

# 13장. Java – 오라클 DB 연동



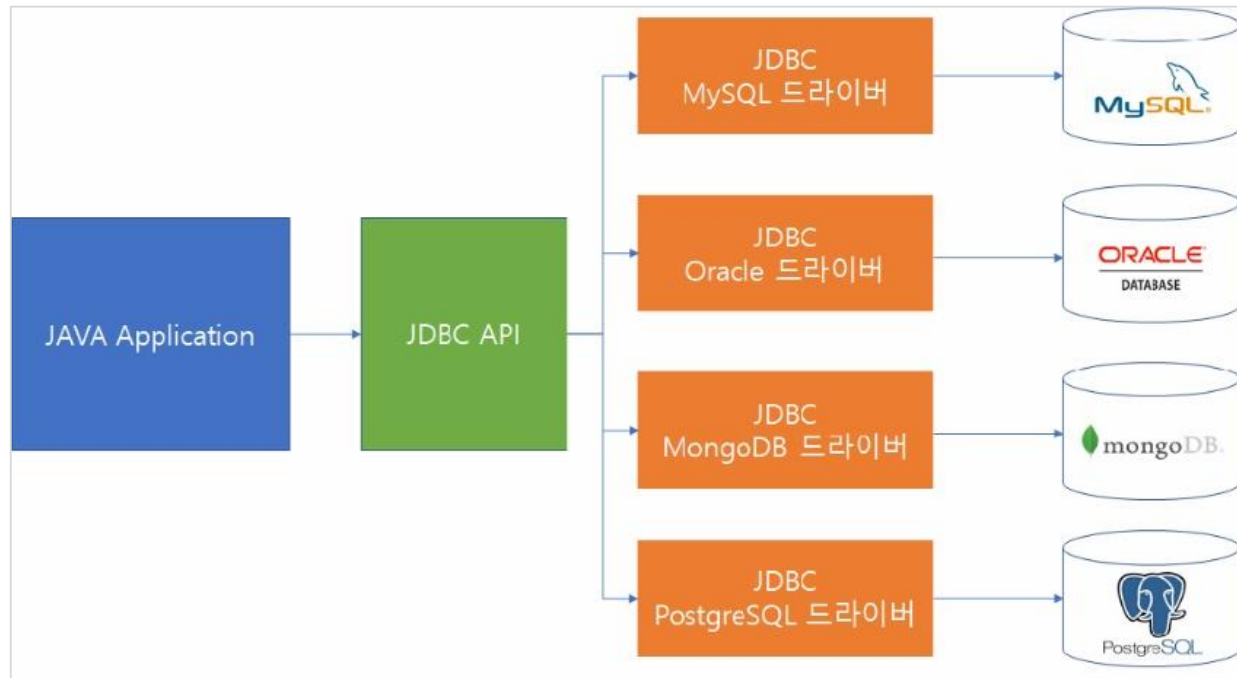
*JDBC Connection*



# JDBC(Java Database Connectivity)

## ◆ JDBC(Java Database Connectivity) 정의와 사용

- 자바 애플리케이션에서 DBMS에 연결해주는 기능을 갖는 API이다.
- 오라클, MySQL, MS-SQL 개발사가 구현한 '드라이버'가 존재함



# 오라클 데이터 베이스

## ❖ 오라클 데이터베이스와 버전

### ▷ Oracle 데이터베이스

- 오라클사가 만든 DBMS 제품이다.
- 최신 버전은 2021년에 출시한 21c 버전이다.

현재 일반적으로 Express-Edition 버전을 많이 사용한다.

### ▷ Oracle 데이터베이스 설치

1. 계정 생성하고 로그인 하기
2. 다운로드 하기
3. 파일 압축 풀기
4. 설치 프로그램 실행하기



# 오라클 데이터 베이스

## ❖ oracle download > oracle Database XE download

### 데이터베이스

[Audit Vault and Database Firewall](#)

[Berkeley DB](#)

[Big Data Connectors](#)

[클러스터 검증 유틸리티](#)

[Database Enterprise/Standard Edition](#)

[Database Express Edition](#)

[데이터베이스 모바일 서버](#)

[Grid Infrastructure](#)

[Instant Client](#)

[MySQL](#)

### Run Oracle Database 21c XE on Windows

[Download Oracle Database XE for Windows \(ZIP\)](#)

[Oracle Database XE Installation Guide for Windows](#)

[Oracle Database 21c XE Quick Start](#)

### 오라클 계정 로그인

사용자 이름

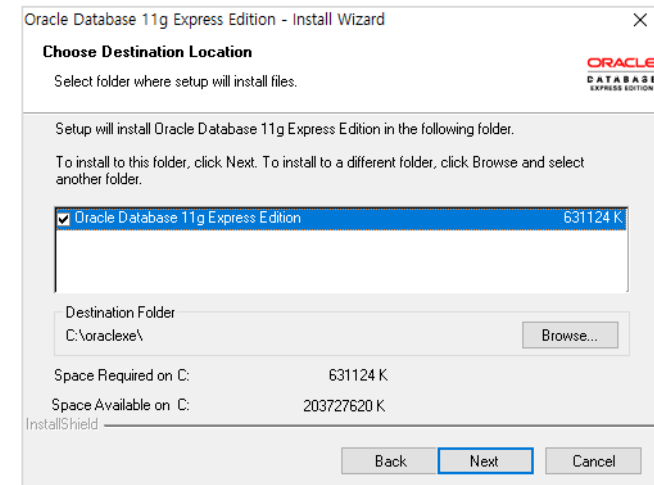
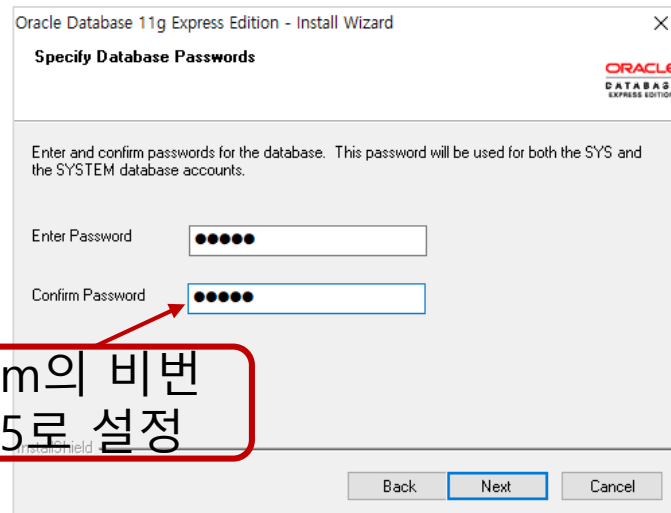
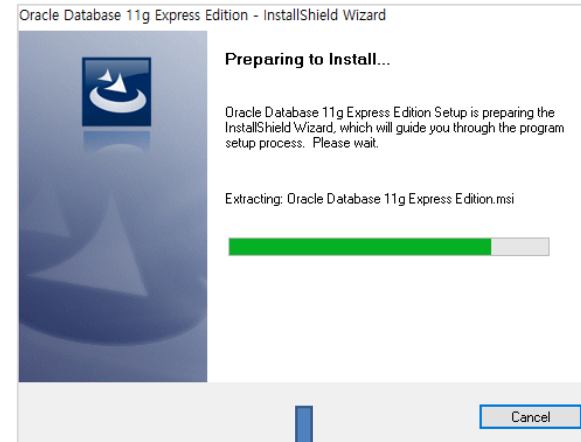
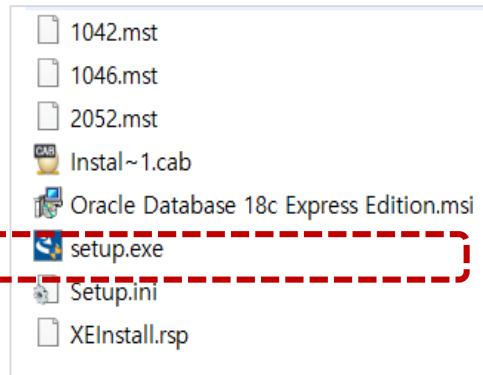


