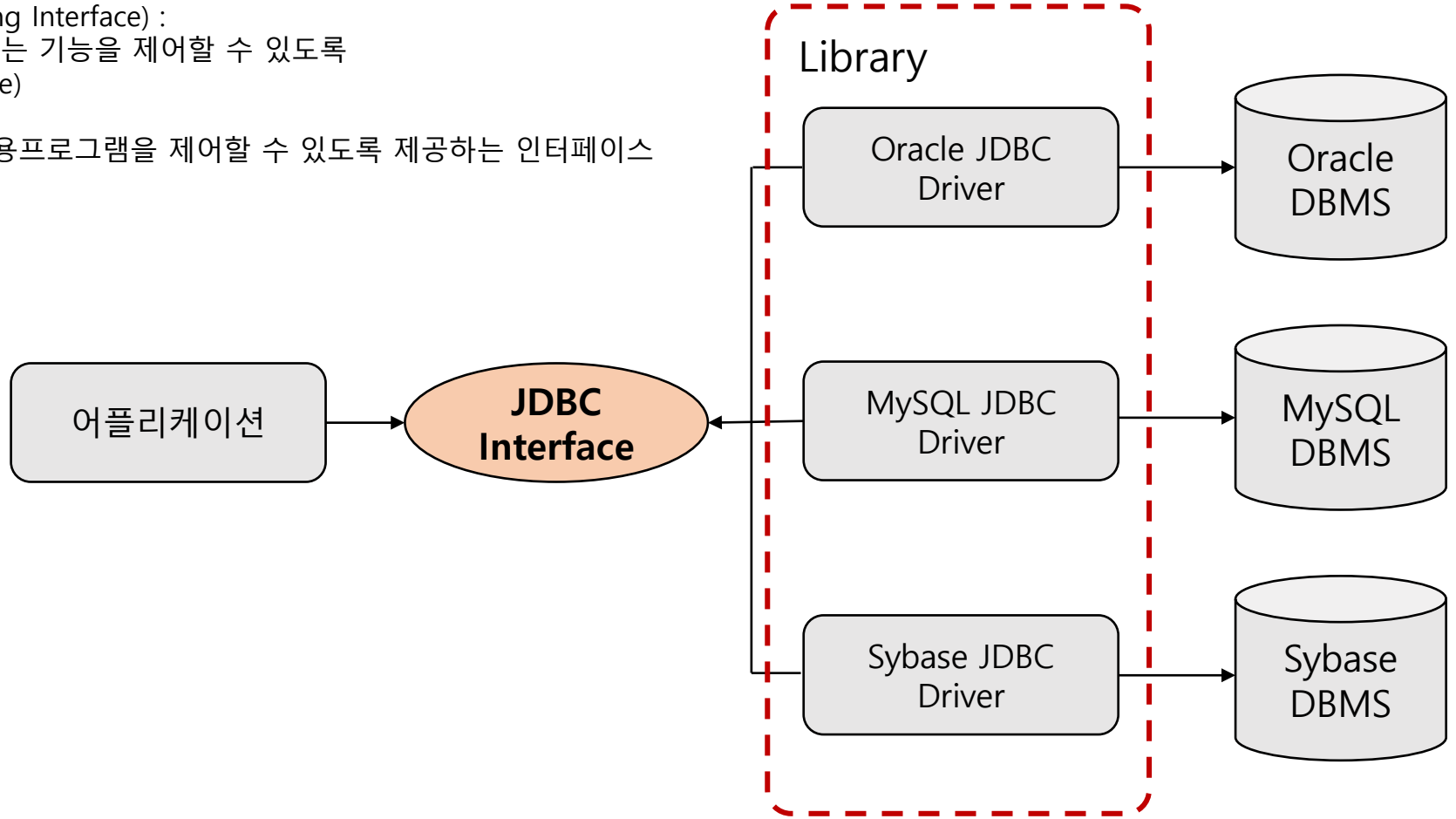
JDBC

JDBC(Java DataBase Connectivity)

