

Hello,

KDT 웹 개발자 양성 프로젝트

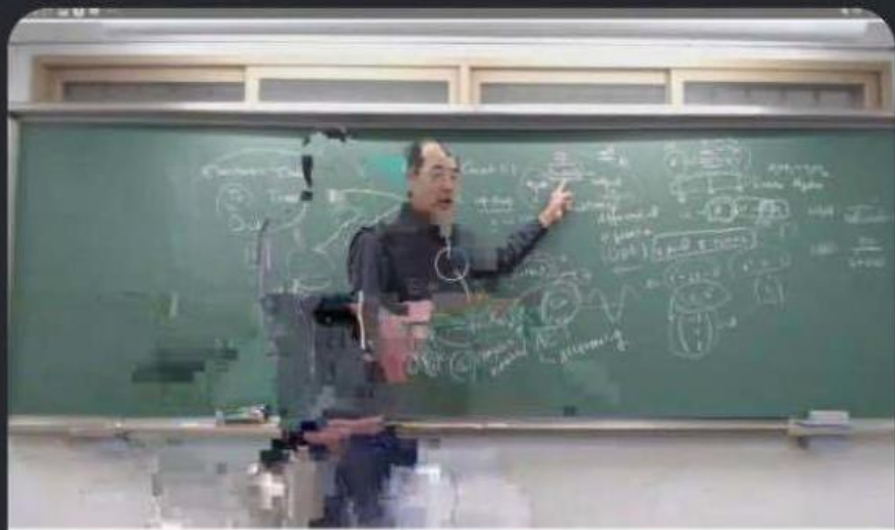
5기!

with





눈빛 애교 어피치



37

교수님 진도가 너무 빨라서 잔상밖에 보이지 않습니다

37

00:09



my computer

DAO, DTO, VO 개념 및 차이



수학·통계학



컴퓨터 공학



머신러닝(ML)



딥러닝(DL)



etc.







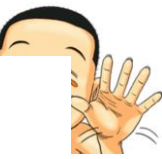
Java



JDBC



RDBMS





간단한

회원 관리 프로그램

원하는 작업 번호를 입력하세요:

[illegible]



회원용 테이블 생성

및 데이터 삽입


```
use jdbc_ex;
```

```
CREATE TABLE user_table
```

```
(  
    id            INT AUTO_INCREMENT PRIMARY KEY,  
    userid        VARCHAR(50)  NOT NULL UNIQUE,  
    name          VARCHAR(100) NOT NULL,  
    password      VARCHAR(255) NOT NULL,  
    age           INT           NOT NULL,  
    membership    BOOLEAN      NOT NULL DEFAULT 0,  
    signup_date   TIMESTAMP     DEFAULT CURRENT_TIMESTAMP  
);
```

```
INSERT INTO user_table (userid, name, password, age, membership)  
VALUES ('xensign', '이효석', '1234', 40, 1);
```

```
INSERT INTO user_table (userid, name, password, age, membership)  
VALUES ('kimsiwan', '김시완', '1234', 30, 1);
```

```
INSERT INTO user_table (userid, name, password, age, membership)  
VALUES ('imsiwan', '임시완', '1234', 38, 0);
```

```
INSERT INTO user_table (userid, name, password, age, membership)  
VALUES ('pororo', '뽀로로', '1234', 10, 0);
```

```
INSERT INTO user_table (userid, name, password, age, membership)  
VALUES ('luppy', '김루피', '1234', 9, 1);
```

```
SELECT *  
FROM user_table;
```

회원 관리 테이블 생성

운영을 위한 더미 데이터 삽입

	id	userid	name	password	age	membership	signup_date
1	1	xenosign	이효석	1234	40	1	2024-07-16 09:06:20
2	2	kimsiwan	김시완	1234	30	1	2024-07-16 09:06:20
3	3	imsiwan	임시완	1234	38	0	2024-07-16 09:06:20
4	4	pororo	뽀로로	1234	10	0	2024-07-16 09:06:20
5	5	luppy	김루피	1234	9	1	2024-07-16 09:06:20





User 클래스 작성

(VO 패턴)



왜? 클래스를 따로 만들죠!?

```
CREATE TABLE user_table
(
    id            INT AUTO_INCREMENT PRIMARY KEY,
    userid       VARCHAR(50)  NOT NULL UNIQUE,
    name         VARCHAR(100) NOT NULL,
    password     VARCHAR(255) NOT NULL,
    age          INT          NOT NULL,
    membership   BOOLEAN      NOT NULL DEFAULT 0,
    signup_date  TIMESTAMP    DEFAULT CURRENT_TIMESTAMP
);
```

회원 정보는 고정 된,
구조를 가지게 됩니다!

