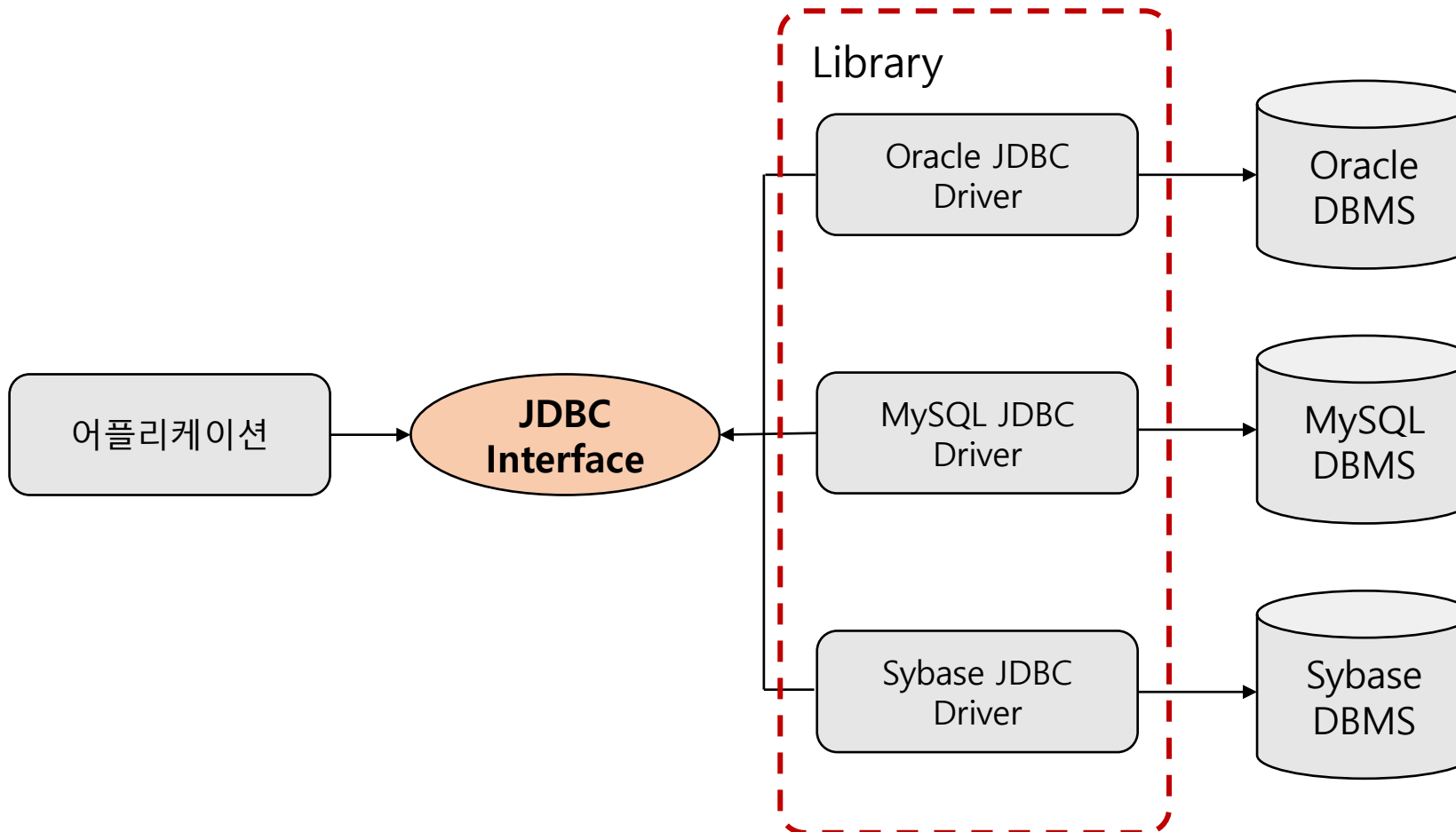


# JDBC

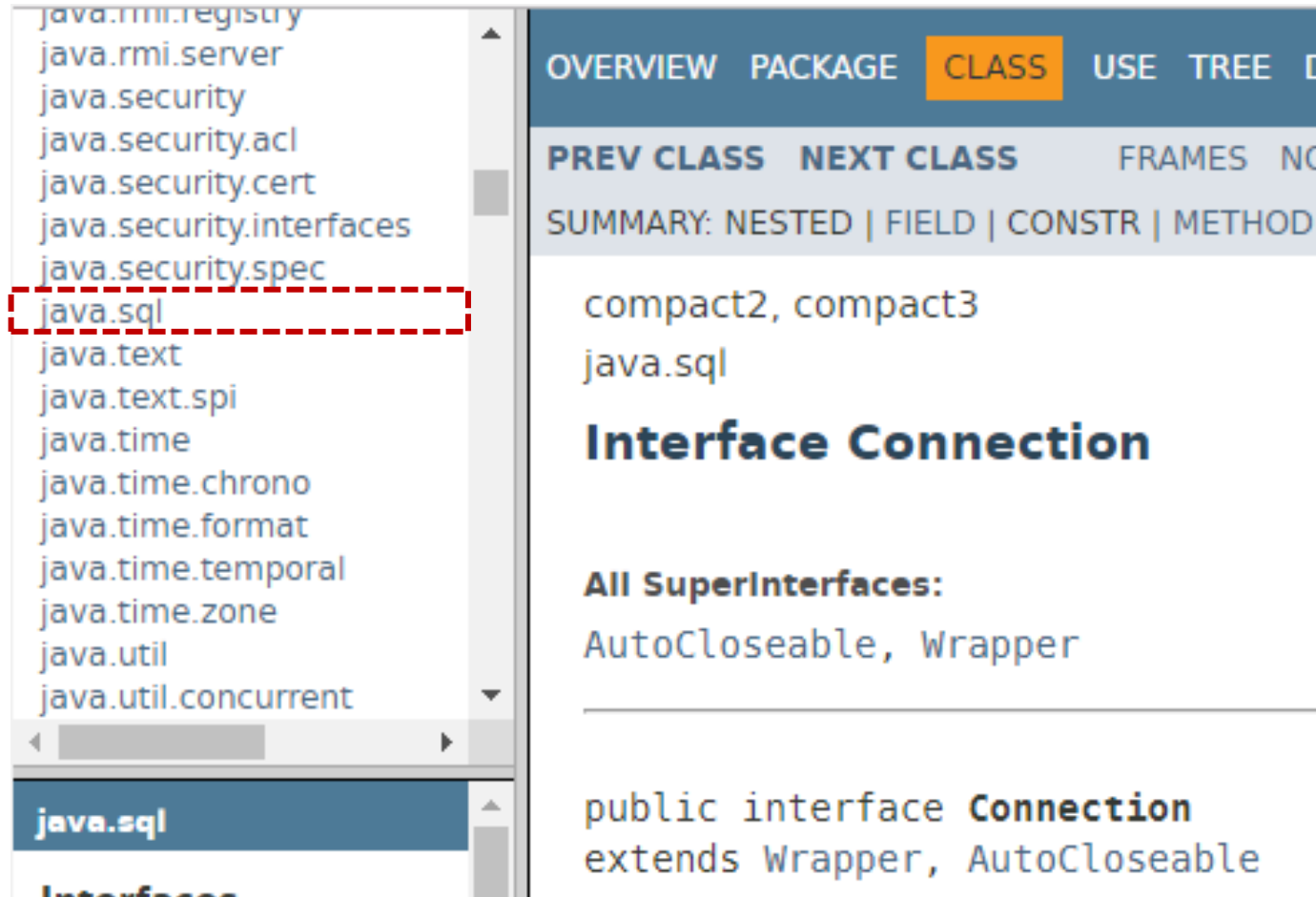
# ▶ JDBC(Java DataBase Connectivity)

Java에서 DB에 접근할 수 있게 해주는 Java Programming API



# ▶ JDBC(Java DataBase Connectivity)

## ✓ java.sql 패키지



java.rmi.registry  
java.rmi.server  
java.security  
java.security.acl  
java.security.cert  
java.security.interfaces  
java.security.spec  
**java.sql**  
java.text  
java.text.spi  
java.time  
java.time.chrono  
java.time.format  
java.time.temporal  
java.time.zone  
java.util  
java.util.concurrent

OVERVIEW PACKAGE **CLASS** USE TREE D  
PREV CLASS NEXT CLASS FRAMES NC  
SUMMARY: NESTED | FIELD | CONSTR | METHOD

compact2, compact3  
java.sql

### Interface Connection

**All SuperInterfaces:**  
AutoCloseable, Wrapper

---

public interface **Connection**  
extends Wrapper, AutoCloseable

# ▶ OJDBC

## ✓ OJDBC란?

오라클에서 제공하는 오라클 DB와 자바가 연결하기 위한 라이브러리  
-> **Oracle JDBC Driver** 제공

## ✓ OJDBC 다운로드

메이븐 레파지토리에서 다운로드( <https://mvnrepository.com/> )

