이클립스 연동

SQL

환경설정

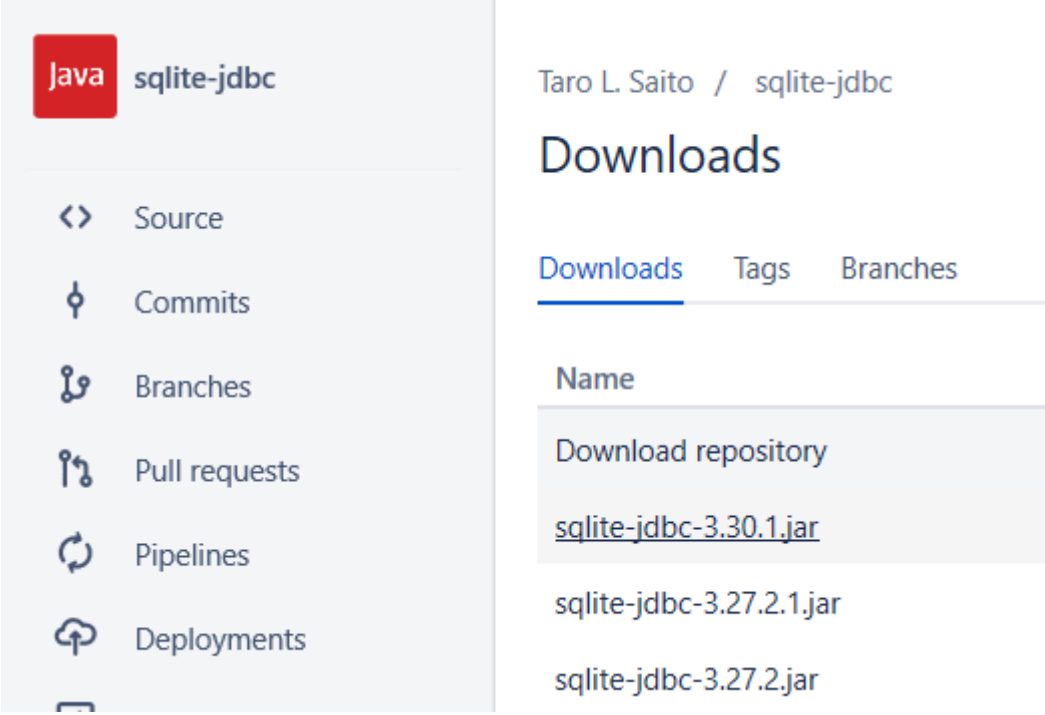
SQLite-jdbc

❖ 이해하기

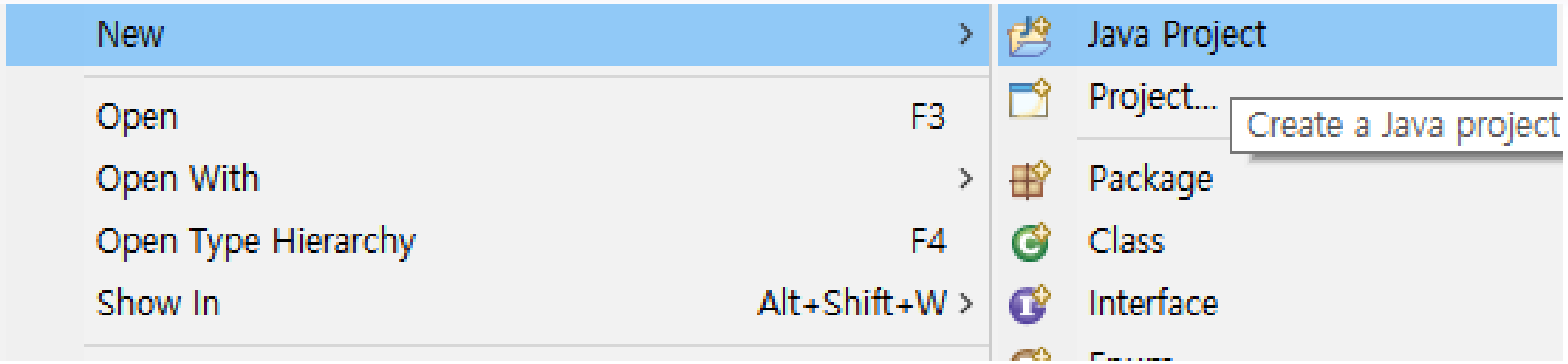
- DataBase를 JAVA와 연동하기 위해 JDBC 설치

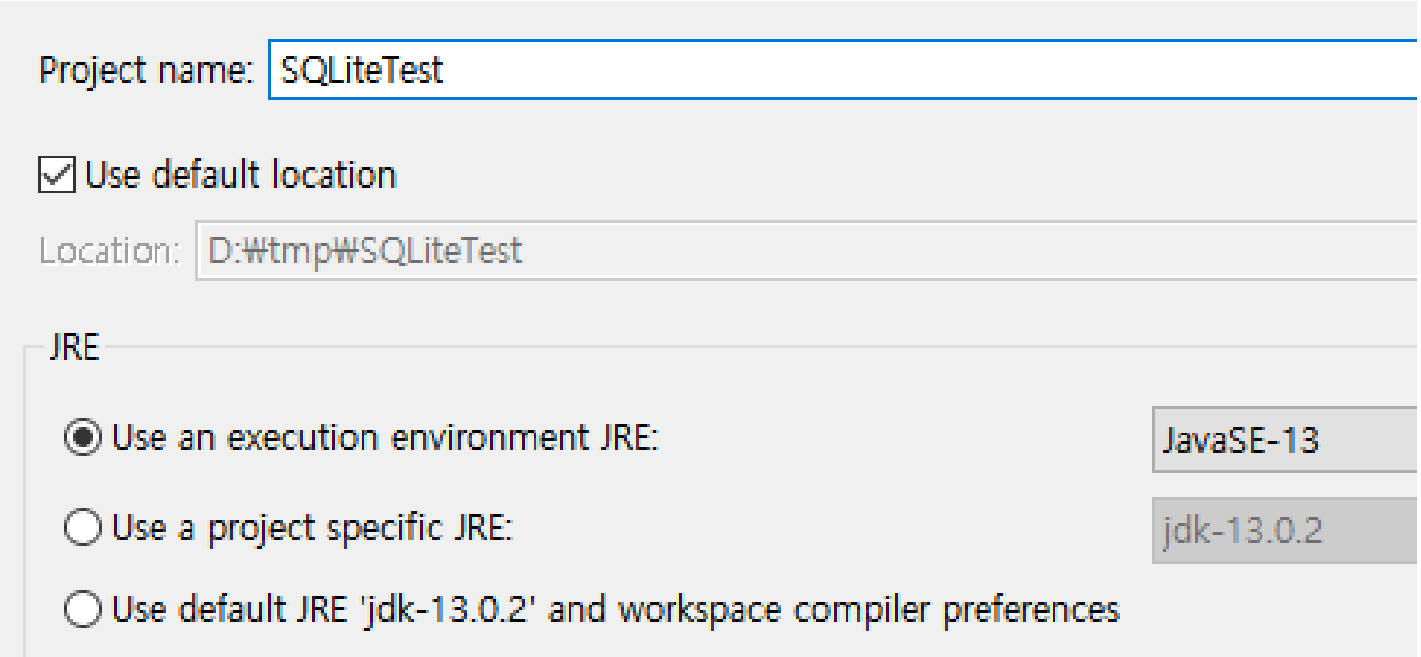
❖ 다운로드

- https://bitbucket.org/xerial/sqlite-jdbc/downloads/



프로젝트 생성


New
Open F3
Open With
Open Type Hierarchy F4
Show In Alt+Shift+W
Java Project
Project...
Create a Java project
Package
Class
Interface


Project name: SQLiteTest
Use default location
Location: D:₩tmp₩SQLiteTest
JRE
Use an execution environment JRE:
JavaSE-13
Use a project specific JRE:
jdk-13.0.2
Use default JRE 'jdk-13.0.2' and workspace compiler preferences

클래스 생성

Name: SQLiteTest

Modifiers: (●) public ( ) package ( ) private ( ) protected

[ ] abstract [ ] final [ ] static

Superclass: java.lang.Object [Browse...]

