUND_WHITE + Font.FONT_GREEN
        + "                랭킹                " + Font.RESET);
    System.out.println(" — [ 메뉴를 선택해주세요 ] — ");
    System.out.println("      1.목록조회      |      9.이전메뉴      ");
    System.out.println(" ————— ");
    int rankMenu = sc.nextInt();
    return rankMenu;
}
```

# 메뉴구성

// 방명록 중분류 메뉴

```
public int guestBookMenu() {  
    System.out.println(Font.BACKGROUND_WHITE + Font.FONT_GREEN  
        + "                방 명 록                " + Font.RESET);  
    System.out.println(" _____[ 메뉴를 선택해주세요 ]_____");  
    System.out.println("1.목록조회 | 2.등록 | 3.삭제 | 8.로그아웃 | 9.이전메뉴 | 0.종료");  
    System.out.println(" _____");  
    int guestBookMenu = sc.nextInt();  
    return guestBookMenu;  
}
```

●중분류에서 [방명록]을 선택하면 나오는 메뉴

●중분류에서 [회원탈퇴]를 선택하면 나오는 메뉴

// 회원탈퇴 중분류 메뉴

```
public int signOutMenu() {  
    System.out.println(Font.BACKGROUND_WHITE + Font.FONT_GREEN  
        + "                이대로 탈퇴를 이어가시겠습니까?                " + Font.RESET);  
    System.out.println("                1.YES | 2.NO                ");  
    int userCategory = sc.nextInt();  
  
    return userCategory;  
}
```

# 로그인

1)main에서 loginInput 메소드와 login 메소드를 호출

```
case 1: // 로그인 (●eunae)
    System.out.println(Font.BACKGROUND_WHITE + Font.FONT_GREEN + "          [ LOGIN ]          " + Font.RESET);
    UserVO mem = input.loginInput();

    String login_result = user.login(mem.getId(), mem.getPw());

    switch (login_result) {
        case "SIGNOUT_USER": // 회원탈퇴
```

```
InfoInput input = new InfoInput();
MenuPrint menu = new MenuPrint();
User user = new User();
```



2)입력값을 통해 유저 정보 가져옴

```
// 로그인 시 입력받을 값
public UserVO loginInput() {
    User user = new User();

    System.out.println(">> 아 이 디 입력 <<");
    String id = sc.next();
    System.out.println(">> 비밀번호 입력 <<");
    String pw = sc.next();

    return user.userInfo(id, pw);
}
```

3)UserVO 클래스에 쿼리 결과값을 담는다

```
// 로그인 후 유저 정보 가져오기
public UserVO userInfo(String id, String pw) {
    UserVO nvo = new UserVO();
    try {
        conn = DriverManager.getConnection(DBConnection.JDB
        sql = "SELECT * FROM member WHERE id = ?";
        PreparedStatement pre = conn.prepareStatement(sql);
        pre.setString(1, id);

        rs = pre.executeQuery();

        if(rs.next()) {
            nvo.setId(rs.getString("id"));
            nvo.setPw(pw);
            nvo.setName(rs.getString("name"));
        }
    }
```

# 로그인

```
conn = DriverManager.getConnection(DBConnection.JDBC_URL, DBConne
sql = "SELECT * FROM member WHERE id = ?";
PreparedStatement pre = conn.prepareStatement(sql);
// 파라미터 설정
pre.setString(1, id);

// 쿼리 실행
try (ResultSet resultSet = pre.executeQuery()) {
    if (resultSet.next()) {
        // 탈퇴유저 확인(1:탈퇴회원, 0:탈퇴안한회원)
        String userState = resultSet.getString("state");
        // 사용자가 존재하는 경우 비밀번호 확인
        String storedPassword = resultSet.getString("pw");

        if(userState.equals("1")) {           // 탈퇴 회원일 경우
            return "SIGNOUT_USER";
        } else if(userState.equals("0")) { // 탈퇴 안한 회원일 경우
            if (pw.equals(storedPassword)) {
                // 비밀번호가 일치하는 경우 로그인 성공
                return "SUCCESS";
            } else {
                // 비밀번호가 일치하지 않는 경우
                return "WRONG_PASSWORD";
            }
        }
    }
} else {
    // 사용자가 존재하지 않는 경우
    return "USER_NOT_FOUND";
}
```

4) UserVO에 담긴 값 중 id와 pw를 통해 login 메소드를 실행

4-1) 쿼리문 결과에 따라 반환되는 문자열이 다름