Java에서 DB에 접근할 수 있게 해주는 **Java Programming API**

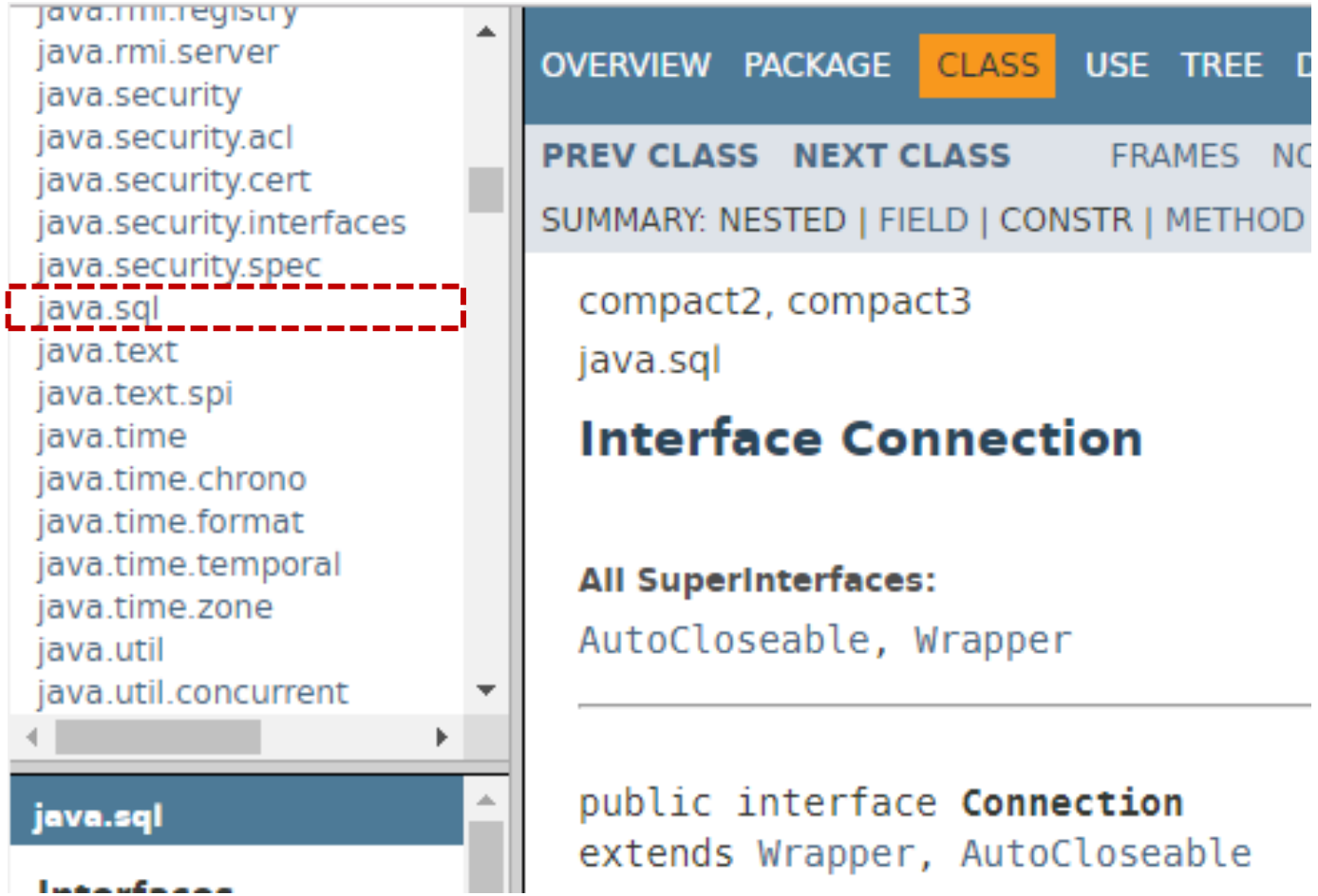
API(Application Programming Interface) :
다른 응용프로그램이 제공하는 기능을 제어할 수 있도록
제공하는 방법, 접점(Interface)

Java API : 자바에서 다른 응용프로그램을 제어할 수 있도록 제공하는 인터페이스



▶ JDBC(Java DataBase Connectivity)

✓ java.sql 패키지



▶ OJDBC

✓ OJDBC란?

