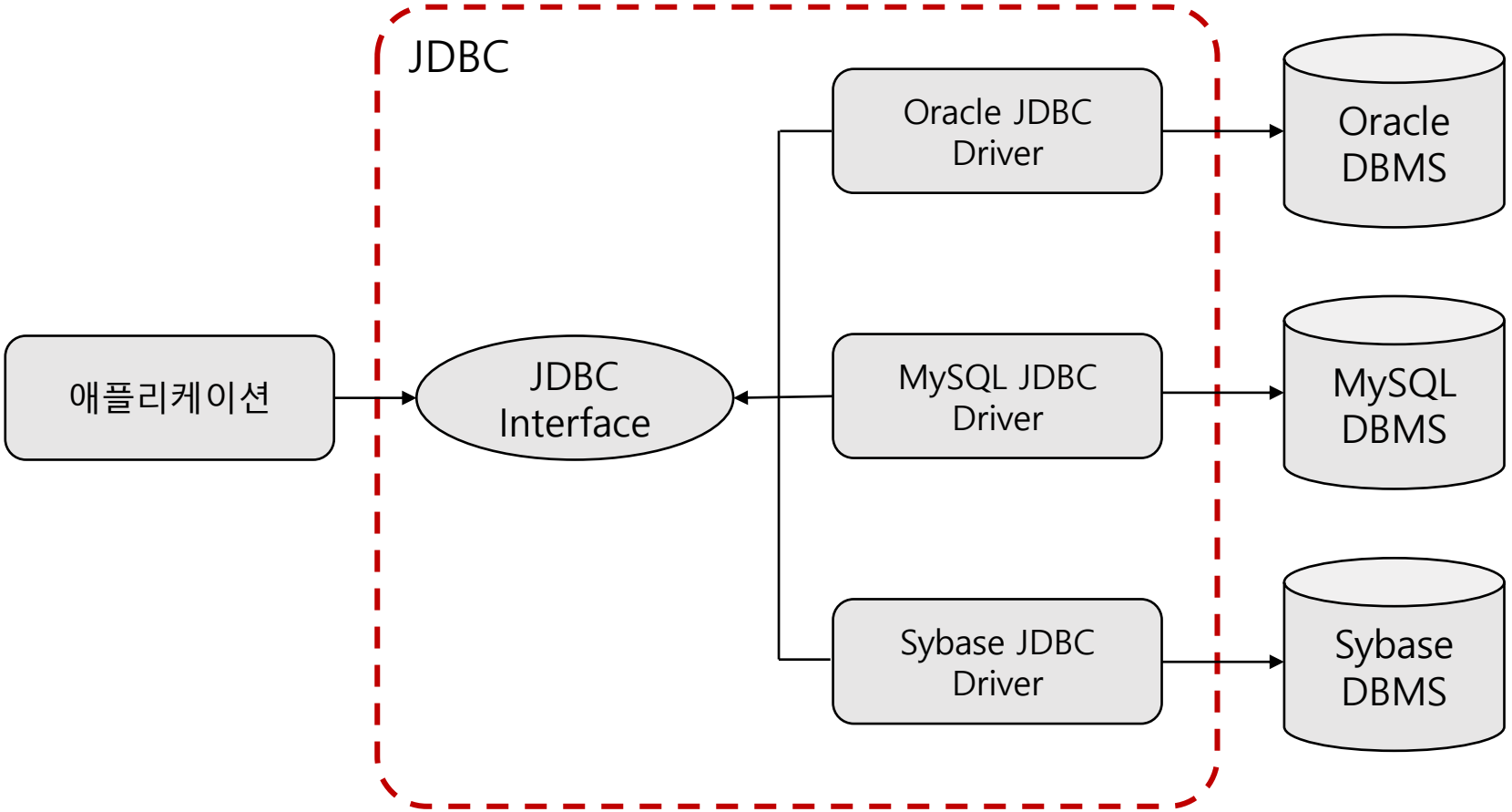
JDBC

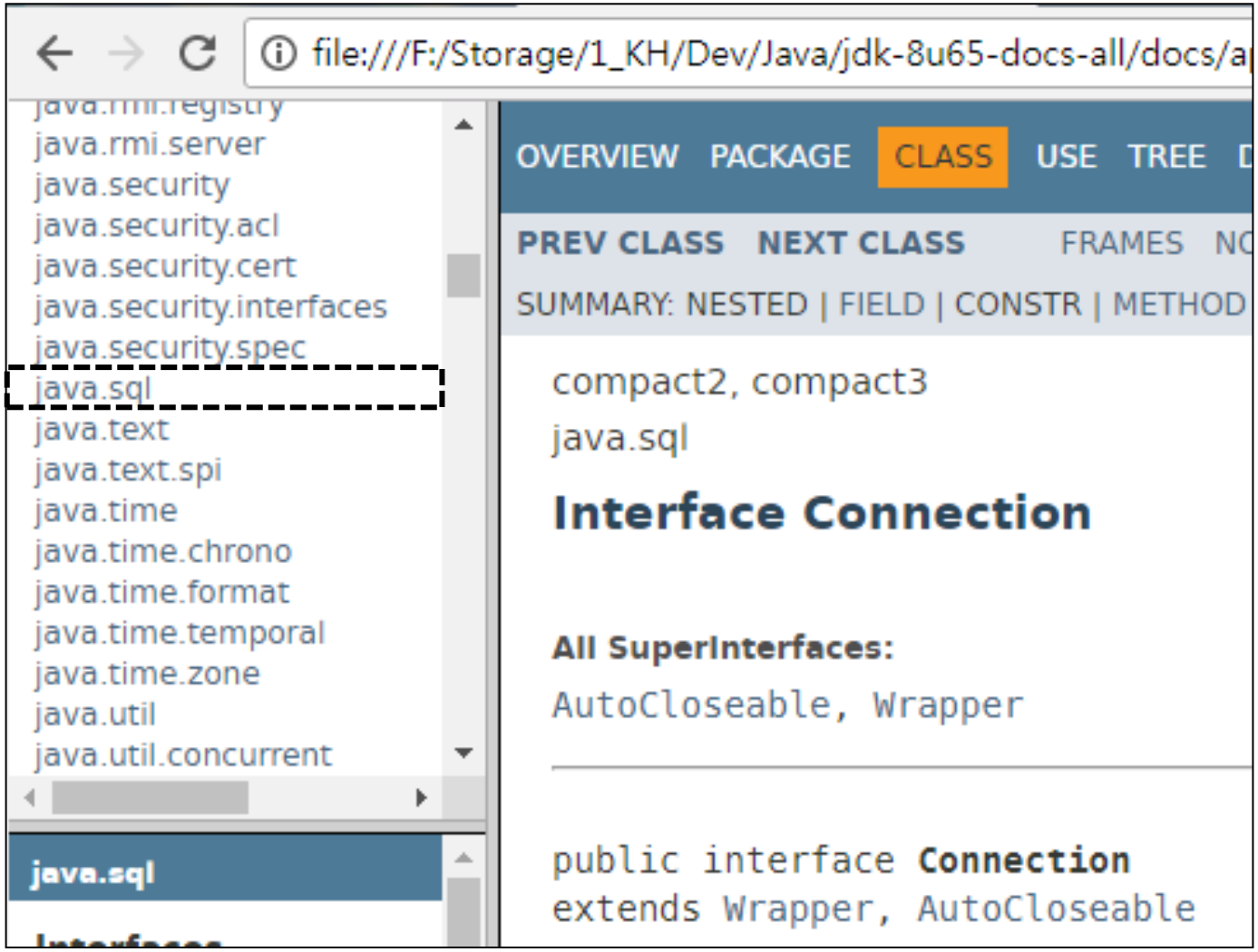
JDBC(Java DataBase Connectivity)