```
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        if(conn != null) {
            try {
                conn.close();
            } catch (SQLException e) {
                System.err.println(e.getMessage());
            }
        }
    }

    return "ERROR";
};
```

# 탈퇴

```
case 9: // 회원탈퇴 (●●●●●●)
    int userMenu = menu.signOutMenu();
    switch (userMenu) {
        case 1:
            String signOut_result = user.login(mem.getId(), input.pwCheck());

            switch (signOut_result) {
                case "WRONG_PASSWORD":
                    System.out.println(Font.BACKGROUND_BLACK + Font.FONT_RED + "비밀번호가 일치하지 않음");
                    break;
                case "SUCCESS":
                    boolean randomIntCheck = input.randomIntCheck();
                    // 입력값과 난수가 동일하지 않으면 true
                    if (randomIntCheck) {
                        boolean randomIntDoubleCheck = input.randomIntDoubleCheck();
                        if (randomIntDoubleCheck) {
                            System.out.println(Font.BACKGROUND_BLACK + Font.FONT_RED + "틀렸습니다!");
                            break;
                        } else {
                            int result = user.signOut(mem.getId());
                        }
                    }
            }
        }
    }
```

- 1)main에서 signOutMenu 메소드  
 , pwCheck 메소드  
 , login 메소드  
 , signOut 메소드를 차례대로 호출함

2)비밀번호를 입력받음  
입력받은 문자열을 login 메소드에 활용함.

```
// 회원탈퇴 시 유저의 비번을 입력받음
public String pwCheck() {
    System.out.println("비밀번호를 입력해주세요.");
    String pwCheck = sc.next();
    return pwCheck;
}
```

# 탈퇴

```
// 회원탈퇴 시 랜덤숫자를 받아 정상 탈퇴를 진행함
```

```
// 랜덤 1차 입력
```

```
public boolean randomIntCheck() {
```

```
    boolean result = false;
```

```
    int random = (int) (Math.random() * 555) + 1;
```

```
    System.out.println("탈퇴 진행을 위해 번호를 입력해주세요.");
```

```
    System.out.println("입력하기 >> [ " + random + " ]");
```

```
    int randomNumberCheck = sc.nextInt();
```

```
    if (randomNumberCheck != random) {
```

```
        result = true;
```

```
    }
```

```
    return result;
```

```
}
```

3)난수 입력을 통해 탈퇴를 진행함.  
입력 기회는 단 2번.

```
// 랜덤 숫자 2차 입력
```

```
public boolean randomIntDoubleCheck() {
```

```
    boolean result = false;
```

```
    int randomDouble = (int) (Math.random() * 555) + 1;
```

```
    System.out.println(Font.BACKGROUND_BLACK + Font.FONT_RED + "틀렸습니다!  
+ Font.RESET);
```

```
    System.out.println("입력하기 >> [ " + randomDouble + " ]");
```

```
    int randomNumberDoubleCheck = sc.nextInt();
```

```
    if(randomNumberDoubleCheck != randomDouble) {
```

```
        result = true;
```

```
    }
```

```
    return result;
```

```
}
```



# 게임 순위 조회

```
sql = "SELECT CASE "  
+ "    WHEN state = 1 THEN '탈퇴회원' "  
+ "    ELSE r.id "  
+ "    END AS \"id\" "  
+ "    , score "  
+ "    , DATE_FORMAT(in_date, '%Y-%m-%d') AS \"in_date\" "  
+ " FROM `rank` r "  
+ " JOIN member m "  
+ "    ON r.id = m.id "  
+ " ORDER BY score DESC "  
+ " LIMIT 0,10";
```

↳ 결과를 10개까지만 출력

쿼리문 결과

♡랭크는 10등까지만 보입니다♡

[순위]	[id]	[점수]	[등록일]
1	id	20점	2024-01-24
2	abc	20점	2024-02-02
3	abc	20점	2024-02-02
4	id	20점	2024-02-06
5	id	18점	2024-02-05
6	id	17점	2024-02-06
7	탈퇴회원	13점	2024-01-31
8	abc	11점	2024-02-02

# 게임 순위 조회

게임 점수 순위를 출력하는 로직