오라클에서 제공하는 오라클 DB와 자바가 연결하기 위한 라이브러리

-> **Oracle JDBC Driver** 제공

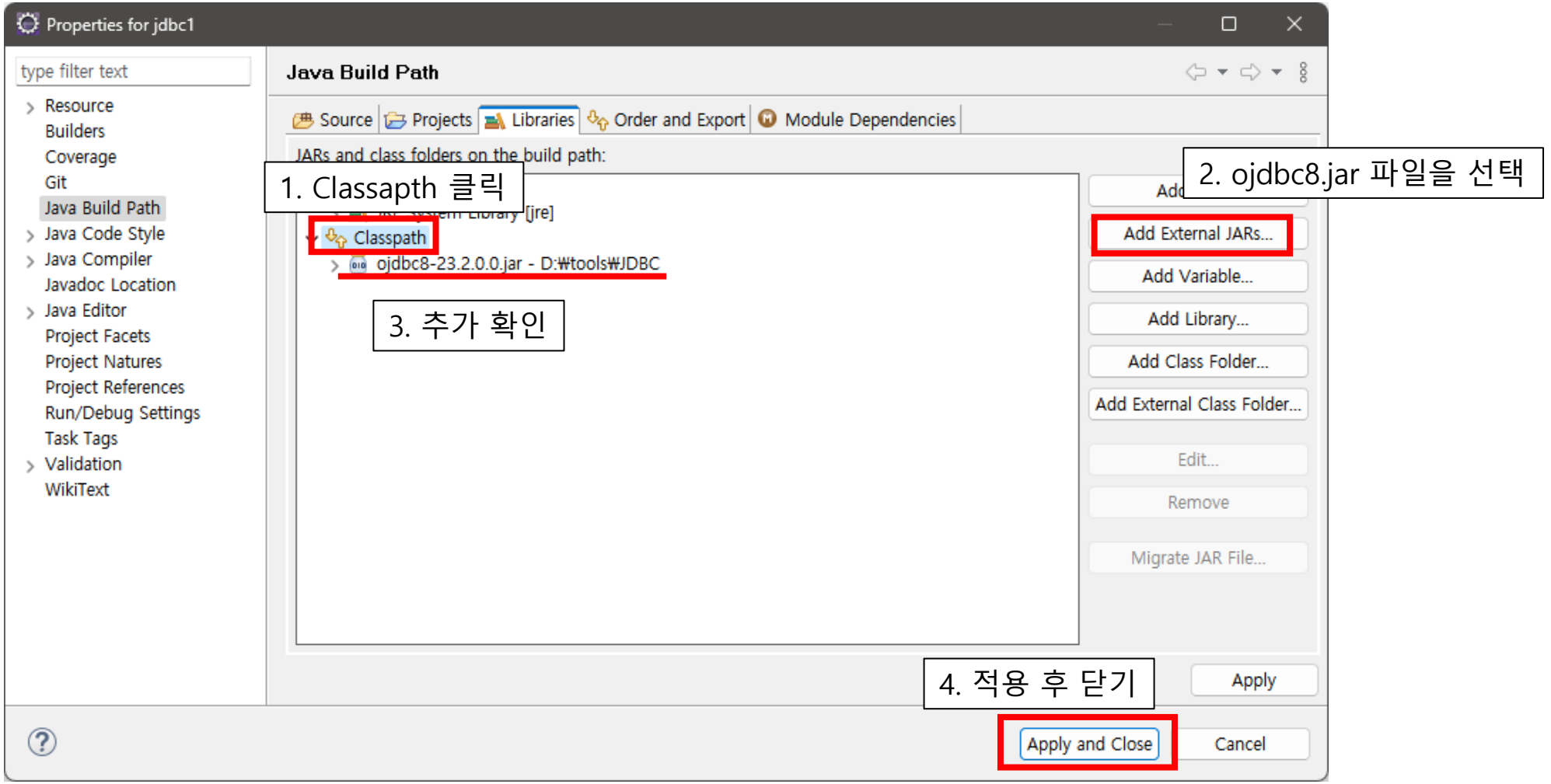
✓ OJDBC 다운로드

메이븐 레파지토리에서 ojdbc8.jar 다운로드

https://mvnrepository.com/artifact/com.oracle.database.jdbc/ojdbc8/23.2.0.0

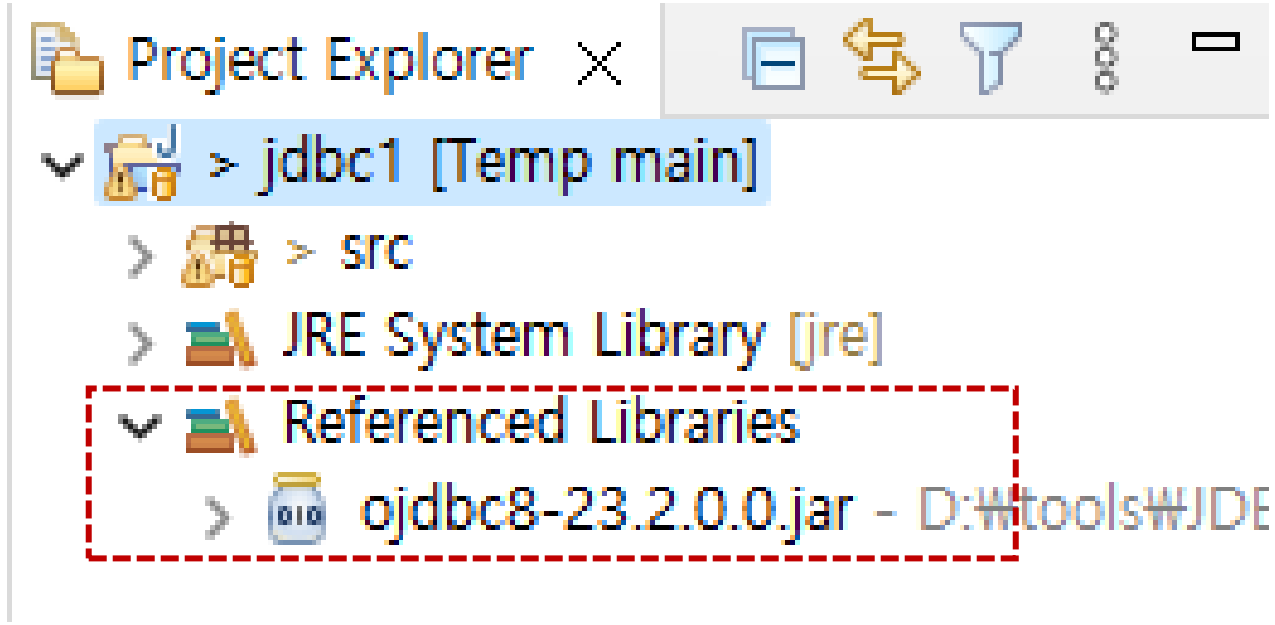
▶ Library 등록 방법

✓ Java Project 우클릭 -> Properties -> Java Build Path _> Libraries탭



▶ 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 데이터베이스와의 연결 정보를 저장한 객체.

Connection.createStatement() 메소드를 호출하여 Statement객체 생성

SQL문장을 실행시키기 전에 우선 Connection객체가 있어야 함

▶ JDBC 사용 객체

✓ Statement

Connection객체를 이용해 생성하는 객체로 Connection.createStatement() 메소드로 생성.

SQL을 String 형태로 전달하여 수행 결과를 반환 받음.

executeQuery() : SELECT문을 수행하고 결과 집합(ResultSet)을 반환

executeUpdate() : DML을 수행하고 결과 행의 수(int)를 반환

✓ 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

Connection객체를 이용해 생성하는 객체로 Connection. preparedStatement() 메소드로 생성.