자바에서 데이터베이스에 접근할 수 있게 해주는 Programming API



JDBC(Java DataBase Connectivity)

✓ java.sql 패키지



▶ OJDBC

오라클에서 제공하는 오라클 DB와 자바가 연결하기 위한 라이브러리

※ 두 가지 방법 중 한가지 선택

✓ 홈페이지 다운로드

https://www.oracle.com

Resources -> Software Downloads

-> Drivers and Utilities -> JDBC Drivers

-> Oracle Database 11g Release 2 (11.2.0.4) drivers -> ojdbc6.jar

✓ 오라클에서 복사

- **Express 버전**

  C:₩oraclexe₩app₩oracle₩product₩11.2.0₩server₩jdbc₩lib₩ojdbc6.jar

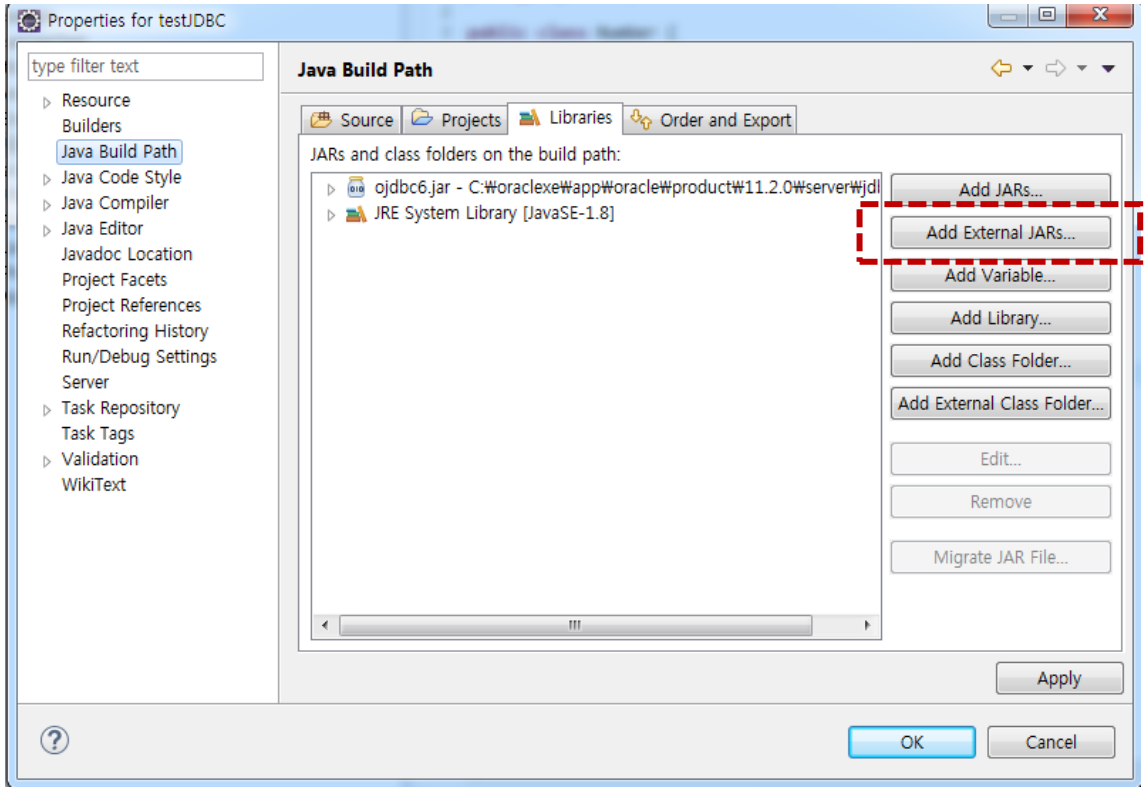
- **Enterprise 버전**

  C:₩app₩사용자계정₩product₩11.2.0₩dbhome_1₩jdbc₩lib₩ojbdc6.jar

▶ Library 등록

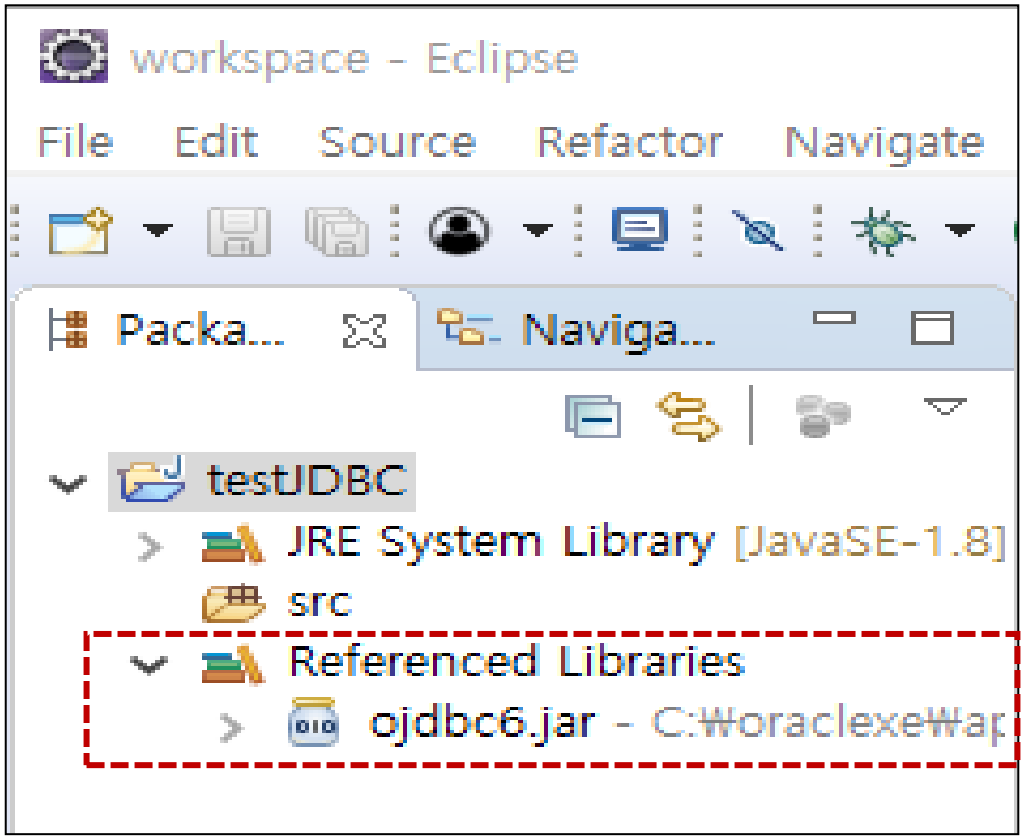
✓OJDBC Library 등록

프로젝트에서 ojdbc를 사용할 수 있도록 Library를 등록해 줘야 함
프로젝트명 우 클릭 -> properties클릭 -> Java Build Path
-> Libraries -> Add External JARs.. 클릭 후 ojdbc6파일 선택 -> apply클릭