```
System.out.println("♡랭크는 10등까지만 보입니다♡");
System.out.println("[순위]\t[id]\t[점수] \t [등록일]");
int rank = 0;
while(rs.next()) {
    String myId = Font.FONT_GREEN + rs.getString("id") + Font.RESET + "\t\t ";
    rank++;
    if(rs.getString("id").equals(id)) {
        System.out.println(rank + "\t" + myId +
                           rs.getString("score") + "점\t" + rs.getString("in_date"));
    } else {
        System.out.println(rank + "\t" + rs.getString("id") + "\t" +
                           rs.getString("score") + "점\t" + rs.getString("in_date"));
    }
}
```



# 김 형 진

## 역 할

- 단어 게임
- 게임 점수 등록

# 단어 게임

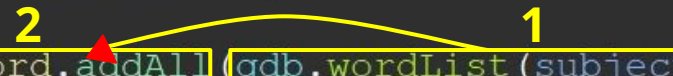
1. Quiz 클래스의 game메서드를 (주제, 멤버ID, 멤버name)을 매개변수로 호출

```
case 1: // 색상 (●hyoungjin)
    quiz.game("색상", mem.getId(), mem.getName());
    break;
case 2: // 동물 (●hyoungjin)
    quiz.game("동물", mem.getId(), mem.getName());
    break;
case 3: // 음식 (●hyoungjin)
    quiz.game("음식", mem.getId(), mem.getName());
    break;
```

2. 매개변수로 받아온 주제를 QuizDB의 wordList함수에 “주제”를 매개변수로 호출 후 word에 리턴값을 add

```
public void game(String subject, String mem_id, String mem_name) {
    List<String> word = new ArrayList<String>();
    List<String> hint = new ArrayList<String>();
    String answer;
    int index = 0;

    word.addAll(qdb.wordList(subject));
}
```



A yellow curved arrow originates from the parameter 'subject' in the 'qdb.wordList(subject)' call (labeled with a yellow '1') and points to the 'word' parameter in the 'word.addAll' call (labeled with a yellow '2').

# 단어 게임

## 1. sql 구문에 매개변수를 넣고 word를 셔플하여 리턴

```
public List<String> wordList(String subject) {  
    List<String> word = new ArrayList<String>();  
    try {  
        con = DriverManager.getConnection(DBConnection.JDBC_URL, DBConnection.USERNAME, DBConnection.PASSWORD);  
        pstmt = con.prepareStatement("select * from word where `subject`='" + subject + "'");  
        rs = pstmt.executeQuery();  
        while (rs.next()) {  
            word.add(rs.getString("word"));  
        }  
        Collections.shuffle(word);  
        return word;  
    }  
}
```

1. The 'subject' parameter is passed to the SQL query.

2. The 'word' list is populated with words from the database.

3. The 'word' list is shuffled.

```
word.addAll(qdb.wordList(subject));  
  
aa: for (int i = 0; i < 5; i++) {  
    hint = qdb.hintList(word.get(i));  
    System.out.println(i + 1 + "번째 문제");  
  
    for (int j = 0; j < hint.size(); j++) {  
        System.out.println(j + 1 + "번째 힌트 " + hint.get(j));  
        System.out.print("정답 : ");  
  
        answer = sc.next();  
        index = qdb.wordGame(answer, hint.get(j), j);  
    }  
}
```

## 2. 해당 word 값에 맞는 힌트 출력하고 답을 체크하는 wordGame 함수 호출

# 단어 게임

## 1. 매개변수로 검색되는 값으로 if구문 실행

```
public int wordGame(String answer, String hint, int j) {  
    try {  
        con = DriverManager.getConnection(DBConnection.JDBC_URL, DBConnection.USERNAME,  
        pstmt = con.prepareStatement(  
            "select * from word_hint where word='" + answer + "' and hint='" + hint + "'");  
        rs = pstmt.executeQuery();  
  
        if(rs.next()) {  
            System.out.println("정답");  
            return 1;  
        }  
        else if(j < 2){  
            System.out.println("다음힌트");  
        }  
        else {  
            System.out.println("맞추지 못했습니다 다음문제");  
        }  
        return 0;  
    }  
}
```

# 게임 점수 등록