암호



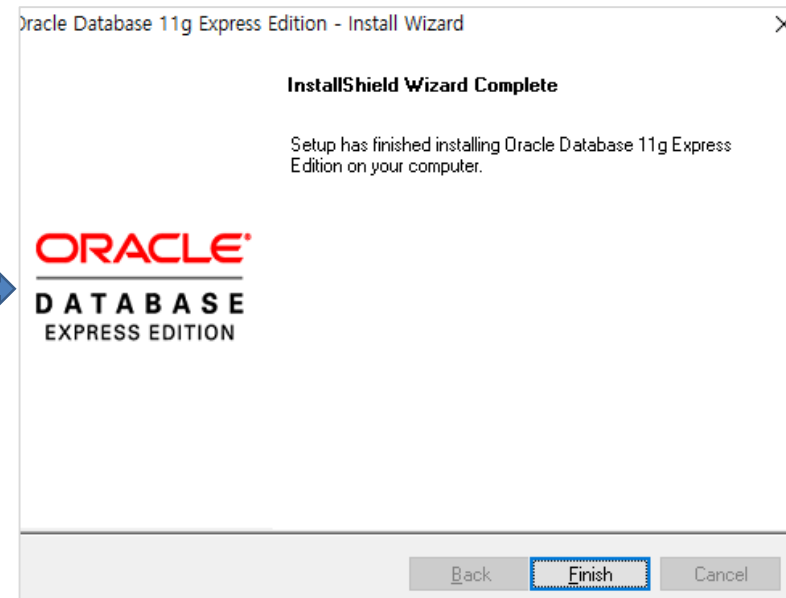
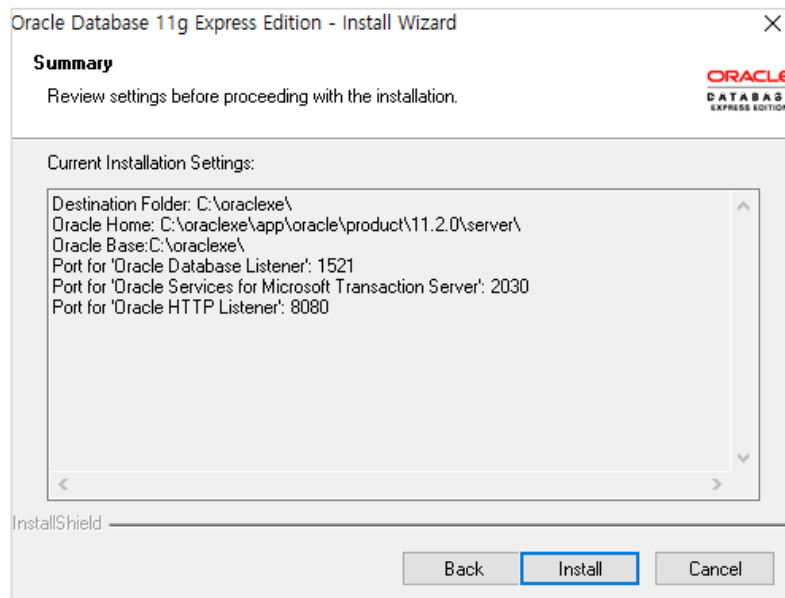
# 오라클 데이터 베이스

## ❖ 오라클 데이터베이스와 설치



# 오라클 데이터 베이스



## ❖ 오라클 데이터베이스와 설치



# 오라클 데이터 베이스

## ❖ 오라클 데이터베이스와 설치 후 확인

내컴퓨터 – 우측마우스 – 관리 – 서비스 및 응용프로그램 - 서비스

 OracleJobSchedulerXE		사용 안 함
 OracleOraDB21Home1MTSR...	실행 ...	자동
 OracleOraDB21Home1TNSLi...	실행 ...	자동
 OracleServiceXE	실행 ...	자동
 OracleVssWriterXE		자동



# sqlplus – 개발 도구

## ■ sqlplus 사용하기

오라클 설치시 자동으로 지원되는 프로그램이다.

- ① 명령프롬프트(cmd) 열기
- ② **sqlplus**(DBMS 소프트웨어) 입력
- ③ 사용자명(user) : **system**
- ④ 비밀번호 : **12345**

```
C:\Users\kkyon>sqlplus

SQL*Plus: Release 21.0.0.0.0 - Production on 토 7월 2 08:28:19 2022
Version 21.3.0.0.0

Copyright (c) 1982, 2021, Oracle. All rights reserved.

사용자명 입력: system
비밀번호 입력:
마지막 성공한 로그인 시간: 토 7월 02 2022 08:21:30 +09:00

다음에 접속됨:
Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0

SQL>
```





# SQL developer

## ◆ 오라클 SQL 디벨로퍼 : (sqldeveloper 다운로드 검색)

Oracle SQL developer는 오라클 데이터베이스에서 SQL 작업을 수행하는 통합 개발환경(IDE)이다.

SQL과 PL/SQL 코드 작성을 위해 다양한 기능을 제공하며 프리웨어이다.

### Developer Resources

#### Tools

[APEX](#)

[Oracle REST Data Services \(ORDS\)](#)

[SQL Developer](#)

[SQL Developer Command Line \(SQLcl\)](#)

#### Drivers

[JDBC and UCP](#)

[ADO.NET and OLE DB Providers](#)




[Python Driver](#)

[Instant Client - C/C++ Drivers \(OCI Utilities\)](#)



# 오라클 SQL 디벨로퍼

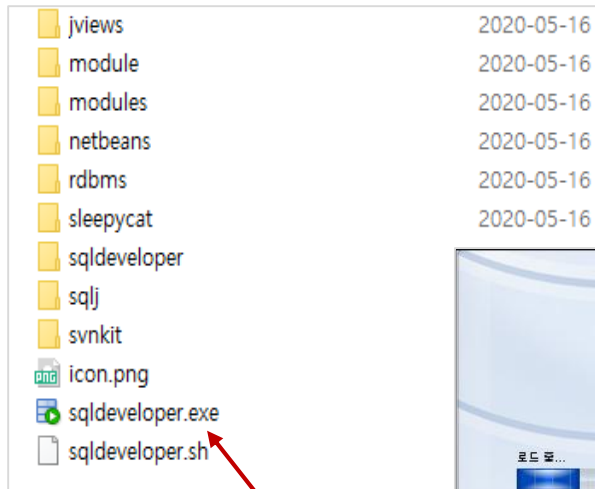
## ◆ 오라클 SQL 디벨로퍼 Download

Platform	Download
SQL Developer Extension for VSCode	 <a href="#">Download</a>
Windows 64-bit with JDK 17 included	 <a href="#">Download (555 MB)</a>
Windows 32-bit/64-bit	 <a href="#">Download (557 MB)</a>