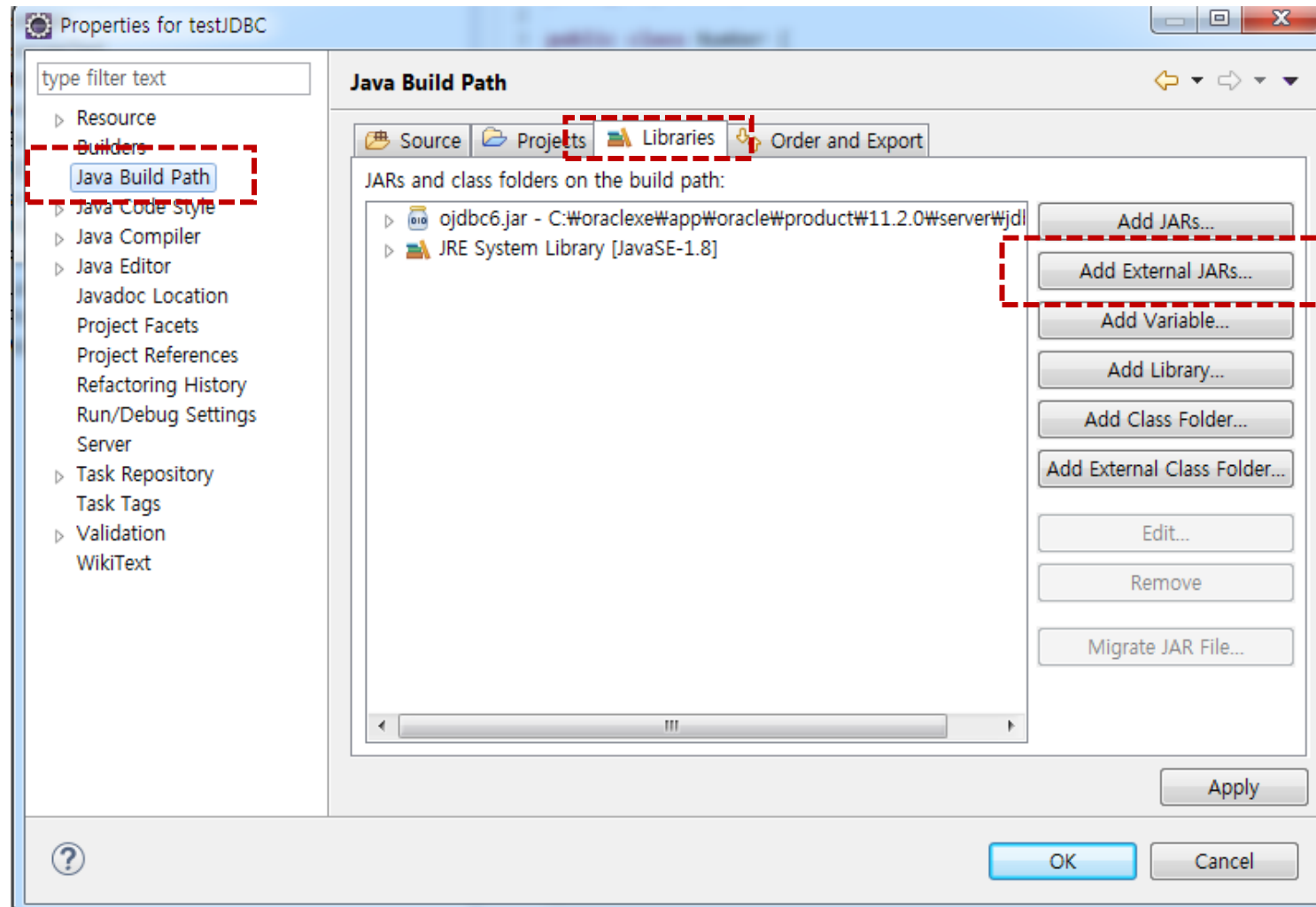
**ojdbc11.jar**( JDK11 ~ 17포함)

**ojdbc8.jar**( JDK8 ~ 15포함)

(알맞은 버전을 검색해서 다운로드)