▶ Library 등록

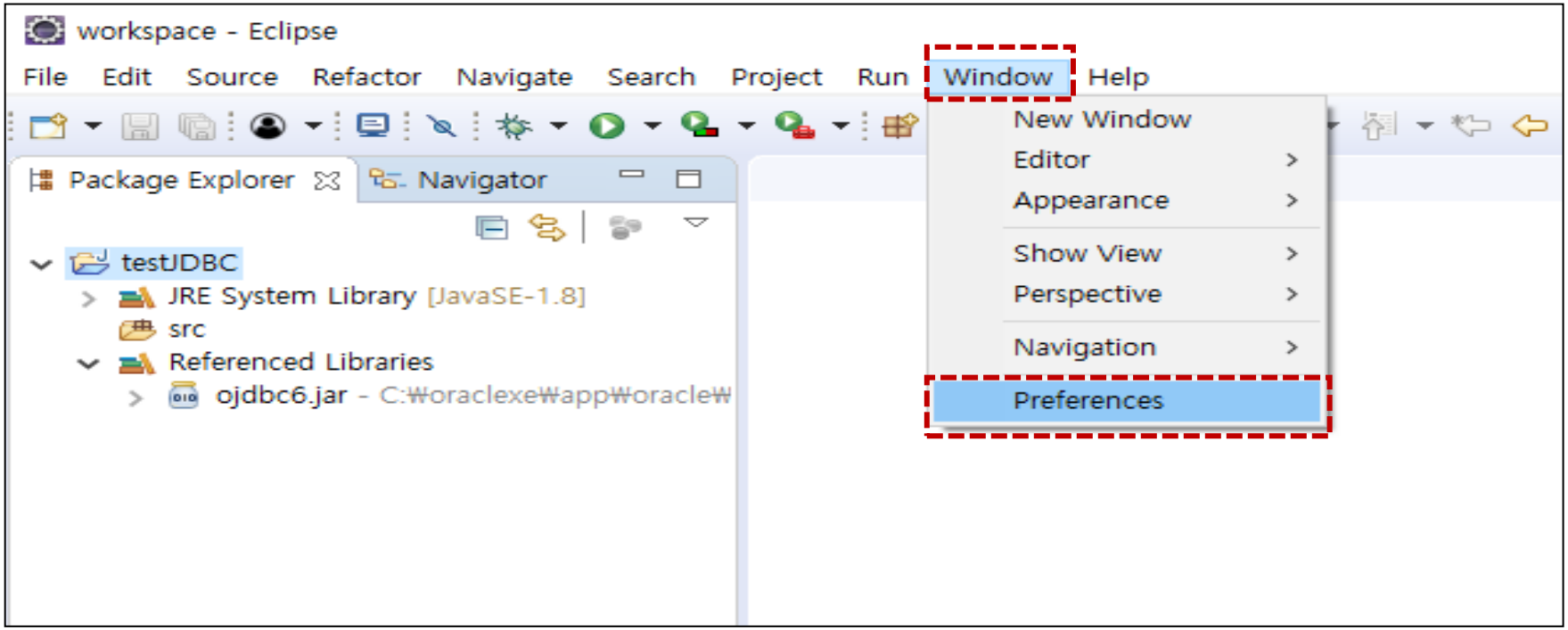
✓OJDBC Library 등록



* package Explorer에 Referenced Libraries가 생기고 ojdbc6.jar가 생성됨

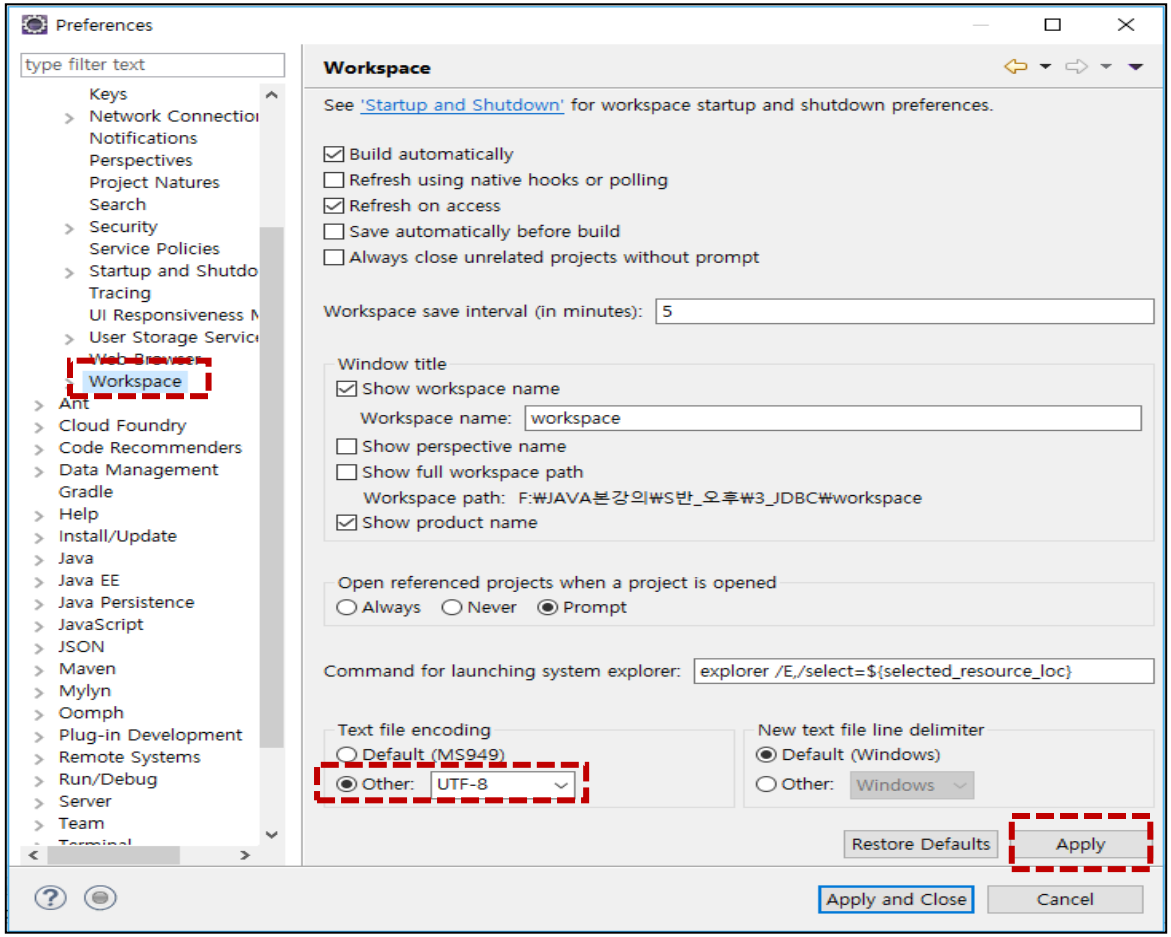
▶ Character Set 설정

문자 인코딩 방식이 맞지 않으면 데이터 베이스 연동 시 해당 문자가 제대로 출력되지 않아 이클립스 작업 파일에 대한 문자셋을 일치시켜야 함(UTF-8)