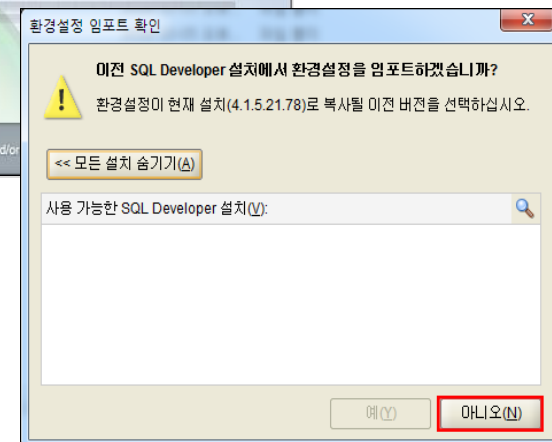


# 오라클 SQL 디벨로퍼

## ◆ Sqldeveloper 설치 – 다운로드후 압축풀기



실행

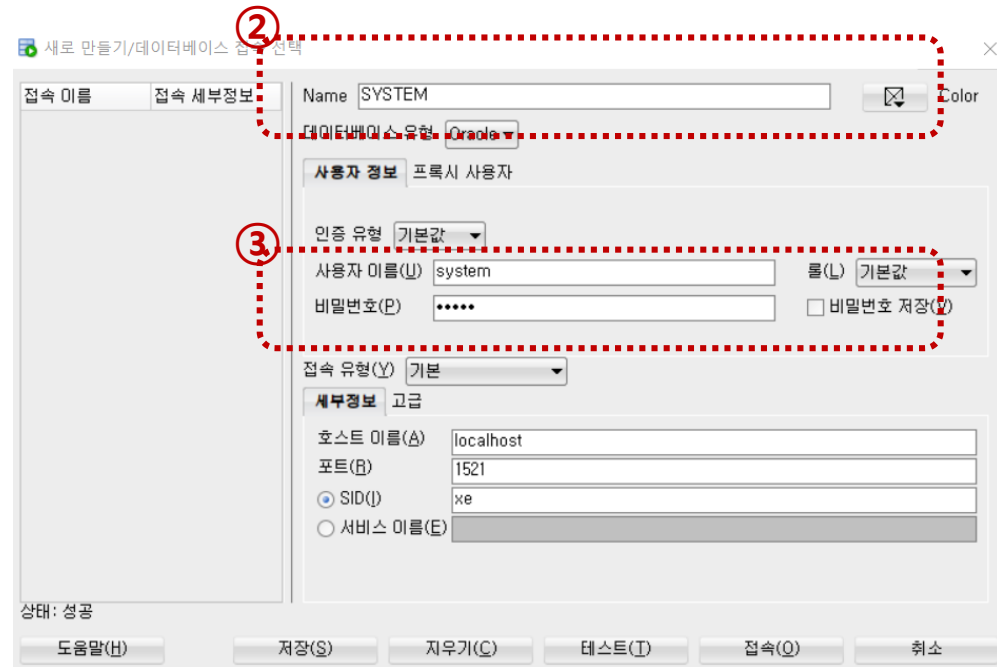
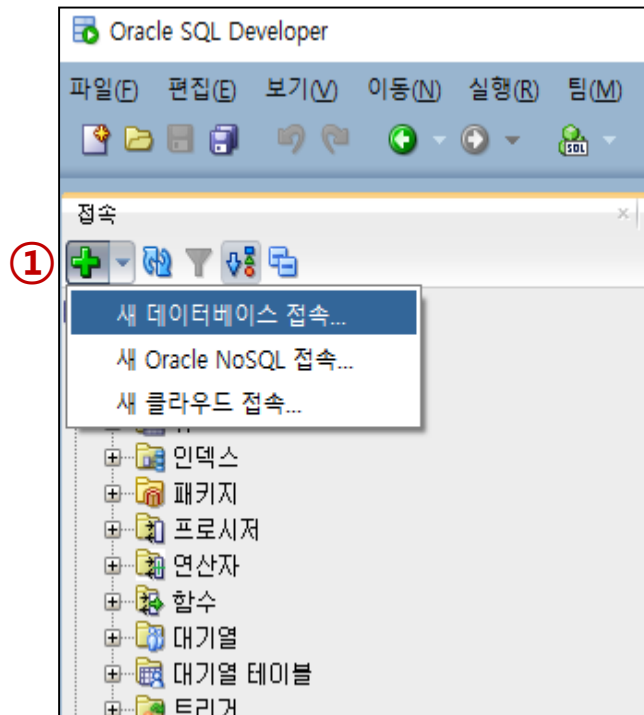


# 데이터베이스 생성

## ◆ 데이터베이스 만들기 및 접속

### 1. 관리자 (system) 계정만들기

Name-SYSTEM, 사용자이름- system, 비밀번호 - 12345

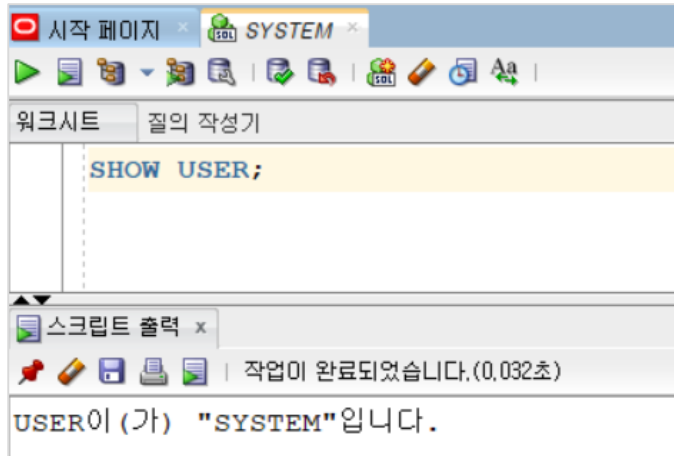


테스트후 성공  
하면 접속



# User(계정) 보기

## USER 보기



system 파일 저장하기 – user.sql



# SQL – DDL(데이터 정의어)

## ● DDL(Data Definition Language)

- 데이터를 저장할 테이블의 구조를 만드는 명령어이다.
- DDL은 번역한 결과가 데이터 사전이라는 특별한 파일에 여러 개의 테이블로써 저장된다.
- CREATE SCHEMA, CREATE TABLE, CREATE VIEW, CREATE INDEX, ALTER, DROP 등이 있다.

명령어	설 명
CREATE	테이블을 생성함
ALTER	테이블의 구조를 변경함
DROP	테이블을 삭제함



# SQL – DML(데이터 조작어)

## ● DML(Data Manipulation Language)

- 데이터를 조작하는 명령어이다.
- 데이터를 조작하여 저장하는 일련의 과정을 **트랜잭션(transition)**이라 하며, DML은 트랜잭션을 다루는 명령어이다.

명령어	설 명
SELECT	테이블에 있는 행을 검색
INSERT	테이블에 새로운 행 삽입
UPDATE	테이블에 있는 행의 내용 갱신
DELETE	테이블의 행을 삭제
COMMIT	작업을 수행
ROLLBACK	작업 수행을 취소



## ● DCL(Data Control Language)

- DCL은 데이터의 보안, 무결성, 회복, 병행 제어 등을 정의하는 데 사용하는 언어이다.
- DDL은 데이터 관리자가(DBA)가 데이터 관리를 목적으로 사용한다.
- DCL에는 GRANT, REVOKE, COMMIT, ROLLBACK, SAVEPOINT 등이 있다.

### ▪ GRANT / REVOKE

**GRANT:** 권한 부여를 위한 명령어

**REVOKE:** 권한 취소를 위한 명령어

```
GRANT RESOURCE TO TODAY;
```

사용자 ID가 'TODAY'인 사람에게 DB 및 테이블을 생성할 수 있는 권한을 부여함

```
GRANT CONNECT TO CLOUD;
```

사용자 ID가 'CLOUD'인 사람에게 DB에 있는 정보를 검색할 수 있는 권한을 부여





# SQL – DML(데이터 조작용어)

- DML(Data Manipulation Language)

- INSERT 문 – 자료 삽입

```
INSERT INTO 테이블 이름(열이름1, 열이름2...)  
VALUES (데이터 값1, 데이터값 2, ...)
```

- SELECT 문 – 자료 검색

```
SELECT 열이름 (or 별칭)  
FROM 테이블 이름
```



# SQL – DML(데이터 조작어)

- DML(Data Manipulation Language)

- UPDATE 문 – 레코드 수정

```
UPDATE 테이블 이름 SET 칼럼이름 = 변경할 값  
WHERE 칼럼이름 = 칼럼값
```

- DELETE 문 – 레코드 삭제

```
DELETE 열이름 (or 별칭)  
FROM 테이블 이름 WHERE 칼럼이름 = 칼럼값
```



# DB – 사원 관리

## ● 사원 테이블 생성 및 자료 삽입

```
-- 테이블 생성
CREATE TABLE employee(
    id CHAR(4),          -- 사원 아이디
    name VARCHAR2(10),  -- 이름
    salary NUMBER        -- 급여
);

-- 데이터 삽입 (추가)
INSERT INTO employee(id, name, salary) VALUES ('e101', '신유빈', 3000000);
INSERT INTO employee(id, name, salary) VALUES ('e102', '이정후', 2000000);
INSERT INTO employee(id, name) VALUES ('e103', '임시현');

-- 사원의 모든 정보 검색
SELECT * FROM employee;
```

	ID	NAME	SALARY
1	e101	신유빈	3000000
2	e102	이정후	2000000
3	e103	임시현	(null)



## ● 사원 검색

```
-- 사원의 아이디와 이름 검색
SELECT id, name FROM employee;

-- 아이디가 'e101'인 사원 검색
SELECT * FROM employee WHERE id = 'e101';

-- 이름이 '신유빈'인 사원 정보 검색
SELECT * FROM employee WHERE name LIKE '신유빈';
```



# DB – 사원 관리

## ● 사원 수정

```
-- 이정후의 급여를 5000000원으로 변경
UPDATE employee SET salary = 5000000
WHERE name = '이정후';

-- id가 'e103'인 사원의 이름을 '안산', 급여를 3500000원으로 수정
UPDATE employee SET name = '안산', salary = 3500000
WHERE id = 'e103';
```

	ID	NAME	SALARY
1	e101	신유빈	3000000
2	e102	이정후	5000000
3	e103	안산	3500000



# DB – 사원 관리

- 사원 삭제

```
-- id가 'e103'인 사원 삭제  
DELETE FROM employee WHERE id = 'e103';
```

	ID	NAME	SALARY
1	e101	신유빈	3000000
2	e102	이정후	5000000



## ● 제약 조건

테이블들은 각 속성(칼럼)에 대한 무결성을 유지하기 위한 다양한 제약 조건 (Constraints)이 적용되어 있다.