하지만 매번 이러한 데이터를
따로 구조를 만들고 각각 문자열로 처리한다면!?



사실 별로 하는거 없지만



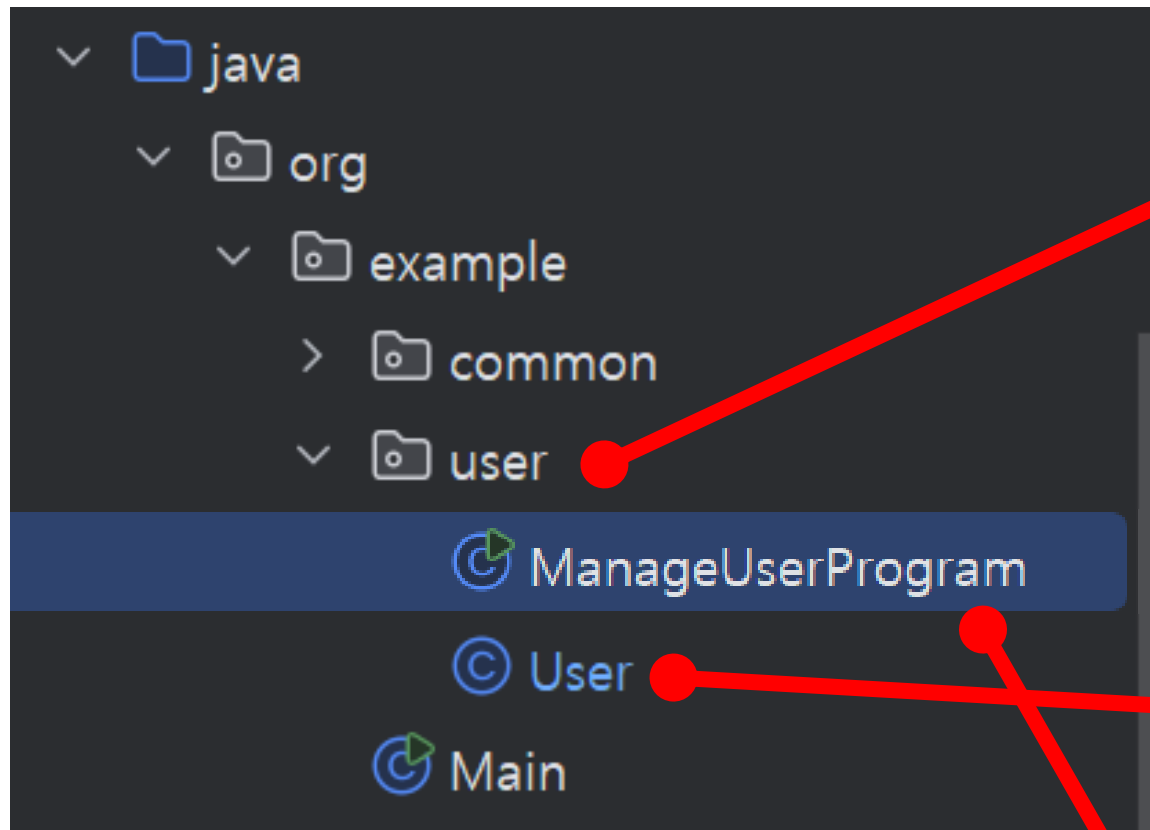
오늘은 더 적극적으로 안할거야



클래스

클래스는 결국
객체를 만드는 설계도입니다!