Character Set 설정

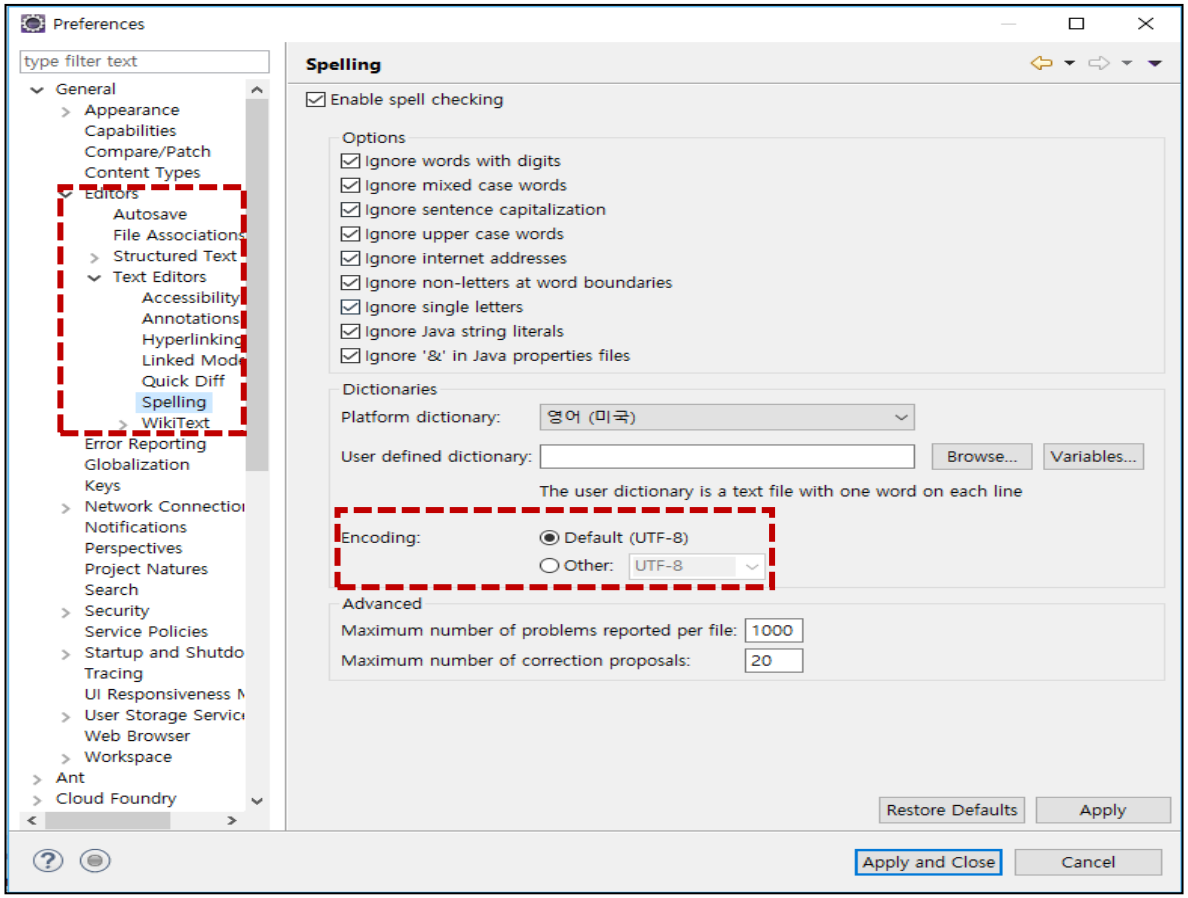
✓ 기본 문자 인코딩



* General – Workspace – Text file encoding – other에서 UTF-8 선택

Character Set 설정

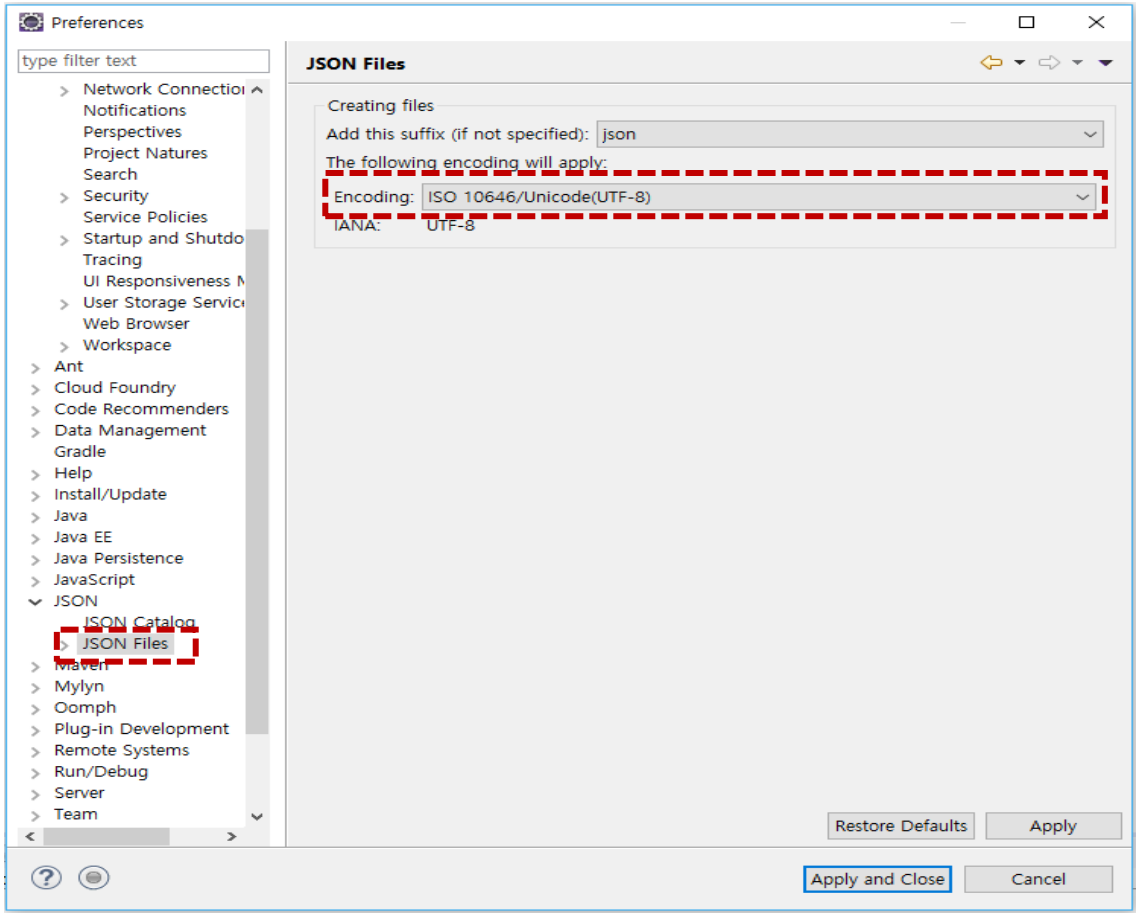
✓ 기본 문자 인코딩



* General – Editors – Text Editors – Spelling UTF-8 변경

Character Set 설정

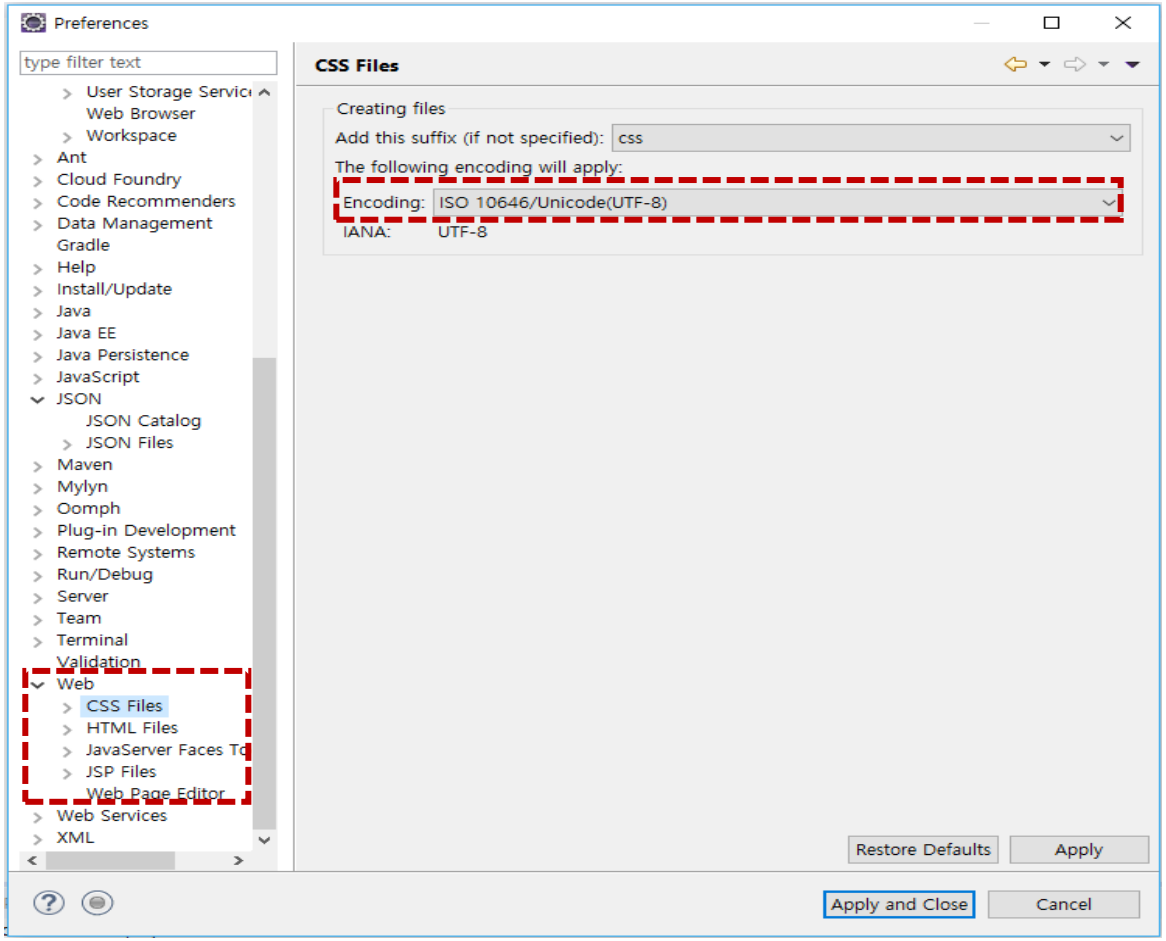
✓ JSON 인코딩



* JSON – JSON Files – Encoding – ISO 10646/Unicode(UTF-8) 선택

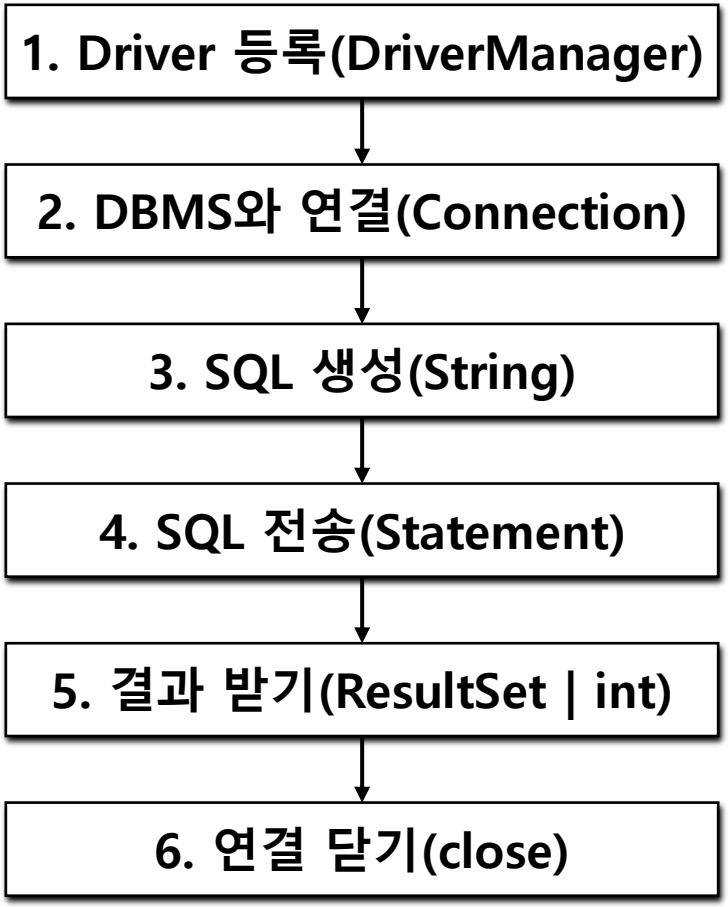
Character Set 설정

✓ Web 인코딩



* CSS Files, HTML Files, JSP Files도 수정

▶ JDBC Coding 절차



▶ JDBC 사용 객체

✓Class.forName() 메소드

문자열 형태로 전달된 클래스를 JVM으로 로드해주는 기능

Oracle JDBC 드라이버를 지정해 로드하여 DriverManager에 등록이 되도록 하는 코드

JDBC 4.0 이후로는 메소드를 호출하지 않아도 자동으로 로드 됨

Class.forName("oracle.jdbc.driver.OracleDriver");

✓ DriverManager

JDBC드라이버를 통하여 Connection을 만드는 역할

직접 객체 생성이 불가능하고 getConnection() 메소드를 사용하여 객체 생성 가능

DriverManager.getConnection("JDBC형식 URL", "계정", "비밀번호");

▶ JDBC 사용 객체

✓ JDBC 형식 URL