제약 조건에는 NOT NULL, 기본키, 외래키, CHECK 등이 있다.

### ✓ NOT NULL

칼럼을 정의할 때 NOT NULL 제약 조건을 명시하면 반드시 데이터를 입력해야 한다.

칼럼명 데이터 타입 **NOT NULL**



- 제약 조건

- ✓ 기본키(PRIMARY KEY)

기본키는 Primary Key라고도 하며, UNIQUE와 NOT NULL 속성을 동시에 가진 제약 조건으로 테이블 당 1개의 기본키만 생성할 수 있다.

칼럼명 데이터 타입 **PRIMARY KEY**

또는

**PRIMARY KEY**(칼럼명, ...)





## ● 테이블 삭제후 테이블 생성

```
-- 테이블 생성
CREATE TABLE employee(
    id CHAR(4) PRIMARY KEY,      -- 사원 아이디 (기본키)
    name VARCHAR2(10) NOT NULL,  -- 이름 (필수 입력)
    salary NUMBER                -- 급여
);

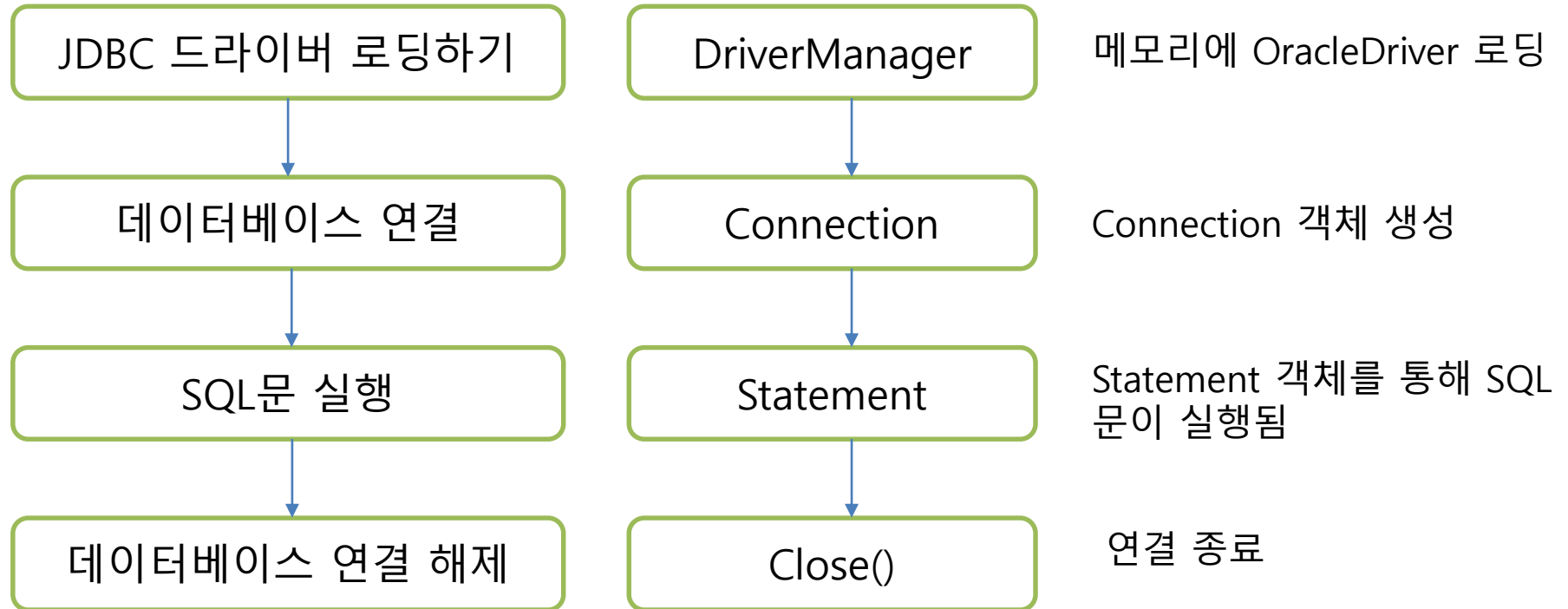
-- 테이블 삭제
DROP TABLE employee;

-- 데이터 삽입 (추가)
INSERT INTO employee(id, name, salary) VALUES ('e101', '신유빈', 3000000);
INSERT INTO employee(id, name, salary) VALUES ('e102', '이정후', 2000000);
INSERT INTO employee(id, name) VALUES ('e103', '임시현');
-- 무결성 제약 조건 위반 (기본키 중복)
INSERT INTO employee(id, name) VALUES ('e103', '허미미');
-- 무결성 제약 조건 위반 (NULL 불가)
INSERT INTO employee(id, salary) VALUES ('e104', 4000000);
```



# JDBC(Java Database Connectivity)

## ➤ 데이터베이스 연결 순서



# JDBC(Java Database Connectivity)

## ◆ JDBC를 이용한 데이터베이스 연동하기

ojdbc 드라이버 구하기

1. 오라클 설치 경로    2. sql developer 설치 경로

로컬 디스크 (C:) > app > kiyon > product > 21c > dbhomeXE > jdbc > lib				
이름	수정된 날짜	유형	크기	
ojdbc8.jar	2021-07-21 오전 2:06	ALZip JAR File	4,936KB	
ojdbc8_g.jar	2021-07-21 오전 2:39	ALZip JAR File	8,213KB	
ojdbc8dms.jar	2021-07-21 오전 2:18	ALZip JAR File	6,852KB	
ojdbc8dms_g.jar	2021-07-21 오전 2:43	ALZip JAR File	8,215KB	
ojdbc11.jar	2021-07-21 오전 2:07	ALZip JAR File	5,027KB	
ojdbc11_g.jar	2021-07-21 오전 2:47	ALZip JAR File	8,357KB	

로컬 디스크 (C:) > sqldeveloper > jdbc > lib	
이름	
ojdbc8.jar	



# JDBC(Java Database Connectivity)

## ◆ JDBC를 이용한 데이터베이스 연동하기 ojdbc 이클립스 프로젝트에 복사하기

The first screenshot shows the project structure of 'Chapter16 [javaWorks master]'. It includes a 'src' folder with subfolders 'dbconnection' and 'person', and a 'lib' folder containing 'ojdbc11.jar'. An arrow points from this jar file to the first step in the list below.

The second screenshot shows a right-click context menu on the 'lib' folder, with 'Build Path' > 'Configure Build Path...' selected.

The third screenshot shows the 'Java Build Path' dialog box. In the 'Libraries' tab, 'JRE System Library [JavaSE-10]' is selected. A red dashed box highlights the 'Add External JARs...' button, which is labeled with a circled '2'.

**JAR Selection**  
Choose the archives to be added to the build path:  
type filter text  
BankApp  
Chapter1  
Chapter10  
Chapter11  
Chapter12  
Chapter13  
Chapter14  
Chapter15  
Chapter16  
  .settings  
  .bin  
  .lib  
    ojdbc6.jar  
  .src  
    .classpath  
  .project

1. 프로젝트 만든후, lib폴더 만들기
2. 오라클 드라이버 .jar파일 복사
3. 클래스 패스 설정

프로젝트 > 우측마우스 > Build Path > Cofigure Build Path > Libraries(Classpath) > Add JARs



# 연결 테스트(Connection Test)

## ◆ 오라클 DBMS에 연결하기

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class ConnectionTest {
    public static void main(String[] args) {
        Connection conn = null;

        try {
            // jdbc 드라이버 등록
            Class.forName("oracle.jdbc.OracleDriver");
            // 연결
            conn = DriverManager.getConnection(
                "jdbc:oracle:thin:@localhost:1521/xe", //db url
                "system", //user 계정
                "pw1234"); //password
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```