```
index = qdb.wordGame(answer, hint.get(j), j);
```

```
    if (index == 1) {  
        score(j);  
        continue aa;  
    }
```

```
public void score(int j) {
```

```
    int score1 = 5;  
    int score2 = 3;  
    int score3 = 1;
```

```
    switch(j) {
```

```
    case 0:
```

```
        score += score1;  
        break;
```

```
    case 1:
```

```
        score += score2;  
        break;
```

```
    case 2:
```

```
        score += score3;  
        break;
```

```
}
```

1.리턴받은 index 값이 1이면 score 함수 실행하고 i값을 continue

```
quiz.game("색상", mem.getId(), mem.getName());
```

```
System.out.println(mem_name+"님의 점수는 " + score + "점 입니다.");
```

```
if(score <= 0){
```

```
    System.out.println("0점은 랭킹에 등록되지 않습니다.");
```

```
}
```

```
else {
```

```
    qdb.rankInsert(score, subject, mem_id);
```

```
}
```

```
score = 0;
```

```
public void rankInsert(int score, String subject, String mem_id) {
```

```
    try {
```

```
        con = DriverManager.getConnection(DBConnection.JDBC_URL, DBConnection.USERNAME, DBConnec
```

```
        pstmt = con.prepareStatement
```

```
        ("insert into `rank` (score,id,in_date) values (" + score + "," + mem_id + ",now())");
```

```
        pstmt.executeUpdate();
```

2.처음 game 함수를 호출 할때 받은 매개변수로 rankInsert함수 실행하여 DB에 점수 등록

# 신 지 훈

## 역 할

- 방명록 등록
- 방명록 삭제

# 방명록 등록

```
public int guestBookInsert(String id, String title, String content) {  
    int result = 0;  
  
    try (Connection conn = DriverManager.getConnection(DBConnection.JDBC_URL,  
                                                         DBConnection.USERNAME,  
                                                         DBConnection.PASSWORD);  
         PreparedStatement pre = conn.prepareStatement(  
             "INSERT INTO guest_book(id, title, content, write_date) "  
             + "VALUES (?, ?, ?, NOW())")) {
```

1. DB 연결
2. 방명록 테이블에 글 등록 쿼리 작성

3. 생성해둔 객체를 사용해 동적인 값을 바인딩
4. 콘솔을 통해 입력한 값을 방명록 테이블에 등록

```
pre.setString(1, id);  
pre.setString(2, title);  
pre.setString(3, content);  
  
result = pre.executeUpdate();
```

5. return 값을 통해 등록성공하면 1, 실패하면 0을 반환

```
    } catch (SQLException e) {  
        e.printStackTrace();  
    }  
  
    // 결과 반환  
    return result > 0 ? 1 : 0;  
}
```



# 방명록 삭제 목록

1. 방명록 등록 때와 같이 DB에 연결
2. 방명록 테이블에서 사용자가 작성한 글을 조회하는 쿼리 작성

```
public void deleteGuestBookList(String id) {  
    try (Connection conn = DriverManager.getConnection(DBConnection.JDBC_URL,  
                                                         DBConnection.USERNAME,  
                                                         DBConnection.PASSWORD);  
         PreparedStatement pre = conn.prepareStatement("SELECT guest_no, "  
                                                         + "title, content, DATE_FORMAT(write_date, '%Y-%m-%d') "  
                                                         + "AS write_date FROM guest_book WHERE id = ?")) {
```

3. SQL쿼리를 실행하고 rs(ResultSet)에 저장
4. if문과 do-while 루프를 통해 rs(ResultSet)에 저장된 정보를 출력

```
pre.setString(1, id);  
ResultSet rs = pre.executeQuery();  
  
if (rs.next()) {  
    System.out.println("[글번호]\t[제목]\t[내용]\t[등록일]");  
    do {  
        System.out.println(rs.getString("guest_no") + "\t"  
                             + rs.getString("title") + "\t"  
                             + rs.getString("content") + "\t"  
                             + rs.getString("write_date"));  
    } while (rs.next());  
} else {  
    System.out.println("작성한 글이 없습니다.");  
}
```



# 방명록 삭제