SQL문장이 미리 컴파일 되고 실행 시간동안 인수 값을 위한 공간을 확보한다는 점에서 Statement와 다름

각각의 인수에 대해 placeholder(?)를 사용하여 SQL문장을 정의할 수 있게 함

✓ 예시

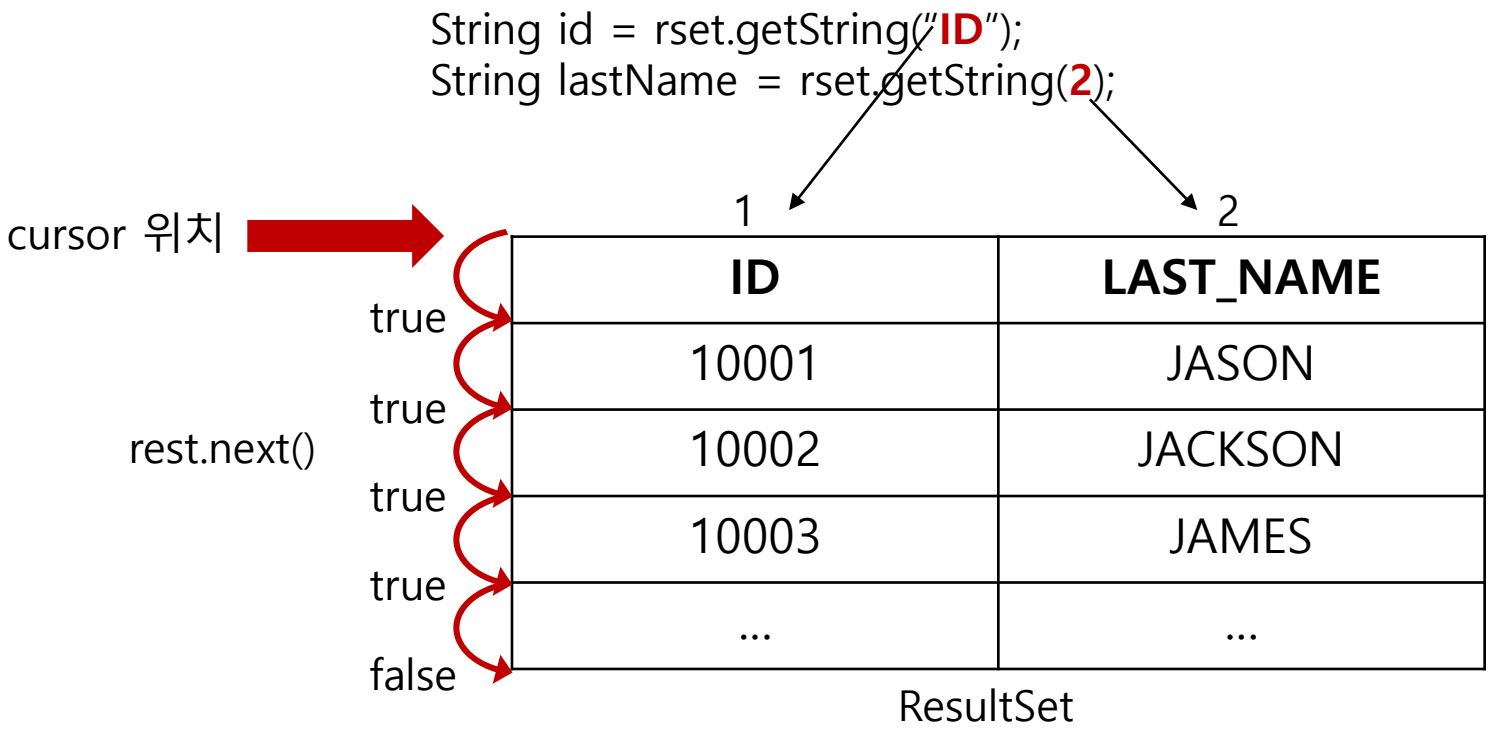
```
try{
        String query = "INSERT INTO MEMBER VALUES(?, ?)";
        pstmt = conn.preparedStatement(query);
        pstmt.setString(1, id);
        pstmt.setString(2, password);
} catch(SQLException e){
        e.printStackTrace();
}
```