즉, 회원 정보를 클래스로
관리하면 편리합니다!



user 패키지 생성

User 클래스 생성

실제 회원 관리 프로그램 클래스

```
public class User { 15 usages  👤 kdtTetz
    private int id; 4 usages
    private String userid; 4 usages
    private String name; 4 usages
    private String password; 3 usages
    private int age; 4 usages
    private boolean membership; 4 usages
    private Timestamp signupDate; 4 usages
```

회원 정보를 필드로 가지는
클래스 생성



편리한 객체 생성을 위한
생성자 코드 추가!

```
public User(int id, String userid, String name, String password, int age,
    this.id = id;
    this.userid = userid;
    this.name = name;
    this.password = password;
    this.age = age;
    this.membership = membership;
    this.signupDate = signupDate;
}
```

```
// Getters and setters
```

```
public int getId() { return id; }
```

```
public void setId(int id) { this.id = id; }
```

```
public String getUserId() { return userid; }
```

```
public void setUserId(String userid) { this.userid = userid; }
```

```
public String getName() { return name; }
```

```
public void setName(String name) { this.name = name; }
```

```
public String getPassword() { return password; }
```

```
public void setPassword(String password) { this.password = password; }
```

```
public int getAge() { return age; }
```

Getter & Setter



1/5/0

0/5/1



준비됐어 야? 물론이지 요!!

@Override kdtTetz *

```
public String toString() {  
    return "[" +  
        "id=" + id +  
        ", userid='" + userid + '\'' +  
        ", name='" + name + '\'' +  
        ", age=" + age +  
        ", membership=" + membership +  
        ", signupDate=" + signupDate +  
        "']";  
}
```

원활한 출력을 위한
toString 오버라이드!





User 클래스를 이용한 사용자 추가

```
public class ManageUserProgram { new *
    public static void main(String[] args) { new *
        Scanner scanner = new Scanner(System.in);

        // 회원 정보를 입력 받기
        System.out.print("추가할 회원의 ID: ");
        String newId = scanner.nextLine();
        System.out.print("이름: ");
        String name = scanner.nextLine();
        System.out.print("비밀번호: ");
        String newPassword = scanner.nextLine();
        System.out.print("나이: ");
        int age = scanner.nextInt();
        scanner.nextLine(); // Consume newline
        System.out.print("멤버십 여부 (true/false): ");
        boolean membership = scanner.nextBoolean();
        scanner.nextLine(); // Consume newline
    }
}
```



사용자로부터
회원 정보를 입력 받기

// 입력 받은 정보로 새로운 회원 객체 생성

```
User newUser = new User(id: 0, newId, name, newPassword, age, membership, signupDate: null);
```



// 데이터 베이스 접속

```
Connection conn = JDBCUtil.getConnection();
```

입력 받은 정보를 전달하여
생성자를 통해 새로운 회원 객체 생성

어제 생성한 JDBCUtil 을 활용하여
DB 서버에 접속





불 엔터테인먼트



시금치 빼는ing.

PreparedStatement 개념과

사용법 알고

가실게요~!



Prepared Statement

4:56



쿼리

준비됐어?

<https://youtu.be/GgS78QS-uPg>



```
INSERT INTO user_table (userid, name, password, age, membership)
VALUES ('xenosign', '이호석', '1234', 40, 1);
```

```
INSERT INTO user_table (userid, name, password, age, membership)
VALUES ('imsiwan', '임시완', '1234', 38, 0);
```

```
INSERT INTO user_table (userid, name, password, age, membership)
VALUES ('pororo', '뽀로로', '1234', 10, 0);
```

```
INSERT INTO user_table (userid, name, password, age, membership)
VALUES ('luppy', '김루피', '1234', 9, 1);
```



Java™



하지만 PreparedStatement 를 사용한다면!?



```
// 사용자 추가
public void addUser(User user) { 1 usage   kdtTetz
    Connection conn = JDBCUtil.getConnection();
    String sql = "INSERT INTO user_table (userid, name, password, age, membership) VALUES (?, ?, ?, ?, ?)";

    try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
        pstmt.setString(parameterIndex: 1, user.getUserid());
        pstmt.setString(parameterIndex: 2, user.getName());
        pstmt.setString(parameterIndex: 3, user.getPassword());
        pstmt.setInt(parameterIndex: 4, user.getAge());
        pstmt.setBoolean(parameterIndex: 5, user.isMembership());
        pstmt.executeUpdate();
    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
}
```

ㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋ
ㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋ
ㅋㅋ



// 사용자 추가

```
public void addUser(User user) { 1 usage   👤 kdtTetz
    Connection conn = JDBCUtil.getConnection();
    String sql = "INSERT INTO user_table (userid, name, password, age, membership) VALUES (?, ?, ?, ?, ?)";

    try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
        pstmt.setString(parameterIndex: 1, user.getUserid());
        pstmt.setString(parameterIndex: 2, user.getName());
        pstmt.setString(parameterIndex: 3, user.getPassword());
        pstmt.setInt(parameterIndex: 4, user.getAge());
        pstmt.setBoolean(parameterIndex: 5, user.isMembership());
        pstmt.executeUpdate();
    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
}
```

상황에 따라 다른 값을
넣어야 하는 곳에는
? 를 넣기!

상황에 따라서
? 의 순서에 맞게 값을 대입



그래서 장점이!?

- PreparedStatement 를 쓰면 매번 문자열을 생성할 필요가 없기 때문에 성능 최적화에 좋습니다!
- SQL Injection 이라 불리는, DB 와의 통신 과정에서 SQL 문자열을 넣어서 데이터 베이스의 이상 현상 또는 데이터 변경을 시도하는 공격으로부터 자유로울 수 있습니다!



User 클래스를 이용한 사용자 추가

// 입력 받은 정보로 새로운 회원 객체 생성

```
User newUser = new User(id: 0, newId, name, newPassword, age, membership, signupDate: null);
```



// 데이터 베이스 접속

```
Connection conn = JDBCUtil.getConnection();
```

입력 받은 정보를 전달하여
생성자를 통해 새로운 회원 객체 생성

어제 생성한 JDBCUtil 을 활용하여
DB 서버에 접속

```
// 매번 새로운 회원 데이터를 추가해야 하므로 변경되는 쿼리를 편리하게 작성하기 위한 PreparedStatement
String sql = "INSERT INTO user_table (userid, name, password, age, membership) VALUES (?, ?, ?, ?, ?)";

try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
    pstmt.setString(parameterIndex: 1, newUser.getUserid());
    pstmt.setString(parameterIndex: 2, newUser.getName());
    pstmt.setString(parameterIndex: 3, newUser.getPassword());
    pstmt.setInt(parameterIndex: 4, newUser.getAge());
    pstmt.setBoolean(parameterIndex: 5, newUser.isMembership());
    pstmt.executeUpdate();
} catch (SQLException e) {
    throw new RuntimeException(e);
}

System.out.println("회원이 성공적으로 추가되었습니다.");
```

PreparedStatement 설정

생성된 User 객체로부터
각각의 정보를 가져와서
PreparedStatement 에 삽입

```
// 매번 새로운 회원 데이터를 추가해야 하므로 변경되는 쿼리를 편리하게 작성하기 위한 PreparedStatement
String sql = "INSERT INTO user_table (userid, name, password, age, membership) VALUES (?, ?, ?, ?, ?)";

try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
    pstmt.setString(parameterIndex: 1, newUser.getUserid());
    pstmt.setString(parameterIndex: 2, newUser.getName());
    pstmt.setString(parameterIndex: 3, newUser.getPassword());
    pstmt.setInt(parameterIndex: 4, newUser.getAge());
    pstmt.setBoolean(parameterIndex: 5, newUser.isMembership());
    pstmt.executeUpdate();
} catch (SQLException e) {
    throw new RuntimeException(e);
}

System.out.println("회원이 성공적으로 추가되었습니다.");
```

설정된 SQL 쿼리를 실행!