jdbc:oracle:thin:**[hostname][:port]:dbname**

✓ Connection

특정 데이터 원본과 연결된 Connection

Connection객체의 createStatement() 메소드를 호출하여 Statement 객체를 생성

SQL문장을 실행시키기 전에 우선 Connection 객체가 있어야 함

<Connection객체명>.createStatement();

객체 사용 완료 후 반드시 닫아 줘야 함

<Connection객체명>.close();

▶ JDBC 사용 객체

✓ Statement

데이터베이스 연결을 한 후 실제 SQL문을 수행하기 위한 객체

Connection클래스의 createStatement() 메소드를 호출하여 생성됨

생성된 Statement객체로 질의문장을 String객체에 담아 인자로 전달

executeQuery(),executeUpdate() 메소드를 호출하여 SQL 질의 수행

<Statement객체명>.executeQuery(<SQL문>)

<Statement객체명>.executeUpdate(<SQL문>)

객체 사용 완료 후 반드시 닫아 줘야 함

<Statement객체명>.close();

▶ JDBC 사용 객체

✓executeQuery() 메소드

ResultSet을 반환하는 SQL 문에 사용하는 메소드로 주로 SELECT문을 이용하는 조회에 사용

✓executeUpdate() 메소드

SELECT를 제외한 DML(INSERT, UPDATE, DELETE)과 DDL(CREATE, DROP, ALTER) 구문을 수행하기 위한 메소드로 DML을 수행한 경우 영향 받은 행의 수를 반환하고, DDL을 수행한 경우 0을 반환

▶ JDBC 사용 객체

KH정보교육원

✓ 예시

```
try{
        String query = "SELECT ID, LAST_NAME FROM EMP";