# 연결 테스트(Connection Test)

## ◆ 오라클 DBMS에 연결하기

```
        System.out.println(conn + "DB 연결 성공!!");
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        if(conn != null) {
            try {
                conn.close();
                System.out.println("연결 끊음");
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}
```

oracle.jdbc.driver.T4CConnection@16aa0a0aDB 연결 성공!!  
연결 끊음



# 연결 테스트(Connection Test)

## ◆ 개선된 연결 테스트

```
public class DBConnectionTest {  
    //static{} - 정적 초기화 블록  
    //매번 연결시 마다 드라이버를 로드할 필요가 없어짐  
    static {  
        try {  
            // 클래스 로딩시 드라이버 등록  
            Class.forName("oracle.jdbc.OracleDriver");  
        } catch (ClassNotFoundException e) {  
            e.printStackTrace();  
        }  
    }  
  
    static String url = "jdbc:oracle:thin:@localhost:1521/xes"; //db 경로  
    static String username = "system"; //사용자 계정명  
    static String password = "pw1234"; //사용자 비밀번호  
  
    public static void main(String[] args) {  
        //try ~ resource 구문  
        try(Connection conn = DriverManager.getConnection(url, username, password)){  
            System.out.println(conn + ": DB 연결 성공!");  
        } catch (SQLException e1) {  
            e1.printStackTrace();  
        }  
    }  
}
```



# SQL – DDL, DML

## ◆ 테이블 생성 및 데이터 삽입, 검색

```
-- users (회원) 테이블 생성
CREATE TABLE users(
    userid    VARCHAR2(10) PRIMARY KEY,      -- 아이디
    userpassword VARCHAR2(100) NOT NULL,    -- 비밀번호
    username  VARCHAR2(20) NOT NULL,        -- 이름
    userage   NUMBER(3)                     -- 나이
);

-- 회원 추가
INSERT INTO users(userid, userpassword, username, userage)
VALUES ('today', 't1234$#', '이정후', 26);

INSERT INTO users(userid, userpassword, username)
VALUES ('hangang', 'h1234$#', '한강');

-- 전체 회원 검색
SELECT * FROM users;
```

	USERID	USERPASSWORD	USERNAME	USERAGE
1	today	t1234\$#	이정후	26
2	hangang	h1234\$#	한강	(null)





# 오라클 DB – 데이터 타입

## ◆ 오라클 데이터 타입(Data Type)

데이터 타입	설명
CHAR	고정길이 문자, 최대 2000byte
VARCHAR2	가변길이 문자, 최대 4000byte
CLOB(Character)	매우 큰 크기의 문자열 저장, 최대 4GB
BLOB(Binary)	매우 큰 크기의 바이너리 파일 저장, 최대 4GB
NUMBER	가변 숫자, 십진수 기준 최대 220byte
DATE	날짜 – 연, 월, 일
TIMESTAMP	날짜 – 연, 월, 일, 시, 분, 초, 밀리초



# DTO 정의와 사용법

- DTO(Data Transfer Object)의 정의와 사용법

- 여러 다른 타입의 데이터를 다른 클래스로 전달할 때 사용하는 클래스이다.
- 로직이나 서비스 기능등은 없고 단순히 참조되는 자료형이다.
- 만드는 방법

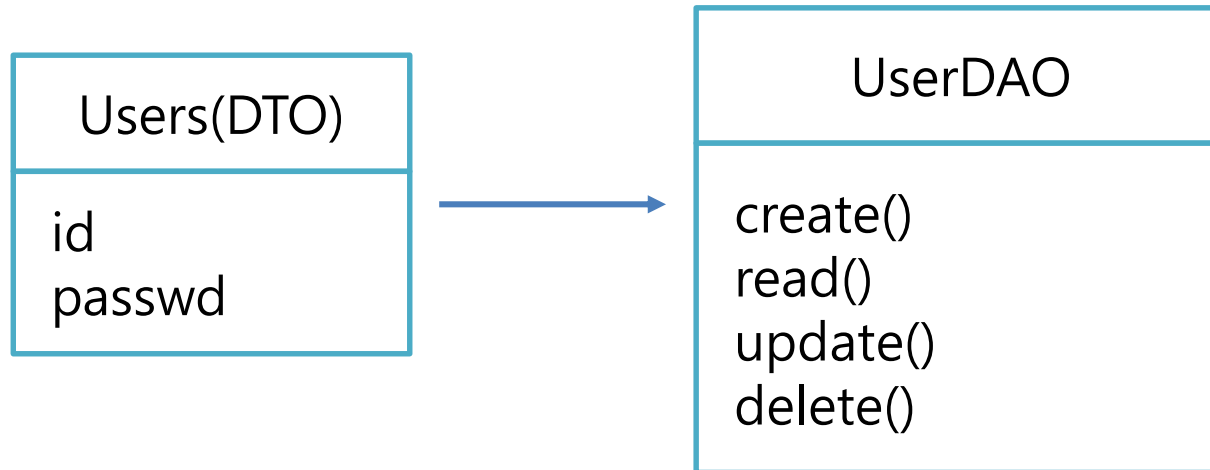
- DB 테이블의 필드명을 속성으로 선언한다.
- 생성자를 구현한다.
- 각 속성에 대한 getter/setter 메서드를 구현한다.



# DAO 정의와 사용법

## ● DAO(Data Access Object)의 정의와 사용법

- 데이터베이스에 연결(접속)하고, 데이터를 생성, 조회, 수정, 삭제 등의 작업을 하는 클래스이다.
- DTO 클래스를 사용하여 작업을 수행한다.



# DTO 정의와 사용법

## ■ DTO(Data Transfer Object)의 정의

```
package dto;
//DTO 정의
public class Users {
    private String userId;        //아이디
    private String userPassword;  //비밀번호
    private String userName;      //이름
    private int userAge;          //나이

    //getter, setter 메서드 정의
    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public String getUserId() {
        return userId;
    }

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



# DTO 정의와 사용법

## ■ DTO(Data Transfer Object)의 정의

```
public String getUserPassword() {  
    return userPassword;  
}  
  
public void setUserPassword(String userPassword) {  
    this.userPassword = userPassword;  
}  
  
public int getUserAge() {  
    return userAge;  
}  
  
public void setUserAge(int userAge) {  
    this.userAge = userAge;  
}  
  
//객체 정보를 문자열로 리턴  
@Override  
public String toString() {  
    return "Users [userId=" + userId + ", userPassword=" + userPassword  
        + ", userName=" + userName + ", userAge=" + userAge + "];"  
}  
}
```



# DAO – 회원 생성

## ■ 회원 생성

```
public class UserDao {  
  
    static {  
        try {  
            // 클래스 로딩시 드라이버 등록  
            Class.forName("oracle.jdbc.OracleDriver");  
        } catch (ClassNotFoundException e) {  
            e.printStackTrace();  
        }  
    }  
  
    static String url = "jdbc:oracle:thin:@localhost:1521/x";  
    static String username = "system";  
    static String password = "pw1234";  
  
    //회원 생성  
    public void insertUser(Users user) {  
        String sql = "INSERT INTO users(userid, userpassword, username, usage) "  
            + "VALUES (?, ?, ?, ?)";  
    }  
}
```



# DAO – DB 연결

## ▪ DB 연결

```
package users.dao;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

public class UserDAO {

    static {
        try {
            // 클래스 로딩시 드라이버 등록
            Class.forName("oracle.jdbc.OracleDriver");
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
    }

    static String url = "jdbc:oracle:thin:@localhost:1521/xe"; //db 경로
    static String username = "system"; //사용자 계정명
    static String password = "pw1234"; //사용자 비밀번호
```



# DAO – 회원 생성

## ■ 회원 생성

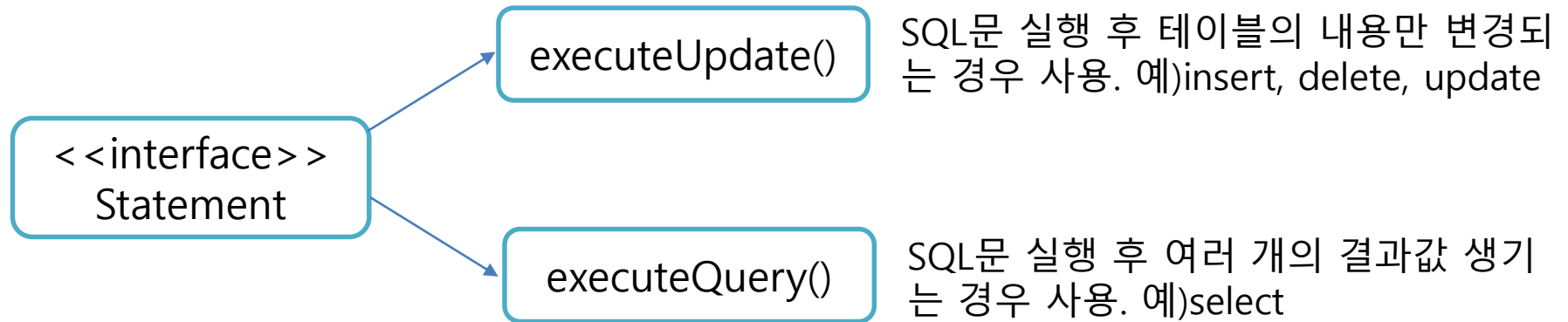
```
public void insertUser(Users user) {  
    //SQL - DML(삽입 구문)  
    String sql = "INSERT INTO users(userid, userpassword, username, userage) "  
        + "VALUES (?, ?, ?, ?)";  
  