추가할 회원의 ID: *tetz*
이름: *고테츠*
비밀번호: *1234*
나이: *12*
멤버십 여부 (true/false): *false*
회원이 성공적으로 추가되었습니다.



	id	userid	name	password	age	membership	signup_date
1	1	xenosign	이효석	1234	40	1	2024-07-16 09:06:20
2	2	kimsiwan	김시완	1234	30	1	2024-07-16 09:06:20
3	3	imsiwan	임시완	1234	38	0	2024-07-16 09:06:20
4	4	pororo	뽀로로	1234	10	0	2024-07-16 09:06:20
5	5	luppy	김루피	1234	9	1	2024-07-16 09:06:20
6	6	tetz	고테츠	1234	12	0	2024-07-16 10:23:18





그랑제(사이즈) 팔입니다



O **OBJECT** **O** **ORIENTED** **P** **PROGRAMMING**





DB와 통신하는 기능은 따로 분리하기



DB 통신을 담당하는
ManageUser 클래스 생성

```
▼ java
  ▼ org
    ▼ example
      > common
      ▼ user
        © ManageUser
        © ManageUserProgram
```

```
public class ManageUser { 2 usages kdtTetz *
```

```
// 사용자 추가
```

```
public void addUser(User user) { usage kdtTetz
```

```
    Connection conn = JDBCUtil.getConnection();
```

```
    String sql = "INSERT INTO user_table (userid, name, password, age, membership)
```

```
    try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
```

```
        pstmt.setString(parameterIndex: 1, user.getUserid());
```

```
        pstmt.setString(parameterIndex: 2, user.getName());
```

```
        pstmt.setString(parameterIndex: 3, user.getPassword());
```

```
        pstmt.setInt(parameterIndex: 4, user.getAge());
```

```
        pstmt.setBoolean(parameterIndex: 5, user.isMembership());
```

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

```
    } catch (SQLException e) {
```

```
        throw new RuntimeException(e);
```

```
    }
```

```
}
```

회원 추가 기능을
addUser 라는 메서드로 변경

```
public class ManageUser { 2 usages  kdtTetz *  
    // 사용자 추가  
    public void addUser(User user) {  
        Connection conn = JDBCUtil.getConnection();  
        String sql = "INSERT INTO user_table (userid, name, pass  
  
        try (PreparedStatement pstmt = conn.prepareStatement(sql)  
            pstmt.setString(parameterIndex: 1, user.getUserid());  
            pstmt.setString(parameterIndex: 2, user.getName());  
            pstmt.setString(parameterIndex: 3, user.getPassword());  
            pstmt.setInt(parameterIndex: 4, user.getAge());  
            pstmt.setBoolean(parameterIndex: 5, user.isMembership());  
            pstmt.executeUpdate();  
        } catch (SQLException e) {  
            throw new RuntimeException(e);  
        }  
    }  
}
```

회원 정보는
ManageUserProgram 에서
입력 받으므로

외부에서 User 객체를
매개변수로 전달 받기



분리된 객체를 가져와서 사용하기



```
public class ManageUserProgram { new *
    public static void main(String[] args) { new *
        Scanner scanner = new Scanner(System.in);
        // 이전 코드 내용
        // 입력 받은 정보로 새로운 회원 객체 생성
        User newUser = new User(id: 0, newId, name, newPassword, age, membership, signupD

        // DB 통신을 전담하는 인스턴스의 메서드를 사용하여 OOP 적 구현
        ManageUser manageUser = new ManageUser();
        // 생성한 회원 객체를 전달
        manageUser.addUser(newUser);
```

DB 통신 및 회원 추가
클래스를 인스턴스 화

회원 추가 메서드에
생성한 회원의 객체를 전달!

추가할 회원의 ID: *faker*
이름: *이상혁*
비밀번호: *goat*
나이: *26*
멤버십 여부 (true/false): *true*

1	1	xenosign	이효석	1234	40
2	2	kimsiwan	김시완	1234	30
3	3	imsiwan	임시완	1234	38
4	4	pororo	뿌로로	1234	10
5	5	luppy	김루피	1234	9
6	6	tetz	고테츠	1234	12
7	7	faker	이상혁	goat	26





실습, 특정 회원 삭제하기 기능 구현!

- 방금 작성한 ManageUser 클래스에 deleteUserById(int id) {} 를 추가해 주세요.
- ManageUserProgram 에서 사용자로부터 int 타입의 id 값을 입력 받아서 해당 id 를 가진 사용자를 삭제하는 기능을 추가해 주세요
- (추가) 해당 id 를 가진 사용자가 없을 경우, “ID 가 xx 인 회원이 존재하지 않습니다” 를 출력하세요. 반대로 삭제가 성공했을 경우 “ID 가 xx 인 회원 정보가 성공적으로 제거 되었습니다” 를 출력해 주세요



실습, 특정 회원 삭제하기 기능 구현!