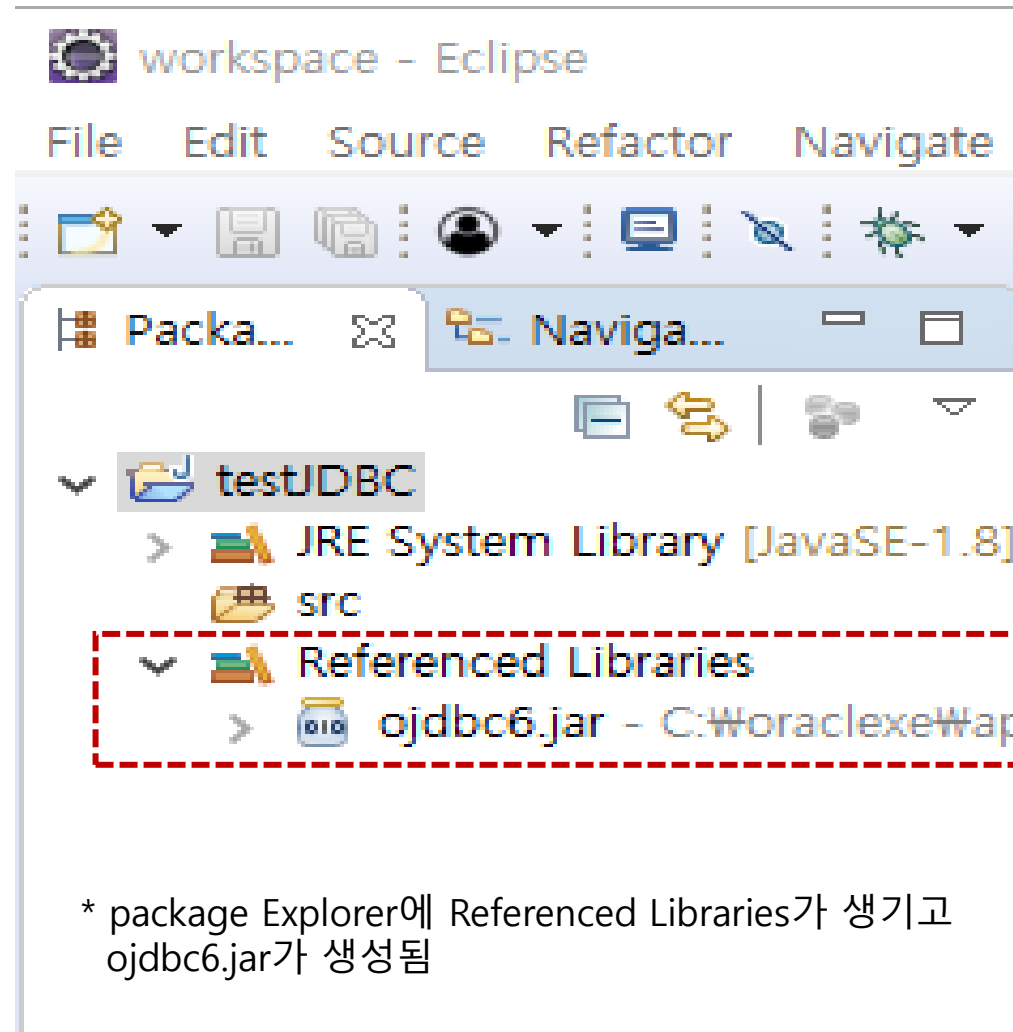
# ▶ Library 등록 방법

✓ Java Project 우클릭 -> Properties에서 등록



## ▶ Library 등록

### ✓ OJDBC Library 등록 확인



## ▶ JDBC 사용 객체

### ✓ DriverManager

데이터 원본에 JDBC드라이버를 통하여 커넥션을 만드는 역할

Class.forName() 메소드를 통해 생성되며 반드시 예외처리를 해야 함

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

### ✓ Connection

특정 데이터 원본과 연결된 커넥션을 나타내며 Statement객체를 생성할 때도

Connection객체를 사용하여 createStatement() 메소드를 호출하여 생성

SQL문장을 실행시키기 전에 우선 Connection객체가 있어야 함

## ▶ JDBC 사용 객체

### ✓ Statement

Connection객체에 의해 프로그램에 리턴되는 객체에 의해 구현되는 일종의 메소드 집합 정의  
Connection클래스의 `createStatement()` 메소드를 호출하여 얻어지며  
생성된 Statement객체로 질의문장을 String객체에 담아 인자로 전달하여  
`executeQuery()` 메소드를 호출하여 SQL질의 수행