    //Connection 객체(db 연결), PreparedStatement 객체(sql 처리) 생성  
    try(Connection conn = DriverManager.getConnection(url, username, password);  
        PreparedStatement pstmt = conn.prepareStatement(sql)){  
  
        pstmt.setString(1, user.getUserId()); //메인(화면)에서 입력된 아이디 가져옴  
        pstmt.setString(2, user.getUserPassword()); //비밀번호  
        pstmt.setString(3, user.getUserName()); //이름  
        pstmt.setInt(4, user.getUserAge()); //나이  
  
        pstmt.executeUpdate(); //sql 실행(커밋)  
    }catch (SQLException e) {  
        e.printStackTrace();  
    }  
}
```



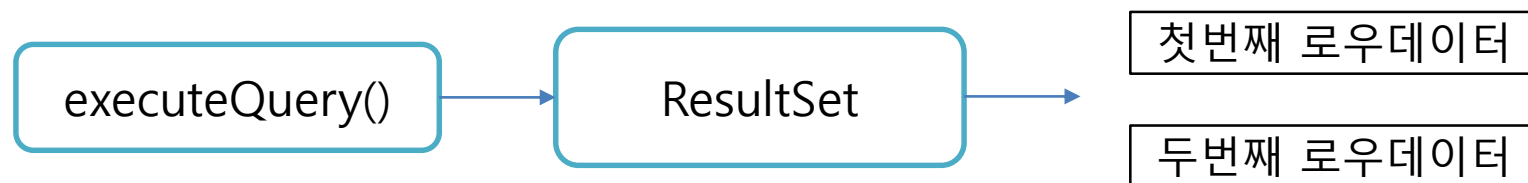


# JDBC(Java Database Connectivity)

## ➤ Statement 객체 살펴보기



### executeQuery() 실행 후 반환 되는 레코드셋



# DAO – 회원 목록 보기

## ■ 회원 목록 보기

```
public List<Users> getUserList() {  
    //SQL - DML(검색 구문)  
    String sql = "SELECT * FROM users";  
    //검색된 user를 저장할 리스트 객체 생성  
    List<Users> userList = new ArrayList<>();  
  
    //ResultSet rs 객체(db에서 가져온 자료)  
    try(Connection conn = DriverManager.getConnection(url, username, password);  
        PreparedStatement pstmt = conn.prepareStatement(sql);  
        ResultSet rs = pstmt.executeQuery()){  
  
        while(rs.next()) { //검색된 자료가 있는 동안 계속  
            Users user = new Users();  
            user.setUserId(rs.getString("userid")); //db에서 가져온 아이디를 보기를 위해 설정  
            user.setUserPassword(rs.getString("userpassword")); //비밀번호(문자형)  
            user.setUsername(rs.getString("username")); //이름(문자형)  
            user.setUserAge(rs.getInt("userage")); //나이(정수형)  
  
            userList.add(user); //user 객체를 리스트에 저장  
        }  
    } catch(SQLException e) {  
        e.printStackTrace();  
    }  
    return userList; //리스트를 반환함  
}
```



# DAO – 회원 상세 보기

## ■ 회원 상세 보기(1건 검색)

```
public Users getUser(String userId) {  
    //SQL - DML(검색 구문)  
    String sql = "SELECT * FROM users WHERE userid = ?";  
    Users user = new Users(); //검색 결과를 보여줄 user 객체 생성  
  
    try(Connection conn = DriverManager.getConnection(url, username, password);  
        PreparedStatement pstmt = conn.prepareStatement(sql)){  
        pstmt.setString(1, userId); //실행 화면에서 입력된 아이디 설정  
  
        try(ResultSet rs = pstmt.executeQuery()){ //검색된 자료 자료가 객체 생성  
            if(rs.next()) { //검색된 자료가 있으면  
                user.setUserId(rs.getString("userid"));  
                user.setUserPassword(rs.getString("userpassword"));  
                user.setUserName(rs.getString("username"));  
                user.setUserAge(rs.getInt("userage"));  
            }  
        }  
    } catch (SQLException e) {  
        e.printStackTrace();  
    }  
    return user; //user 객체 반환  
}
```



# DAO – 회원 수정

## ■ 회원 수정

```
public void updateUser(Users user) {  
    //SQL - DML(수정 구문)  
    String sql = "UPDATE users SET userpassword = ?, username = ?, usage = ? "  
        + "WHERE userid = ?";  
  
    try(Connection conn = DriverManager.getConnection(url, username, password);  
        PreparedStatement pstmt = conn.prepareStatement(sql)){  
  
        pstmt.setString(1, user.getUserPassword());  
        pstmt.setString(2, user.getUserName());  
        pstmt.setInt(3, user.getUserAge());  
        pstmt.setString(4, user.getUserId());  
  
        pstmt.executeUpdate(); //sql 실행(커밋)  
    } catch (SQLException e) {  
        e.printStackTrace();  
    }  
}
```



# DAO – 회원 삭제

## ■ 회원 삭제

```
public void deleteUser(String userId) {  
    //SQL - DML(삭제 구문)  
    String sql = "DELETE FROM users WHERE userid = ?";  
  
    try(Connection conn = DriverManager.getConnection(url, username, password);  
        PreparedStatement pstmt = conn.prepareStatement(sql)){  
        pstmt.setString(1, userId); //외부 입력된 아이디  
  
        pstmt.executeUpdate(); //sql 실행(커밋)  
    } catch (SQLException e) {  
        e.printStackTrace();  
    }  
}
```



# DAO – 회원 관리 테스트

## ■ 회원 관리 테스트

```
public class UserMain {  
  
    public static void main(String[] args) {  
        Users newUser = new Users();    //회원 객체 생성  
        UserDao dao = new UserDao(); //회원 관리 객체 생성  
  
        //회원 추가  
        newUser.setUserId("today");  
        newUser.setUserPassword("t135##");  
        newUser.setUserName("투데이");  
        newUser.setUserAge(6);  
  
        dao.insertUser(newUser); //db에 회원 추가  
        System.out.println("회원 1명 저장 완료!");  
  
        //회원 상세 보기  
        Users user = dao.getUser("cloud");  
        System.out.println(user);  
    }  
}
```



# DAO – 회원 관리 테스트

## ■ 회원 관리 테스트

```
//회원 정보 수정
Users renewUser = new Users();
renewUser.setUserId("today");
renewUser.setUserPassword("t135##");
renewUser.setUserName("박이슬");
renewUser.setUserAge(30);

dao.updateUser(renewUser);
System.out.println("회원 정보 수정 완료!");

//회원 삭제
dao.deleteUser("today");
System.out.println("회원 정보 삭제 완료!");

//회원 목록
List<Users> userList = dao.getUserList();
for(int i = 0; i < userList.size(); i++) {
    Users findUser = userList.get(i);
    System.out.println(findUser);
}
Users [userId=cloud, userPassword=c135##, userName=이정후, userAge=26]
Users [userId=arrow, userPassword=a135##, userName=임시현, userAge=21]
```



# 게시판 구현

## ◆ 게시판 화면 - 1단계

[게시물 목록]

-----  
no   writer            date            title

-----  
1    today12          2023.10.13    오늘도 좋은하루 되세요~

-----  
1.Create | 2.Read | 3.Clear | 4.Exit

선택: 1

create() 메서드 실행됨





# 게시판 구현

## ◆ 게시판 화면

```
public class BoardExample1 {  
  
    private Scanner scanner = new Scanner(System.in);  
    private Connection conn;  
  