```
public int guestBookDelete(String id, int guestNo) {  
    int result = 0;  
    try {  
        conn = DriverManager.getConnection(DBConnection.JDBC_URL, DBConnection.USERNAME, DBConnection.PASSWORD);  
        sql = "DELETE FROM guest_book WHERE guest_no = ? AND id = ?";  
  
        pre = conn.prepareStatement(sql);  
        pre.setInt(1, guestNo);  
        pre.setString(2, id);  
  
        result = pre.executeUpdate();  
    } catch (SQLException e) {  
        e.printStackTrace();  
    }  
  
    if(result > 0) {  
        return 1;  
    } else {  
        return 0;  
    }  
}
```

1. 로그인한 유저가 작성한 방명록중 삭제할 글 번호를 물어봅니다.
2. 입력받은 값(글번호)을 통해 글을 삭제 진행

# 방명록 등록 및 삭제 메소드 호출

```
case 2: // 글 등록 (*jihun)
    System.out.println(mem.getName() + "님, 게시글 등록이 시작됩니다.");
    System.out.println("제목을 입력해주세요.");
    String title = sc.next();
    System.out.println("내용을 입력해주세요.");
    String content = sc.next();

    int result = book.guestBookInsert(mem.getId(), title, content);
    if(result > 0) {
        System.out.println(Font.FONT_GREEN + "방명록 등록이 완료되었습니다!" + Font.RESET);
    } else {
        System.err.println("방명록 등록이 정상적으로 진행되지 않았습니다!");
    }
    break;
```

```
case 3: // 글 삭제 (*jihun)
    book.deleteGuestBookList(mem.getId());
    System.out.println("어떤 글을 삭제하겠습니까?");
    int guestNo = sc.nextInt();
    int deleteResult = book.guestBookDelete(mem.getId(), guestNo);
    if(deleteResult > 0) {
        System.out.println(Font.FONT_GREEN + "===== [ 정상적으로 글이 삭제되었습니다. ] =====" + Font.RESET);
    } else {
        System.err.println("글번호가 올바르지 않습니다!");
    }
    break;
```

# 정민희

## 역 할

- 방명록 조회

# 방명록 조회

SQL 연결 시켜주는 클래스

```
public class GuestBook {  
    private Connection conn;  
    private PreparedStatement pr;  
    private ResultSet rs;
```

DB 연결

```
conn = DriverManager.getConnection(DBConnection.JDBC_URL, DBConnection.USERNAME, DBConnection.PASSWORD);
```

SQL 쿼리문

```
pr = conn.prepareStatement("SELECT id"  
                             + ", title"  
                             + ", content"  
                             + ", DATE_FORMAT(write_date, '%Y-%m-%d') AS \"write_date\" "  
                             + "FROM guest_book "  
                             + "ORDER BY write_date DESC");  
rs = pr.executeQuery(); // rs(결과를 담는 바구니)
```

# 방명록 조회

쿼리 결과를 콘솔로 출력

```
while (rs.next()) {  
    System.out.println((num++) + "\t" + rs.getString("id") + "\t"  
        + rs.getString("title") + "\t"  
        + rs.getString("content") + "\t"  
        + rs.getString("write_date"));  
}  
System.out.println();
```

실행 메소드

```
case 1: // 목록조회 (●eunhee)  
    book.listView();  
    break;
```

메소드가 담겨있는 클래스 선언

```
InfoInput input = new InfoInput();  
MenuPrint menu = new MenuPrint();  
User user = new User(); // 로그인, 회  
Quiz quiz = new Quiz(); // 단어맞추기  
Rank rank = new Rank(); // 게임 순위  
GuestBook book = new GuestBook();
```

# 진영호

## 역 할

- 회원가입

# 회원가입

```
// 대분류 메뉴
public int mainMenu() {
    System.out.println(Font.BACKGROUND_WHITE + Font.FONT_GREEN + "    [ 메뉴를 선택해주세요 ]    " + Font.RESET);
    System.out.println(" 1.로그인 | 2.회원가입 | 0.종료");
    int mainCategory = sc.nextInt();
    return mainCategory;
}
```

●대분류에서 회원가입을 선택하면 케이스 2번 으로  
진행됩니다

```
case 2: // 회원가입 (●sdqwe12)
    input.signUpInput();
    break;
```

# 회원가입

```
case 2: // 회원가입 (●sdqwe12)
    input.signUpInput();
    break;
```

케이스 2번에서 signUpInput(); 메소드로  
진행합니다

순서대로 id, pw 그리고 이름을 입력 받습니다

id를 입력 받을 때 불린과 idcheck();를 사용해서  
아이디 중복체크를 해줍니다