- (추가) 삭제 성공 여부는 삭제 쿼리 실행 시, 몇 개의 데이터가 변경 되었는지를 `affectedRows` 의 수를 체크하면 됩니다!
- 아래의 코드와 실행 결과를 보고 실습을 완성해 주세요

ManageUser 클래스



```
public void deleteUserById(int id) {  
    // 삭제 기능을 구현해 주세요  
}
```

ManageUserProgram

```
// 삭제할 회원의 id 입력 받기  
System.out.print("삭제할 회원의 ID: ");  
int deleteId = scanner.nextInt();  
  
// 기능을 구현한 deleteUserById 를 사용하여 회원 삭제 진행  
manageUser.deleteUserById(deleteId);
```


삭제할 회원의 ID를 입력하세요: 10

ID 가 10 인 회원이 존재하지 않습니다!



삭제할 회원의 ID를 입력하세요: 6

ID 가 6 인 회원 정보가 성공적으로 제거 되었습니다.

1	1	xenosign	이효석	1234
2	2	kimsiwan	김시완	1234
3	3	imsiwan	임시완	1234
4	4	pororo	뽀로로	1234
5	5	luppy	김루피	1234
6	7	faker	이상혁	goat

id 가 6인 회원 정보
삭제 완료!



실제 작동하는

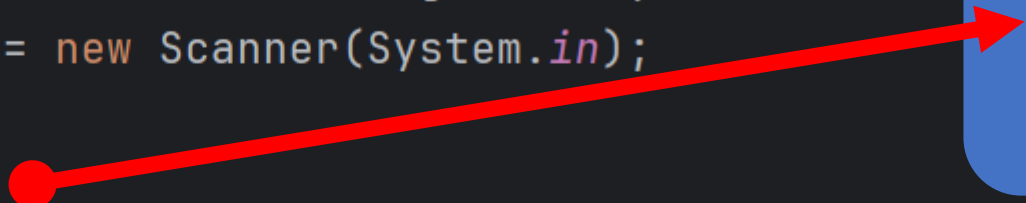
프로그램으로 만듭시다



```
public class ManageUserProgramV4 { new *
    public static void main(String[] args) { new *
        ManageUser manageUser = new ManageUser();
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("==== 회원 관리 프로그램 =====");
            System.out.println("1. 회원 목록 조회");
            System.out.println("2. 회원 추가");
            System.out.println("3. 회원 삭제");
            System.out.print("원하는 작업 번호를 입력하세요: ");

            int choice = scanner.nextInt();
            scanner.nextLine();
        }
    }
}
```



사용자의 입력을 받아서
회원 목록을 조회하고
회원 추가, 삭제를 하는
프로그램으로 구성

사용자의 입력에 따라 각각의 기능을 할 수 있도록 수정

회원 추가

회원 삭제

```
if (choice == 1) {
    manageUser.getAllUsers();
} else if (choice == 2) {
    System.out.print("추가할 회원의 ID: ");
    String newId = scanner.nextLine();
    System.out.print("이름: ");
    String name = scanner.nextLine();
    System.out.print("비밀번호: ");
    String newPassword = scanner.nextLine();
    System.out.print("나이: ");
    int age = scanner.nextInt();
    scanner.nextLine(); // Consume newline
    System.out.print("멤버십 여부 (true/false): ");
    boolean membership = scanner.nextBoolean();
    scanner.nextLine(); // Consume newline

    User newUser = new User(id: 0, newId, name, newPassword, age, membership, signupDate: null);
    manageUser.addUser(newUser);
} else if (choice == 3) {
    System.out.print("삭제할 회원의 ID: ");
    int deleteId = scanner.nextInt();
    manageUser.deleteUserById(deleteId);
} else {
    System.out.println("잘못된 선택입니다. 다시 시도하십시오.");
}
```



전체 회원

조회 기능 구현

ManageUser 클래스



```
public void getAllUsers() {  
    Connection conn = JDBCUtil.getConnection();  
    String sql = "SELECT * FROM user_table";  
}
```

전체 회원 목록 출력을 위한
getAllUsers 메서드 추가

user_table 의 전체 데이터를
불러오는 쿼리 작성

```
try (Statement stmt = conn.createStatement();  
    ResultSet rs = stmt.executeQuery(sql)) {
```

SELECT 쿼리의 결과를 담은
ResultSet 를 이용하여
쿼리의 실행 결과문을 담기!