        stmt = conn.createStatement();
        rset = stmt.executeQuery(query);
} catch(SQLException e){
        e.printStackTrace();
}
```

▶ JDBC 사용 객체

KH정보교육원

✓ PreparedStatement

데이터베이스 연결을 한 후 실제 SQL문을 수행하기 위한 객체
Connection클래스의 prepareStatement() 메소드를 호출하여 생성됨
SQL문장이 미리 컴파일 되고 실행 시간 동안 인수 값을 위한 공간을 확보한다는 점에서 Statement와 다름
각 인수에 대해 위치 홀더(?)를 사용해 SQL문장을 정의할 수 있게 함

✓위치 홀더(?)

SQL문장에 나타나는 토큰으로 SQL구문이 실행되기 전에 실제 값으로 대체되어야 함

<PreparedStatement객체명>.setString(위치,값)

▶ JDBC 사용 객체

KH정보교육원

✓ 예시

```
try{
        String query = "INSERT INTO MEMBER VALUES(?, ?)";
        pstmt = conn.prepareStatement(query);
        pstmt.setString(1, id);
        pstmt.setString(2, password);
        pstmt.excuteUpdate();
} catch(SQLException e){
        e.printStackTrace();
}
```

▶ JDBC 사용 객체

✓ ResultSet

SELECT문을 사용한 질의 성공 시 반환되는 객체

Statement로 SQL질의에 의해 생성된 테이블을 담아서 사용

커서(cursor)로 특정 행에 대한 참조를 조작

<ResultSet객체명>.next();