    //생성자  
    public BoardExample1() {  
        try {  
            Class.forName("oracle.jdbc.OracleDriver");  
            conn = DriverManager.getConnection(  
                "jdbc:oracle:thin:@localhost:1521/xe",  
                "c##khsql",  
                "pwkhsql");  
            //System.out.println("db 연결 성공!!");  
        } catch (Exception e) {  
            e.printStackTrace();  
            exit();  
        }  
    }  
}
```



# 게시판 구현

## ◆ 게시판 화면

```
public void list() {  
    System.out.println();  
    System.out.println("[게시물 목록]");  
    System.out.println("-----");  
    System.out.printf("%-4s%-12s%-12s%-40s\n", "no", "writer", "date", "title");  
    System.out.println("-----");  
    System.out.printf("%-4s%-12s%-12s%-40s\n",  
        "1", "today12", "2023.10.13", "오늘도 좋은하루 되세요~");  
    System.out.println();  
  
    //메인 메뉴 호출  
    mainMenu();  
}
```



# 게시판 구현

## ◆ 게시판 화면

```
public void mainMenu() {  
    System.out.println("-----");  
    System.out.println("1.Create | 2.Read | 3.Clear | 4.Exit");  
    System.out.print("선택: ");  
    String menuNo = scanner.nextLine();  
  
    switch(menuNo) {  
        case "1":  
            create(); break;  
        case "2":  
            read(); break;  
        case "3":  
            clear(); break;  
        case "4":  
            exit(); break;  
    }  
}
```



# 게시판 구현

## ◆ 게시판 화면

```
public void create() {
    System.out.println("create() 메서드 실행됨");
    list();
}
public void read() {
    System.out.println("read() 메서드 실행됨");
    list();
}
public void clear() {
    System.out.println("clear() 메서드 실행됨");
    list();
}
public void exit() {
    System.out.println("*** 게시판을 종료합니다. ***");
    System.exit(0);
}
public static void main(String[] args) {
    BoardExample1 boardExample = new BoardExample1();
    boardExample.list();
}
```



# 게시판 구현

## ◆ board 테이블 생성

```
-- 게시판 테이블 작성
CREATE TABLE board(
    bno          NUMBER PRIMARY KEY,      -- 글번호
    btitle       VARCHAR2(100) NOT NULL,  -- 글제목
    bcontent     CLOB NOT NULL,           -- 글내용
    bwriter      VARCHAR2(50),           -- 글쓴이
    bdate        DATE DEFAULT SYSDATE     -- 작성일
);

-- 시퀀스(일련번호) 생성
CREATE SEQUENCE seq_bno;
```



# 게시판 구현

## ◆ board 테이블 생성

```
-- 게시물 등록 (작성일은 자동 생성됨으로 생략함)
INSERT INTO board(bno, btitle, bcontent, bwriter)
VALUES (seq_bno.NEXTVAL, '삼성 스마트폰', '삼성 갤럭시 24입니다.', '이재용');
INSERT INTO board(bno, btitle, bcontent, bwriter)
VALUES (seq_bno.NEXTVAL, '애플 스마트폰', 'Apple 아이폰입니다.', '스티브잡스');
INSERT INTO board(bno, btitle, bcontent, bwriter)
VALUES (seq_bno.NEXTVAL, '엘지 노트북', 'LG gram 노트북입니다.', '임시현');

-- 최근 게시물 순으로 정렬
SELECT * FROM board ORDER BY bdate DESC;

-- 글 제목에 '스마트폰'을 포함한 게시물 출력
SELECT * FROM board WHERE btitle LIKE '%스마트폰%';

-- 글 제목에 '스마트폰'을 포함하고 글쓴이가 '이재용'인 게시물 검색
SELECT * FROM board
WHERE btitle LIKE '%스마트폰%' AND bwriter LIKE '이재용';
```



# 게시판 구현

## ◆ Board 클래스 생성

```
public class Board implements Serializable{

    private static final long serialVersionUID = 1L;

    private int bno;
    private String btitle;
    private String bcontent;
    private String bwriter;
    private Date bdate;

    public int getBno() {
        return bno;
    }
    public void setBno(int bno) {
        this.bno = bno;
    }
    public String getBtitle() {
        return btitle;
    }
    public void setBtitle(String btitle) {
        this.btitle = btitle;
    }
}
```



# 게시판 구현

## ◆ Board 클래스 생성

```
public String getBcontent() {  
    return bcontent;  
}  
public void setBcontent(String bcontent) {  
    this.bcontent = bcontent;  
}  
public String getBwriter() {  
    return bwriter;  
}  
public void setBwriter(String bwriter) {  
    this.bwriter = bwriter;  
}  
public Date getBdate() {  
    return bdate;  
}  
public void setBdate(Date bdate) {  
    this.bdate = bdate;  
}  
}
```





# 게시판 구현

## ◆ 게시판 화면 - list() 메뉴 개발

[게시물 목록]

```
-----  
no  writer      date      title  
-----  
1   today       2023-10-16  smartphone
```

```
-----  
1.Create | 2.Read | 3.Clear | 4.Exit
```

선택:



# 게시판 구현

## ◆ 게시판 화면 - list() 메뉴 개발

```
public void list() {
    System.out.println("[게시물 목록]");
    System.out.println("-----");
    System.out.printf("%-4s%-12s%-12s%-40s\n", "no", "writer", "date", "title");
    System.out.println("-----");

    //board 테이블에서 게시물 정보를 가져와서 출력하기
    try {
        String sql = "SELECT bno, btitle, bcontent, bwriter, bdate"
            + " FROM board ORDER BY bno DESC";
        pstmt = conn.prepareStatement(sql);
        ResultSet rs = pstmt.executeQuery();
        while(rs.next()) {
            Board board = new Board();
            board.setBno(rs.getInt("bno"));
            board.setBwriter(rs.getString("bwriter"));
            board.setBtitle(rs.getString("btitle"));
            board.setBdate(rs.getDate("bdate"));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```



# 게시판 구현

## ◆ 게시판 화면 - list() 메뉴 개발

```
        System.out.printf("%-4s%-12s%-12s%-40s \n",
                           board.getBno(),
                           board.getBwriter(),
                           board.getBdate(),
                           board.getBtitle()
                           );
    }
    rs.close();
    pstmt.close();
} catch (SQLException e) {
    e.printStackTrace();
    exit(); //종료 함수 호출
}
//메인 메뉴 호출
mainMenu();
}
```



# 게시판 구현

## ◆ 게시판 화면 - Create 메뉴 개발

```
-----  
1.Create | 2.Read | 3.Clear | 4.Exit  
선택: 1
```

[새 게시물 입력]

제목: 가입 인사

내용: 가입 인사 드립니다. 잘 부탁드립니다~

작성자: cloud

```
-----  
1.Ok | 2.Cancel
```

선택: 1

[게시물 목록]

```
-----  
no  writer      date      title  
-----  
4   cloud       2023-10-16  가입 인사  
3   김기용      2023-10-16  제목 수정  
1   today       2023-10-16  smartphone
```



# 게시판 구현

## ◆ 게시판 화면 - Create 메뉴 개발

```
public void create() {  
    Board board = new Board();  
    System.out.println("[새 게시물 입력]");  
    System.out.print("제목: ");  
    board.setBtitle(scanner.nextLine());  
    System.out.print("내용: ");  
    board.setBcontent(scanner.nextLine());  
    System.out.print("작성자: ");  
    board.setBwriter(scanner.nextLine());  
  
    //확인, 취소 메뉴 생성  
    System.out.println("-----");  
    System.out.println("1.Ok | 2.Cancel");  
    System.out.print("선택: ");  
    String menuNo = scanner.nextLine();  
}
```



# 게시판 구현

## ◆ 게시판 화면 - Create 메뉴 개발

```
if(menuNo.equals("1")) { //1번 선택
    try {
        String sql = "INSERT INTO board(bno, btitle, bcontent, bwriter) "
            + "VALUES(SEQ_BNO.NEXTVAL, ?, ?, ?)";
        pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, board.getBtitle());
        pstmt.setString(2, board.getBcontent());
        pstmt.setString(3, board.getBwriter());
        pstmt.executeUpdate();

        pstmt.close();
    } catch (SQLException e) {
        e.printStackTrace();
        exit();
    }
}
//목록 함수 호출
list();
}
```



# 게시판 구현

## ◆ 게시판 화면 - Read 메뉴 개발

[게시물 목록]

no	writer	date	title
2	cloud	2023-10-17	notebook
1	today	2023-10-16	smartphone

1.Create | 2.Read | 3.Clear | 4.Exit

선택: 2

[게시물 읽기]

글번호 입력: 2

\*\*\*\*\*

번호: 2

제목: notebook

내용: LG 노트북입니다.