### ✓ 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

### ✓ PreparedStatement

Connection객체의 `prepareStatement()` 메소드를 사용하여 객체 생성  
SQL문장이 미리 컴파일 되고 실행 시간동안 인수 값을 위한 공간을 확보한다는 점에서  
Statement와 다름  
각각의 인수에 대해 위치 홀더(?)를 사용하여 SQL문장을 정의할 수 있게 함

### ✓ 예시

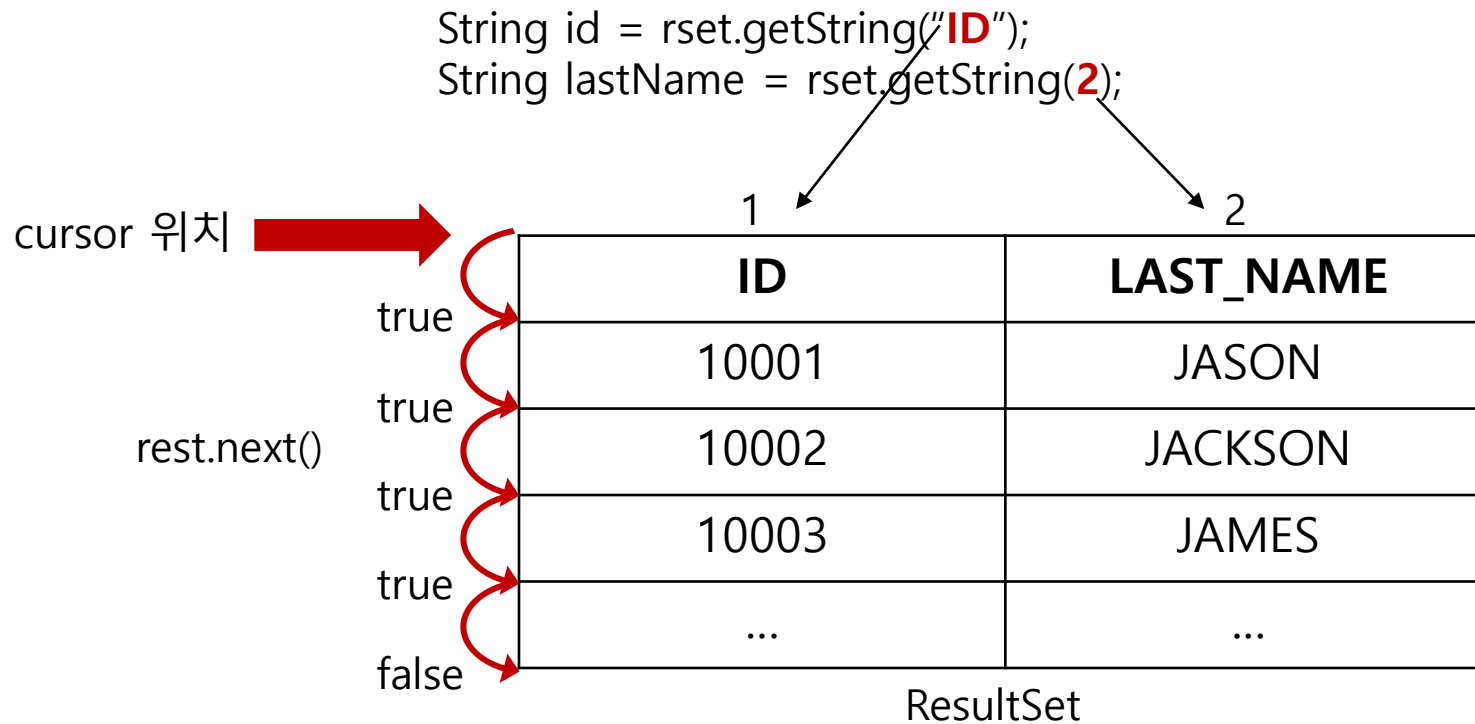
```
try{  
    String query = "INSERT INTO MEMBER VALUES(?, ?)";  
    pstmt = conn.prepareStatement(query);  
    pstmt.setString(1, id);  
    pstmt.setString(2, password);  
} catch(SQLException e){  
    e.printStackTrace();  
}
```