커서가 위치한 행의 데이터 사용

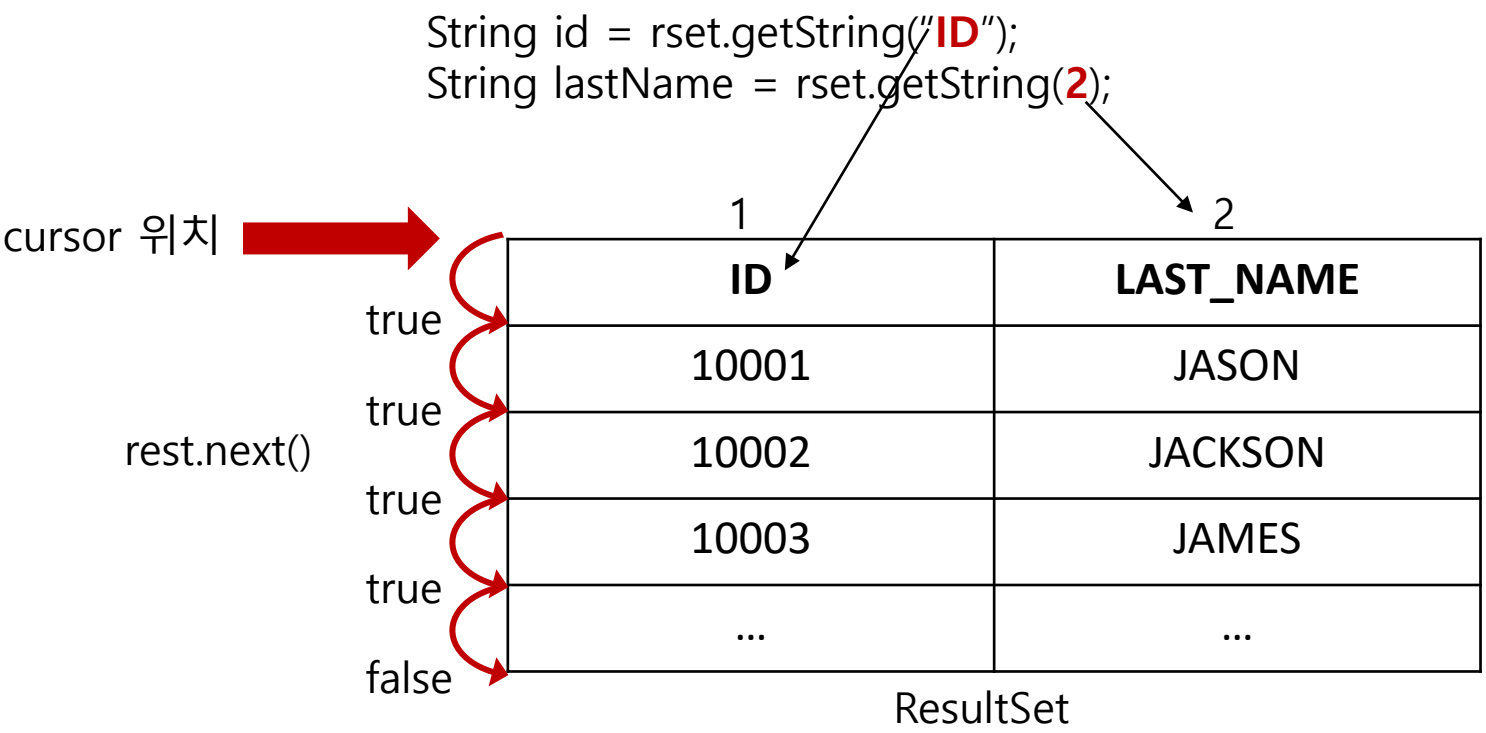
<ResultSet객체명>.getString(<컬럼명|인덱스>);

객체 사용 완료 후 반드시 닫아 줘야 함

<ResultSet객체명>.close();

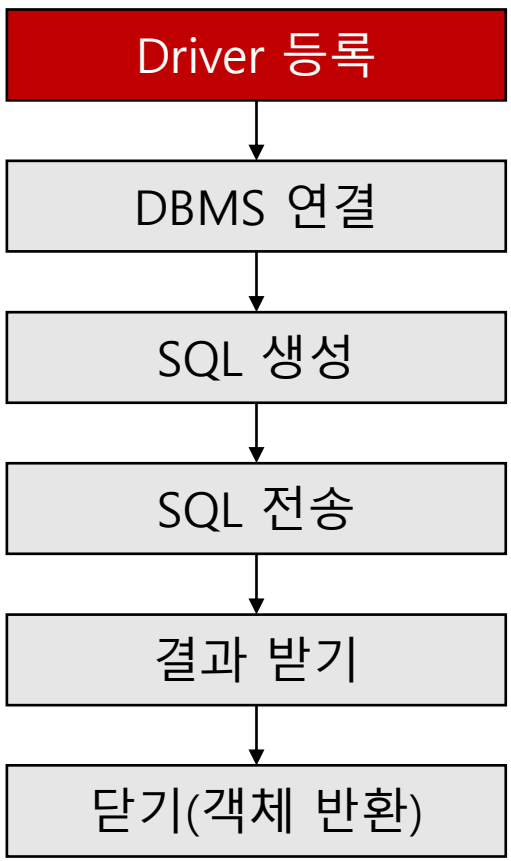
JDBC 사용 객체

✓ ResultSet



▶ JDBC 코딩 절차

✓ DriverManager에 해당 DBMS Driver 등록



DriverManager

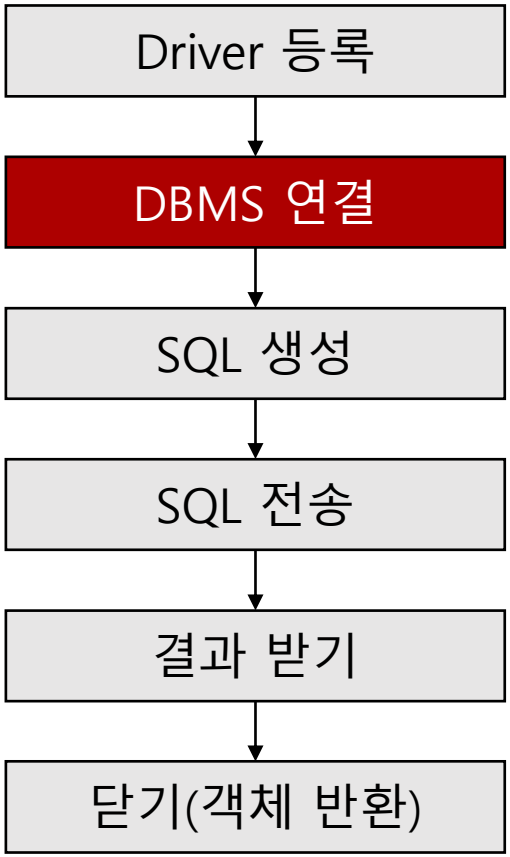
Class.forName("oracle.jdbc.driver.OracleDriver");

cf) 다른 DB설정
MSSQL :
Class.forName("com.Microsoft.jdbc.sqlserver.SQLServerDriver");
MySQL :
Class.forName("org.gjt.mm.mysql.Driver");

* 반드시 ClassNotFoundException 처리를 해야 함

▶ JDBC 코딩 절차

✓ 해당 Driver로부터 Connection instance 획득



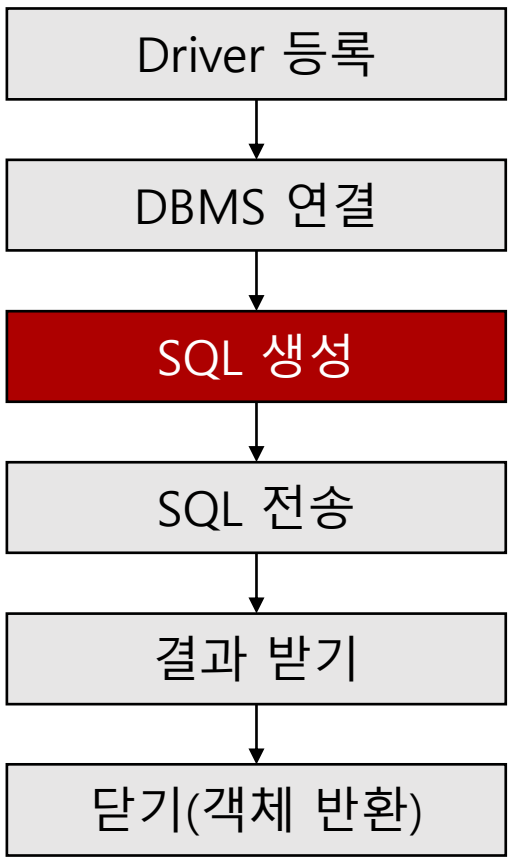
DriverManager Connection

```
Connection conn = DriverManager.getConnection(
                "jdbc:oracle:thin:@127.0.0.1:1521:XE",
                "student", "student");
```