작성자: cloud

작성일: 2023-10-17

\*\*\*\*\*



# 게시판 구현

## ◆ 게시판 화면 - Read 메뉴 개발

```
public void read() {  
    //입력 받기  
    System.out.println("[게시물 읽기]");  
    System.out.print("글번호 입력: ");  
    int bno = Integer.parseInt(scanner.nextLine());  
  
    try {  
        String sql = "SELECT bno, btitle, bcontent, bwriter, bdate"  
            + " FROM board WHERE bno = ?";  
        pstmt = conn.prepareStatement(sql);  
        pstmt.setInt(1, bno);  
        ResultSet rs = pstmt.executeQuery();  
        if(rs.next()) {  
            Board board = new Board();  
            board.setBno(rs.getInt("bno"));  
            board.setBtitle(rs.getString("btitle"));  
            board.setBcontent(rs.getString("bcontent"));  
            board.setBwriter(rs.getString("bwriter"));  
            board.setBdate(rs.getDate("bdate"));  
        }  
    }  
}
```





# 게시판 구현

## ◆ 게시판 화면 - Read 메뉴 개발

```
//게시글 출력
System.out.println("*****");
System.out.println("번호: " + board.getBno());
System.out.println("제목: " + board.getBtitle());
System.out.println("내용: " + board.getBcontent());
System.out.println("작성자: " + board.getBwriter());
System.out.println("작성일: " + board.getBdate());
System.out.println("*****");
}
rs.close();
pstmt.close();
} catch (SQLException e) {
    e.printStackTrace();
    exit();
}
//목록 함수 호출
list();
}
```



# 게시판 구현

## ◆ 게시판 화면 - Clear 메뉴 개발

```
-----  
1.Create | 2.Read | 3.Clear | 4.Exit  
선택: 3
```

[전체 게시물 삭제]

```
-----  
1.Ok | 2.Cancel  
선택: 1  
[게시물 목록]
```

```
-----  
no  writer      date      title  
-----
```



# 게시판 구현

## ◆ 게시판 화면 - Clear 메뉴 개발

```
public void clear() {  
    System.out.println("[전체 게시물 삭제]");  
    System.out.println("-----");  
    System.out.println("1.Ok | 2.Cancel");  
    System.out.print("선택: ");  
    String menuNo = scanner.nextLine();  
    if(menuNo.equals("1")) {  
        String sql = "TRUNCATE TABLE board";  
        try {  
            pstmt = conn.prepareStatement(sql);  
            pstmt.executeUpdate();  
            pstmt.close();  
        } catch (SQLException e) {  
            e.printStackTrace();  
            exit();  
        }  
    }  
    //목록 함수 호출  
    list();  
}
```



# 게시판 구현

## ◆ 게시판 화면 - Clear 메뉴 개발

```
if(menuNo.equals("1")) {  
    try {  
        String sql = "TRUNCATE TABLE board"; //모든 데이터 삭제  
        pstmt = conn.prepareStatement(sql);  
        pstmt.executeUpdate();  
  
        //데이터 생성시 글번호 이전 번호 초기화하여 1부터 시작하기  
        sql = "DROP SEQUENCE seq_bno"; //시퀀스 삭제  
        pstmt = conn.prepareStatement(sql);  
        pstmt.executeUpdate();  
  
        sql = "CREATE SEQUENCE seq_bno"; //시퀀스 생성  
        pstmt = conn.prepareStatement(sql);  
        pstmt.executeUpdate();  
  
        pstmt.close();  
    } catch (SQLException e) {
```



# 게시판 구현

## ◆ 게시판 화면 - Exit 메뉴 개발

[게시물 목록]

no	writer	date	title
2	cloud	2023-10-17	notebook
1	today	2023-10-16	smartphone

1.Create | 2.Read | 3.Clear | 4.Exit

선택: 4

\*\*\* 게시판을 종료합니다. \*\*\*



# 게시판 구현

## ◆ 게시판 화면 - Exit 메뉴 개발

```
public void exit() {  
    if(conn != null) {  
        try {  
            conn.close();  
        } catch (SQLException e) {  
        }  
    }  
    System.out.println("*** 게시판을 종료합니다. ***");  
    System.exit(0);  
}
```



# 게시판 구현

## ◆ Read 화면 - Update 메뉴 개발

```
번호: 2
제목: 날씨가 쌀쌀해요
내용: 날이 춥네요.. 감기 조심하세요
작성자: cloud120
작성일: 2023-10-17
*****

1.Update | 2.Delete | 3.List
선택: 1

[수정할 내용 입력]
제목: 좋은 아침입니다.
내용: 날씨가 쌀쌀해요.. 감기 조심하세요
작성자: cloud100
1.Ok | 2.Cancel
선택: 1
[게시물 목록]

-----
no  writer      date      title
-----
2   cloud100    2023-10-17  좋은 아침입니다.
1   todav10     2023-10-17  가입인사 드려요
```



## ◆ Read() 메서드 - Update 메뉴 개발

```
System.out.println("*****");
System.out.println("번호: " + board.getBno());
System.out.println("제목: " + board.getBtitle());
System.out.println("내용: " + board.getBcontent());
System.out.println("작성자: " + board.getBwriter());
System.out.println("작성일: " + board.getBdate());
System.out.println("*****");

//수정, 삭제, 목록가기 메뉴 생성
System.out.println("1.Update | 2.Delete | 3.List");
System.out.print("선택: ");
String menuNo = scanner.nextLine();
System.out.println();

if(menuNo.equals("1")) {
    update(board);
} else if(menuNo.equals("2")) {
    delete(board);
}
}
rs.close();
pstmt.close();
```





# 게시판 구현

## ◆ Read() 메서드 - Update 메뉴 개발

```
public void update(Board board) {
    System.out.println("[수정할 내용 입력]");
    System.out.print("제목: ");
    board.setBtitle(scanner.nextLine());
    System.out.print("내용: ");
    board.setBcontent(scanner.nextLine());
    System.out.print("작성자: ");
    board.setBwriter(scanner.nextLine());

    //확인, 취소 메뉴 생성
    System.out.println("1.Ok | 2.Cancel");
    System.out.print("선택: ");
    String menuNo = scanner.nextLine();
    if(menuNo.equals("1")) {
        try {
            String sql = "UPDATE board "
                + "SET btitle = ?, bcontent = ?, bwriter = ? "
                + "WHERE bno = ?";
```



# 게시판 구현

## ◆ Read() 메서드 - Update 메뉴 개발

```
        pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, board.getBtitle());
        pstmt.setString(2, board.getBcontent());
        pstmt.setString(3, board.getBwriter());
        pstmt.setInt(4, board.getBno());
        pstmt.executeUpdate();
        pstmt.close();
    } catch (SQLException e) {
        e.printStackTrace();
        exit();
    }
}
//게시물 목록 출력
list();
}
```



# 게시판 구현

## ◆ Read() 메서드 - Delete 메뉴 개발

```
1.Create | 2.Read | 3.Clear | 4.Exit
```

```
선택: 2
```

```
[게시물 읽기]
```

```
글번호 입력: 1
```

```
*****
```

```
번호: 1
```

```
제목: 가입인사 드려요
```

```
내용: 안녕하세요~ 만나서 반갑습니다.
```

```
작성자: today10
```

```
작성일: 2023-10-17
```

```
*****
```

```
1.Update | 2.Delete | 3.List
```

```
선택: 2
```

```
[게시물 목록]
```

```
-----  
no  writer      date      title  
-----  
2   cloud100    2023-10-17  좋은 아침입니다.
```



# 게시판 구현

## ◆ Read() 메서드 - Delete 메뉴 개발

```
public void delete(Board board) {  
    //System.out.println("[게시글 삭제]");  
    try {  
        String sql = "DELETE FROM board WHERE bno = ?";  
        pstmt = conn.prepareStatement(sql);  
        pstmt.setInt(1, board.getBno());  
        pstmt.executeUpdate();  
        pstmt.close();  
    } catch (SQLException e) {  
        e.printStackTrace();  
        exit();  
    }  
    //게시물 목록 출력  
    list();  
}
```