SELECT 쿼리가 가져다 준
여러 개의 데이터를
테이블 형태의 Set 으로 담은
데이터 구조!

Try-with-resources 를
사용하여 DB 접속 후
접속 종료를 자동으로 수행!

→ 안전하게 처리 가능
→ 리소스를 닫는 처리를 따로 안
해도 되며 예외 처리 간소화



cursor		m_id	m_pw	m_name	m_email
rs.next() : true		id001	pw001	홍길동	test01
rs.next() : true		id002	pw002	김철수	test02
rs.next() : true		id003	pw003	김영희	test03
rs.next() : false					



```
try (Statement stmt = conn.createStatement();  
    ResultSet rs = stmt.executeQuery(sql)) {  
    List<User> users = new ArrayList<>();
```

몇 개의 데이터가 들어올지 모르므로
데이터 추가가 용이한
ArrayList 컬렉션 사용!

```
while (rs.next()) {  
    User user = new User(  
        rs.getInt(columnLabel: "id"),  
        rs.getString(columnLabel: "userid"),  
        rs.getString(columnLabel: "name"),  
        rs.getString(columnLabel: "password"),  
        rs.getInt(columnLabel: "age"),  
        rs.getBoolean(columnLabel: "membership"),  
        rs.getTimestamp(columnLabel: "signup_date")  
    );  
    users.add(user);  
}  
users.forEach(System.out::println);  
} catch (SQLException e) {  
    throw new RuntimeException(e);  
}
```

쿼리의 결과 데이터 행이 끝날 때까지
반복하여 사용자 정보를 읽기

각각 컬럼의 데이터 타입에 맞게
rs 로 부터 데이터 값을 받아서
User 생성자에 전달

각각 컬럼의 데이터 타입에 맞게
rs 로 부터 데이터 값을 받아서
User 생성자에 전달



```
while (rs.next()) {
    User user = new User(
        rs.getInt(columnLabel: "id"),
        rs.getString(columnLabel: "userid"),
        rs.getString(columnLabel: "name"),
        rs.getString(columnLabel: "password"),
        rs.getInt(columnLabel: "age"),
        rs.getBoolean(columnLabel: "membership"),
        rs.getTimestamp(columnLabel: "signup_date")
    );
    users.add(user);
}
users.forEach(System.out::println);
// 간결화 안한 버전
// users.forEach(user -> System.out.println(user));
} catch (SQLException e) {
    throw new RuntimeException(e);
}
```

생성자를 통해 생성된 회원 객체를
ArrayList 에 추가

User 리스트에 저장된 회원 정보를
오버라이딩 된 toString 메서드를
실행하여 출력!

간결화된 메서드 참조 구문을 사용



프로그램에서 조회 기능 호출

```
if (choice == 1) {  
    manageUser.getAllUsers();  
} else if (choice == 2) {  
    System.out.print("추가할 회원의 ID: ");  
}
```

회원 목록 조회 선택 시
구현한 getAllUsers() 메서드 실행!



===== 회원 관리 프로그램 =====

1. 회원 목록 조회
2. 회원 추가
3. 회원 삭제

원하는 작업 번호를 입력하세요: 1

[id=1, userid='xenosign', name='이효석', age=40, membership=true, signupDate=2024-07-16 09:06:20.0]

[id=2, userid='kimsiwan', name='김시완', age=30, membership=true, signupDate=2024-07-16 09:06:20.0]

[id=3, userid='imsiwan', name='임시완', age=30, membership=true, signupDate=2024-07-16 09:06:20.0]

[id=4, userid='pororo', name='뽀로로', age=3, membership=false, signupDate=2024-07-16 09:06:20.0]

[id=5, userid='luppy', name='김루피', age=10, membership=true, signupDate=2024-07-16 09:06:20.0]

[id=7, userid='faker', name='이상혁', age=28, membership=true, signupDate=2024-07-16 09:06:20.0]

===== 회원 관리 프로그램 =====



실습, 특정 철자가 들어가는 회원 검색 기능!