# ▶ JDBC 사용 객체

## ✓ ResultSet

SELECT문을 사용한 질의 성공 시 Result Set 반환

SQL질의에 의해 생성된 테이블을 담고 있으며 커서(cursor)로 특정 행에 대한 참조 조작



# ▶ JDBC 코딩 절차

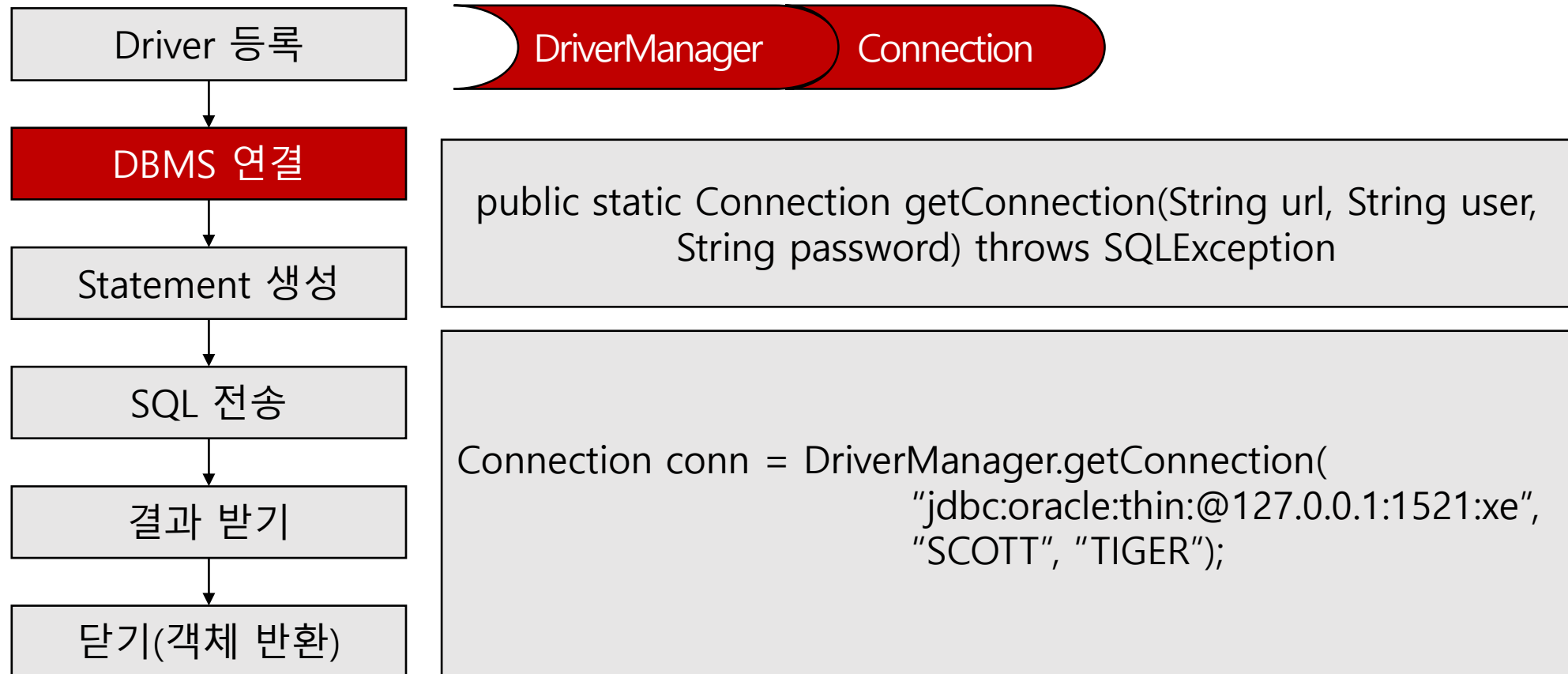