▶ JDBC 사용 객체

✓ ResultSet

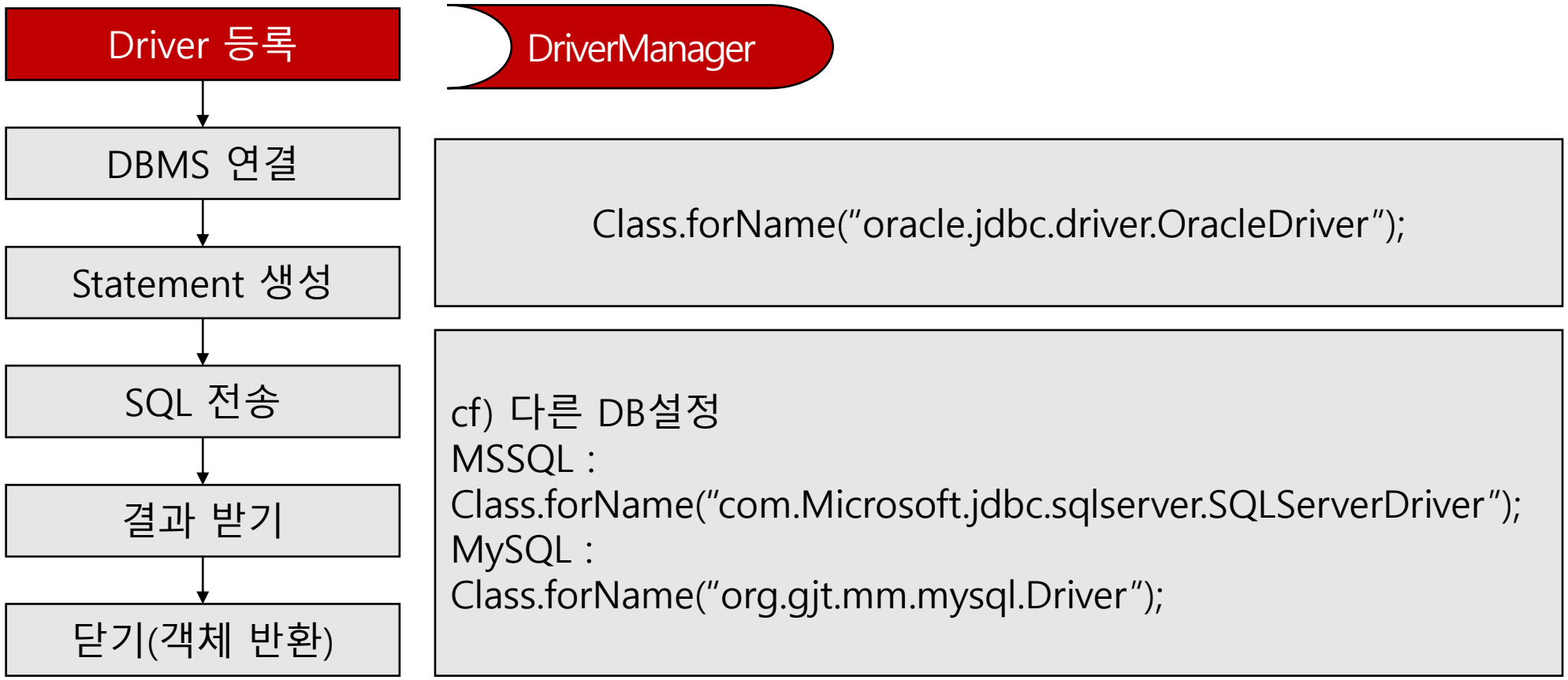
SELECT문을 사용한 질의 성공 시 Result Set 반환

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



▶ JDBC 코딩 절차

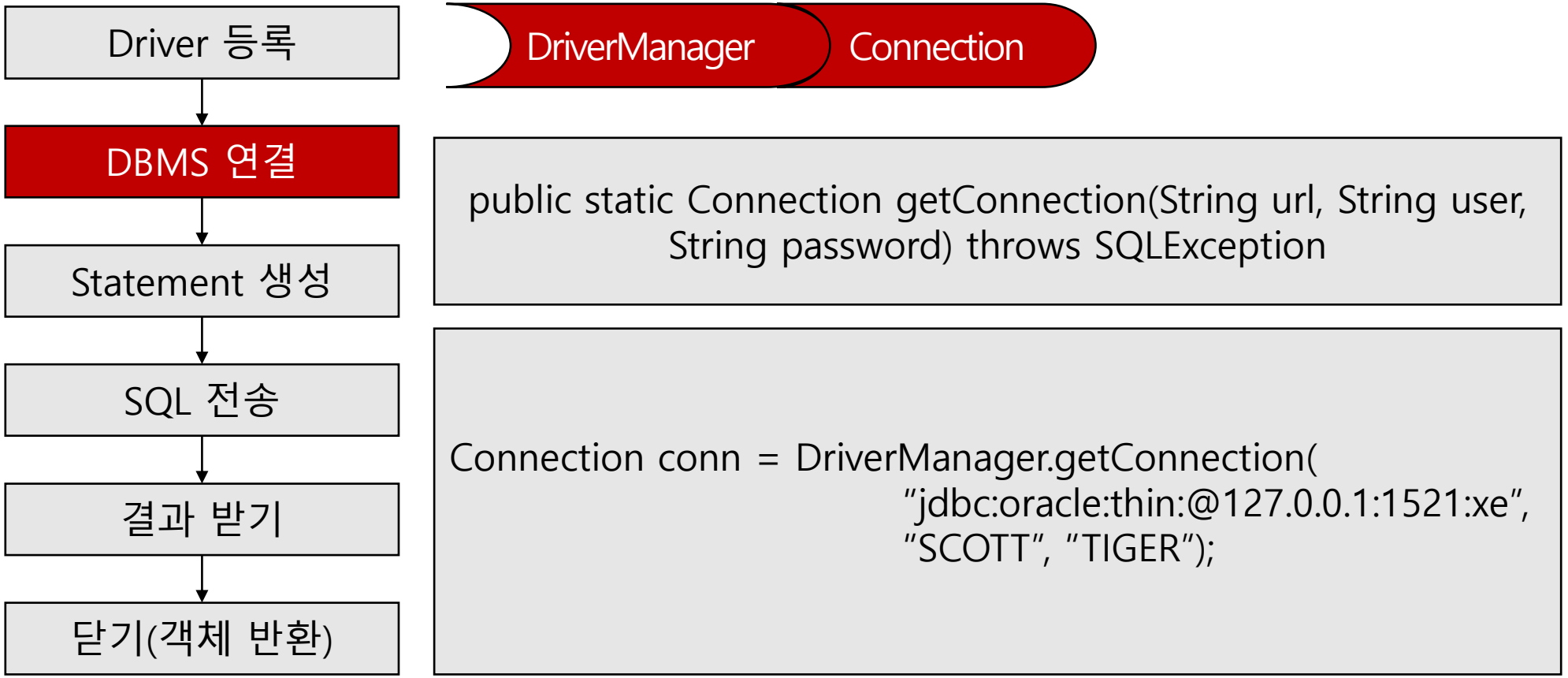
✓ DriverManager에 해당 DBMS Driver 등록



* 반드시 ClassNotFoundException 처리를 해야 함

▶ JDBC 코딩 절차

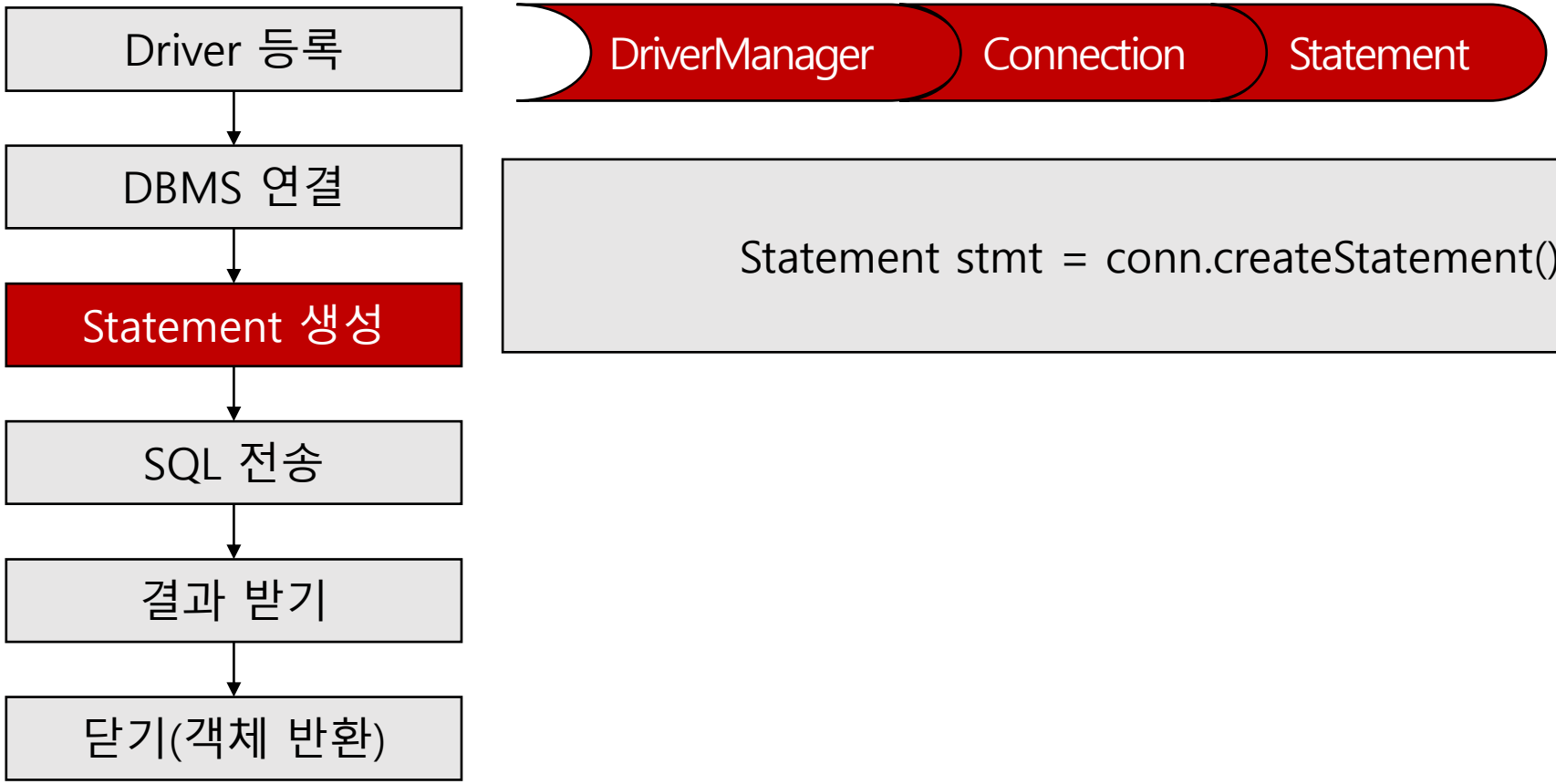