- 방금 작성한 ManageUser 클래스에 searchUsersByName(String namePart) {} 를 추가해 주세요!
- 사용자로 부터 특정 철자, 또는 문자를 입력 받아서 해당 철자 또는 문자가 들어간 사용자 전부를 출력하는 기능을 구현해 주세요
- 아래의 코드와 실행 결과를 보고 실습을 완성해 주세요

ManageUserProgram



```
while (true) {  
    System.out.println("==== 회원 관리 프로그램 =====");  
    System.out.println("1. 회원 목록 조회");  
    System.out.println("2. 회원 추가");  
    System.out.println("3. 회원 삭제");  
    System.out.println("4. 특정 이름을 가지는 회원 조회");  
    System.out.print("원하는 작업 번호를 입력하세요: ");
```

```
} else if (choice == 4) {  
    System.out.print("검색할 이름의 일부를 입력하세요 : ");  
    String namePart = scanner.nextLine();  
    manageUser.searchUsersByName(namePart);  
}
```

ManageUser 클래스의 메서드



```
public void searchUsersByName(String namePart) {  
    // 기능을 구현해 주세요
```

실행 결과 화면

4. 특정 이름을 가지는 회원 조회

원하는 작업 번호를 입력하세요: 4

검색할 이름의 일부를 입력하세요 : 김

[id=2, userid='kimsiwan', name='김시완', age=30, membership=true, signupDate=2024-07-16 09:06:20.0]

[id=5, userid='luppy', name='김루피', age=9, membership=true, signupDate=2024-07-16 09:06:20.0]

원하는 작업 번호를 입력하세요: 4

검색할 이름의 일부를 입력하세요 : 광두팔

해당 철자가 포함 된 회원이 존재하지 않습니다.



도전 실습!

도전 실습1, 유저 정보 수정 기능 구현하기!



1	1	xenosign	이효석	1234	40
2	2	kimsiwan	김시완	1234	30
3	3	imsiwan	임시완	1234	38
4	4	pororo	뽀로로	1234	10
5	5	luppy	김루피	1234	9
6	7	faker	이상혁	goat	26



5. 회원 정보 수정

6. 로그아웃

원하는 작업 번호를 입력하세요: 5

수정할 회원의 ID를 입력하세요: 1

새 이름: 이효석수정

새 비밀번호: abcd

새 나이: 41

새 멤버십 여부 (true/false): true

ID 가 1 인 회원 정보가 성공적으로 업데이트 되었습니다.

데이터 수정이 완료 된 모습!

1	1	xenosign	이효석수정	abcd	41
2	2	kimsiwan	김시완	1234	30
3	3	imsiwan	임시완	1234	38
4	4	pororo	뽀로로	1234	10
5	5	luppy	김루피	1234	9
6	7	faker	이상혁	goat	26



도전 실습2, 로그인 기능 구현하기!

- user_table 의 정보를 이용해서 로그인 기능을 구현해 주세요

```
회원 관리 프로그램에 오신 것을 환영합니다.  
로그인을 해주십시오.
```

```
ID: xenosign
```

```
PASSWORD: abcd
```

```
===== 회원 관리 프로그램 =====
```

1. 회원 목록 조회
2. 회원 추가
3. 특정 이름이 포함된 유저 검색
4. 회원 삭제
5. 회원 정보 수정
6. 로그아웃

```
원하는 작업 번호를 입력하세요:
```

```
회원 관리 프로그램에 오신 것을 환영합니다.  
로그인을 해주십시오.
```

```
ID: xenosign
```

```
PASSWORD: 1234
```

```
로그인 정보가 잘못되었습니다. 다시 시도하십시오.
```

```
회원 관리 프로그램에 오신 것을 환영합니다.
```

```
로그인을 해주십시오.
```

```
ID:
```