## ✓ DriverManager에 해당 DBMS Driver 등록



\* 반드시 ClassNotFoundException 처리를 해야 함

# ▶ JDBC 코딩 절차

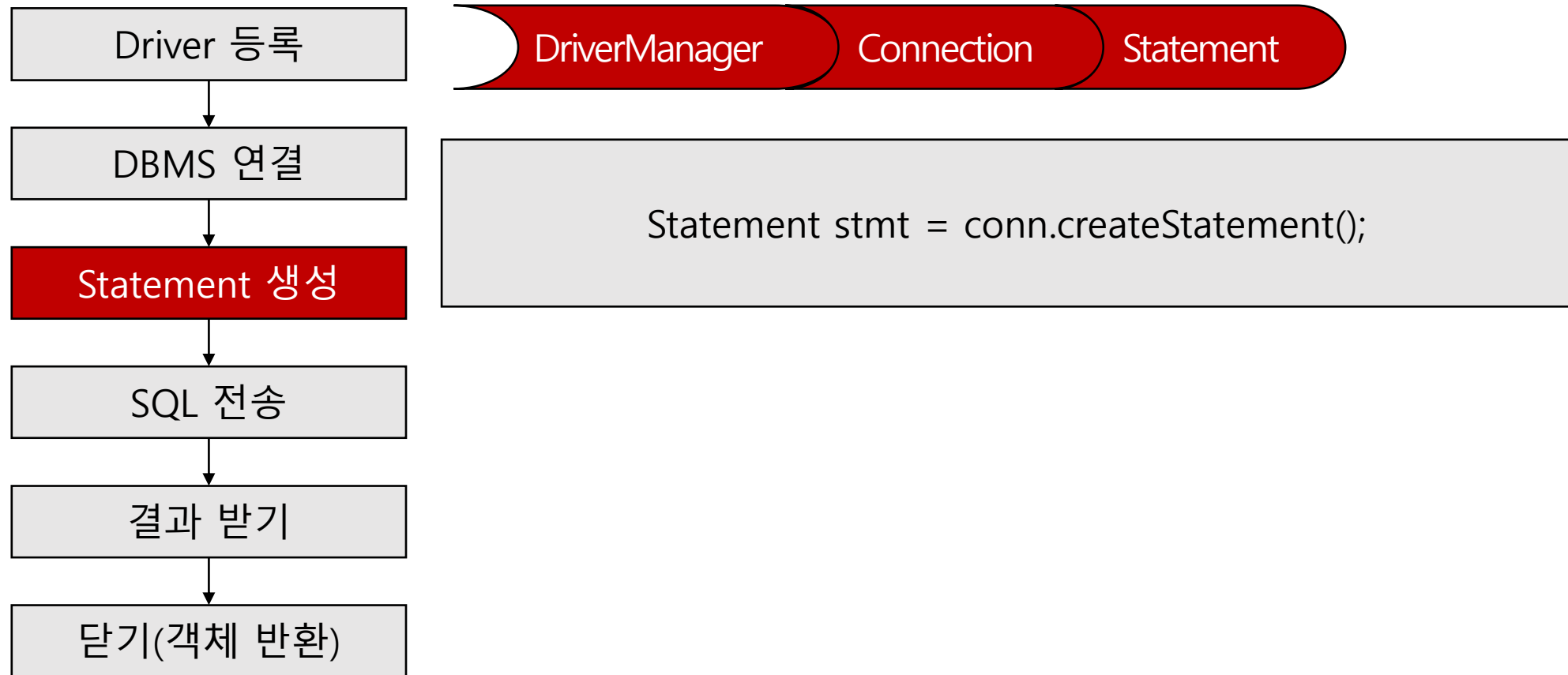
## ✓ 해당 Driver로부터 Connection instance 획득