✓ 해당 Driver로부터 Connection instance 획득



* 반드시 SQLException 처리를 해야 함

▶ JDBC 코딩 절차

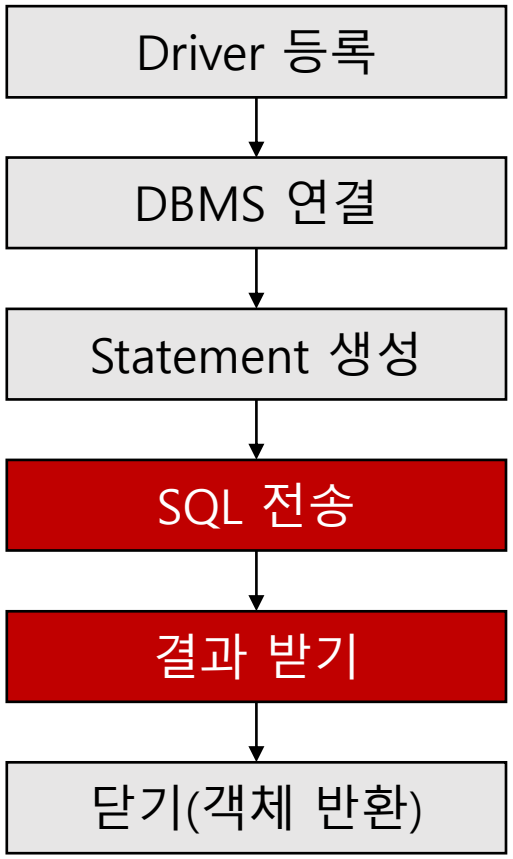
✓ Connection instance로부터 Statement instance 획득



* 반드시 SQLException 처리를 해야 함

▶ JDBC 코딩 절차

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



DriverManager → Connection → Statement → ResultSet

```
String query = "SELECT ID, LAST_NAME FROM EMP";