```
// 회원가입 시 입력받을 값
public void signUpInput() {
    User user = new User();
    String sign_id = "";
    boolean dupl = true;
    while (dupl) {
        System.out.println("회원가입 진행");
        System.out.println(">> 아이디 입력 <<");
        sign_id = sc.nextLine();
        dupl = user.idCheck(sign_id);
    }
    // System.out.println(">> 비밀번호 입력 <<");
    // String sign_pw = sc.nextLine();
    String sign_pw = pwConfirm();


    System.out.println(">> 이름 입력 <<");
    String sign_name = sc.nextLine();
    user.signUp(sign_id, sign_pw, sign_name);
}
```



# 회원가입

```
// 아이디 중복 체크 sdqwe12
public boolean idCheck(String id) {
    try {
        conn = DriverManager.getConnection(DBConnection.JDBC_URL, DBConnection.USERNAME, DBConnection.PASSWORD);
        PreparedStatement pstmt = conn.prepareStatement("select id from member where id='" + id + "'");
        rs = pstmt.executeQuery();
        while (rs.next()) {
            System.out.println("중복된 아이디입니다. 다시 입력해주세요");
            return true;
        }

    } catch (SQLException e) {
        e.printStackTrace();
    }
    System.out.println("사용 가능한 아이디입니다.");
    return false;
}
```



메소드가 진행되어 db의 member 테이블에서 중복을 검색하고, 값이 없다면 false를 반환합니다

# 회원가입

```
String sign_pw = pwConfirm();
```

```
// 비밀번호 정규식 컨펌
private String pwConfirm() {
    String pwRegex = "^(?=.*[a-z])(?=.*[A-Z])(?=.*\\d)(?=.*[@$!%*?&])[A-Za-z\\d@$!%*?&]{4,}$";

    while (true) {
        System.out.println(">> 비밀번호 입력 (최소 4자 이상, 대소문자, 숫자, 특수문자 필수) <<");
        String sign_pw = sc.nextLine();

        // 정규식 패턴과 일치하는지 확인
        if (sign_pw.matches(pwRegex)) {
            return sign_pw; // 유효한 비밀번호 반환
        } else {
            System.out.println("비밀번호가 조건에 맞지 않습니다. 다시 입력하세요.");
        }
    }
}
```

Pwconfirm(); 메소드로  
진행합니다

pwRegex = 조건이 되는 정규식

비밀번호의 조건인 최소 4자 이상  
대소문자, 숫자, 특수문자를  
출력해줍니다

입력 받은 비밀번호가 정규식과  
일치하면 비밀번호를  
반환해줍니다

# 회원가입

```
// 회원가입 sdqwe12
public void signUp(String id, String pw, String name) {
    try {
        conn = DriverManager.getConnection(DBConnection.JDBC_URL, DBConnection.USERNAME, DBConnection.PASSWORD);
        sql = "insert into member (id,pw,name,state,sign_date) values (?,?,,0,now())";
        PreparedStatement pre = conn.prepareStatement(sql);

        pre.setString(1, id);
        pre.setString(2, pw);
        pre.setString(3, name);
        pre.executeUpdate();

    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

Id와 pw를 입력하고 마지막으로 name까지 입력 받으면 db에 데이터를 삽입하고 회원가입이 마무리 됩니다

감사합니다 ㄹ•ㄹ•?



# 프로그램 시 연



# 느 낄 점

**정 은 애**

메소드 활용에 대해 많은걸 알았고, GitHub와 친해지는 시간이 되었습니다.

**김 형 진**

프로젝트를 하면서 코드 이해도가 상승하여 코드를 간결하게 쓸 수 있게 되었습니다.

**신 지 훈**

자바 코딩을 배우고 나서 처음으로 지금껏 배워왔던 것들이 활용되는 모습을 보며 코딩이라는 것이 멋있다고 느꼈습니다.

**정 은 희**

이해가 어려운 부분이 많았지만, 배우는 과정을 통해 더 노력해야겠다는 생각을 했습니다.

**진 영 호**

처음으로 하는 팀 프로젝트라서 어려움도 있었지만, 팀원들 덕분에 많이 배웠습니다.



**감사합니다.**

**3조**