\* 반드시 SQLException 처리를 해야 함

## ▶ JDBC 코딩 절차

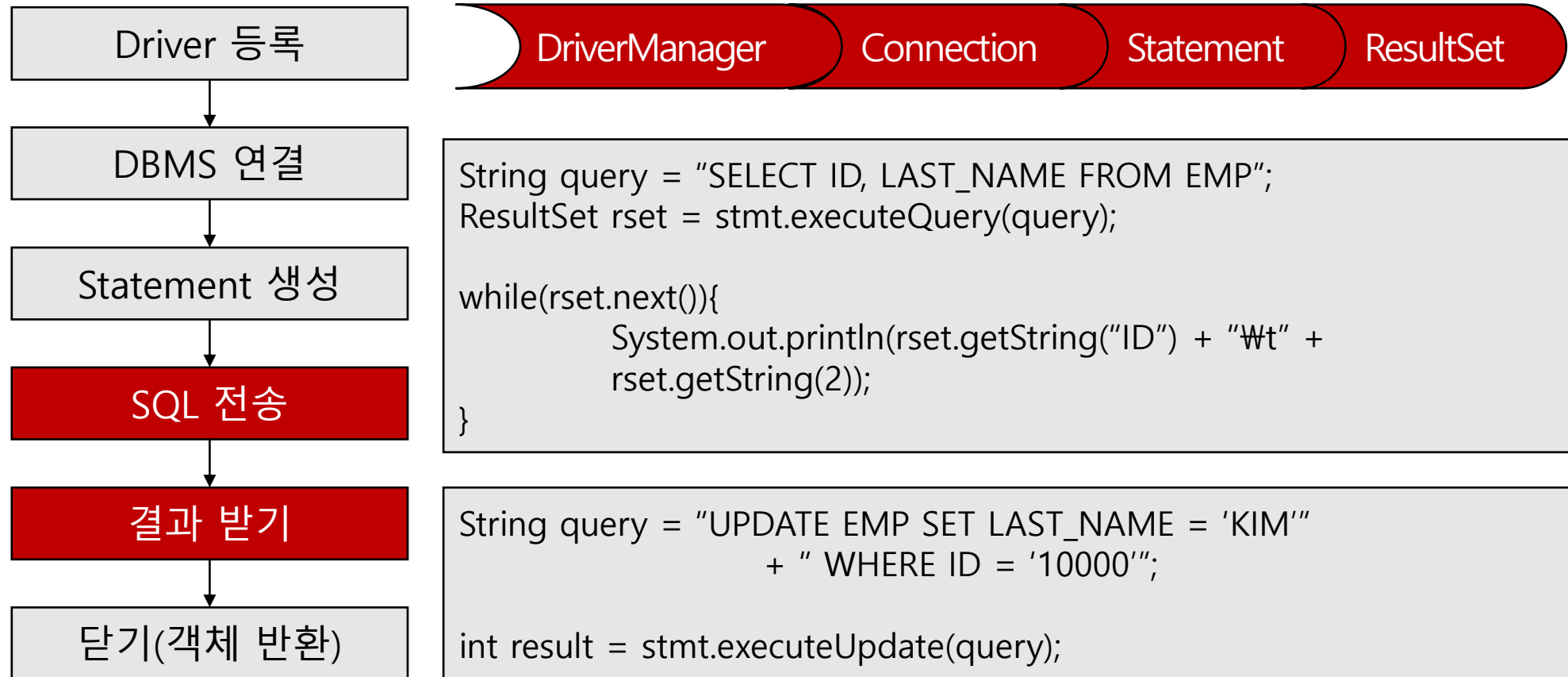
### ✓ Connection instance로부터 Statement instance 획득



\* 반드시 SQLException 처리를 해야 함

## ▶ JDBC 코딩 절차

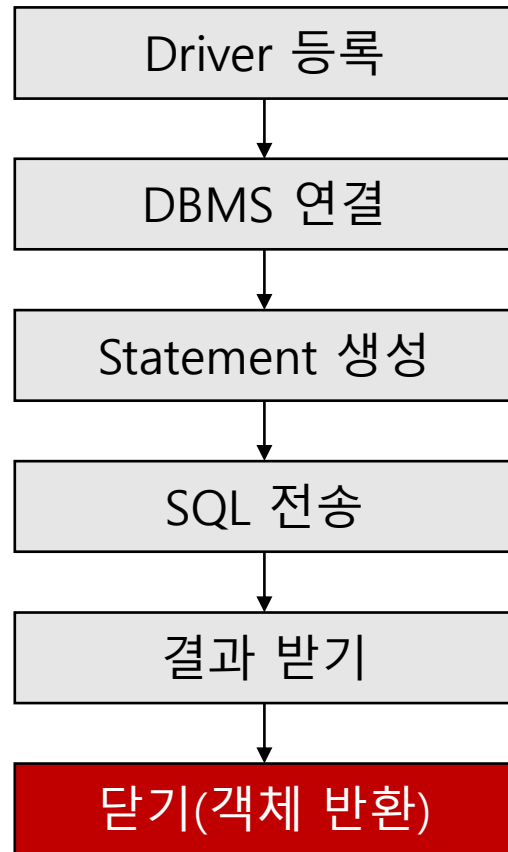
- ✓ Statement method를 이용하여 SQL문 실행
- ✓ 실행결과를 ResultSet(Select) 혹은 int형 변수(DML)로 받아서 처리



\* 반드시 SQLException 처리를 해야 함

## ▶ JDBC 코딩 절차

- ✓ DB로 부터 획득한 instance 들을 획득한 역순으로 반환



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

\* 반드시 SQLException 처리를 해야 함