* 반드시 SQLException처리를 해야 함

▶ JDBC 코딩 절차

✓ 실행할 SQL문을 문자열형태로 작성



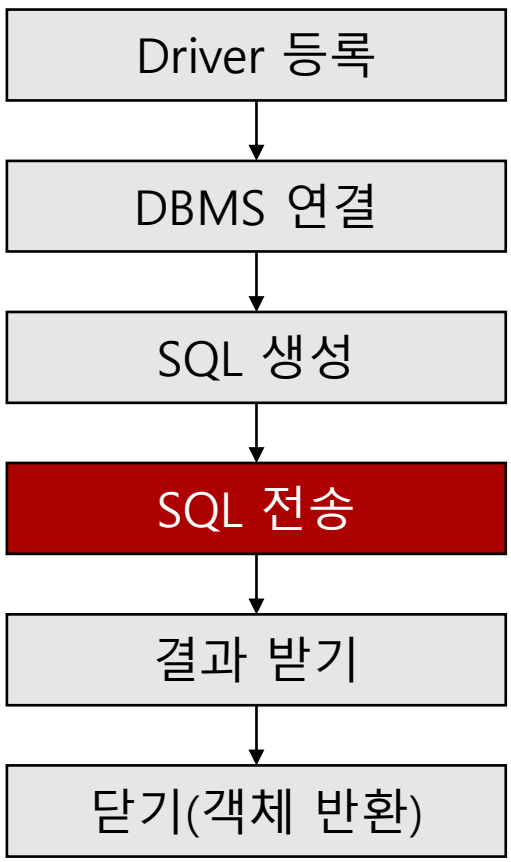
DriverManager Connection

```
String query = "SELECT ID, LAST_NAME FROM EMP";
```

```
String query = "UPDATE EMP SET LAST_NAME = 'KIM'"
             + " WHERE ID = '10000'";
```

▶ JDBC 코딩 절차

✓ 해당 Connection으로부터 Statement instance 획득
✓ Statement method를 이용하여 SQL문 실행



DriverManager Connection Statement

```
Statement stmt = conn.createStatement();

ResultSet rset = stmt.executeQuery(query);
```

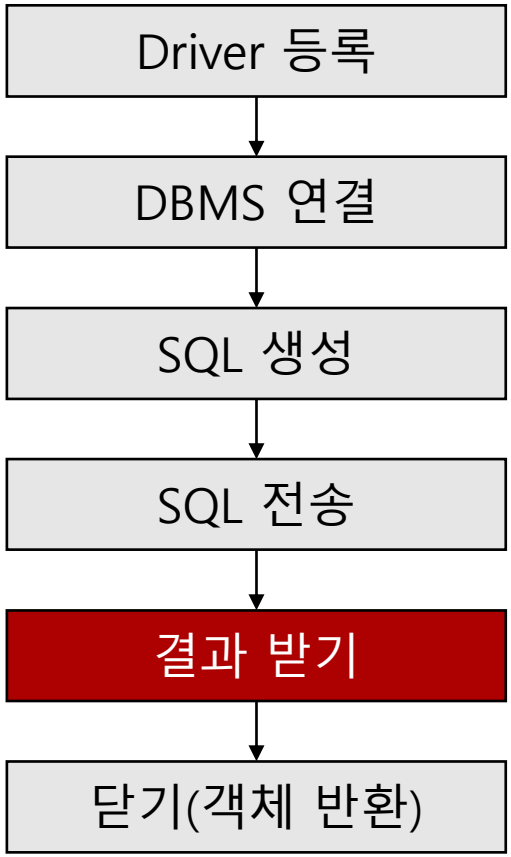
```
Statement stmt = conn.createStatement();

int result = stmt.executeUpdate(query);
```

* 반드시 SQLException처리를 해야 함

▶ JDBC 코딩 절차

KH정보교육원

✓ 실행결과를 ResultSet 혹은 int형 변수로 받아서 처리



DriverManager → Connection → Statement → ResultSet

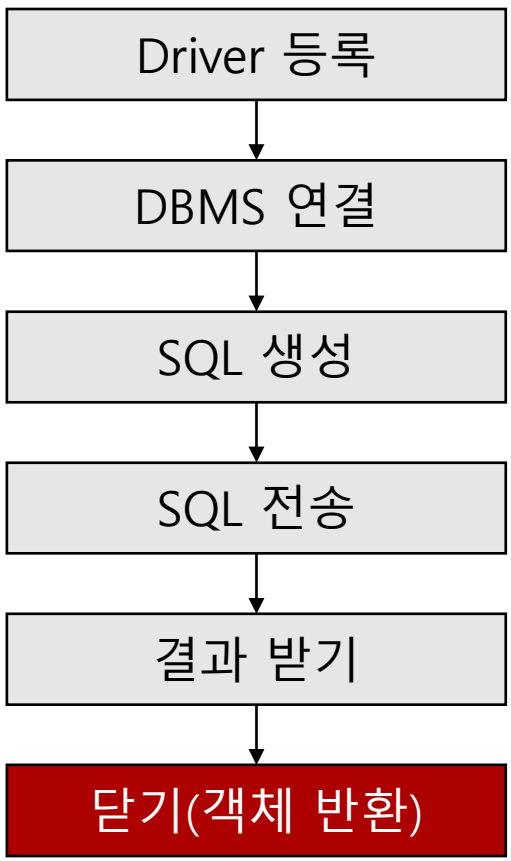
```
ResultSet rset = stmt.executeQuery(query);

while(rset.next()){
        System.out.println(rset.getString("ID") + "₩t" +
        rset.getString(2));
}
```

```
int result = stmt.executeUpdate(query);
```

* 반드시 SQLException처리를 해야 함

▶ JDBC 코딩 절차

✓ 사용 완료된 객체 반납 처리



DriverManager → Connection → Statement → ResultSet

```
rset.close(); //ResultSet 사용한 경우 반납처리
stmt.close();
conn.close();
```

* 반드시 SQLException처리를 해야 함