ResultSet rset = stmt.executeQuery(query);

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

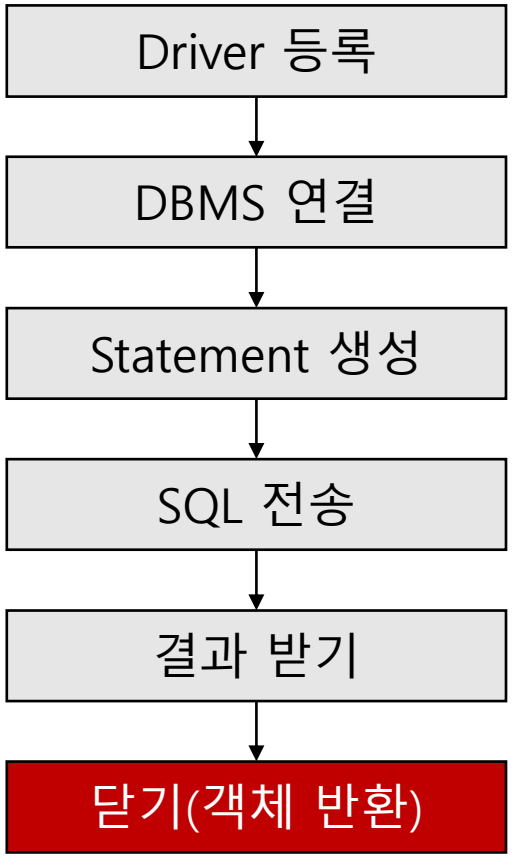
```
String query = "UPDATE EMP SET LAST_NAME = 'KIM'"
                    + " WHERE ID = '10000'";

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

* 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