Interfaces: [Add...] [Remove]

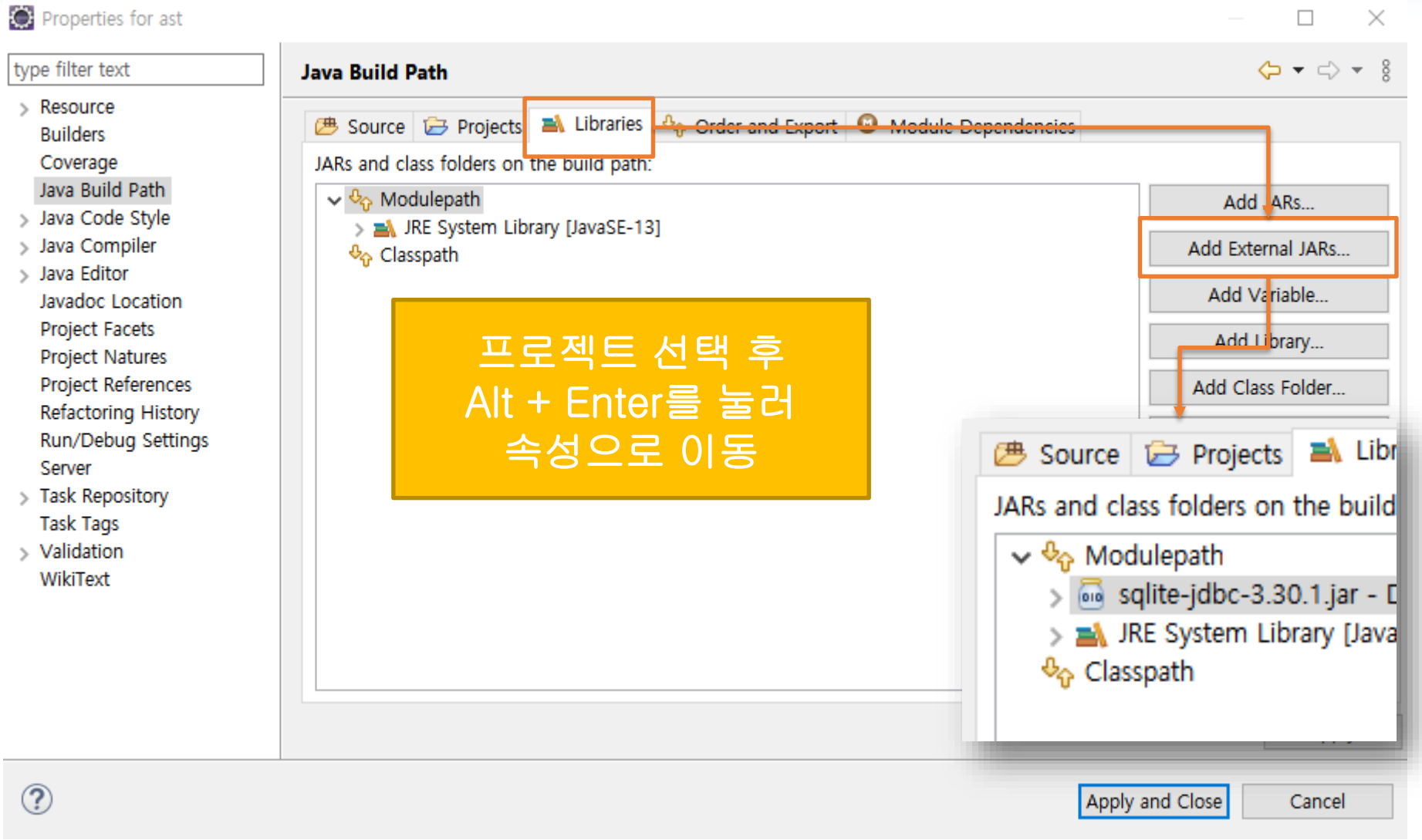
Which method stubs would you like to create?

[x] public static void main(String[] args)

[ ] Constructors from superclass

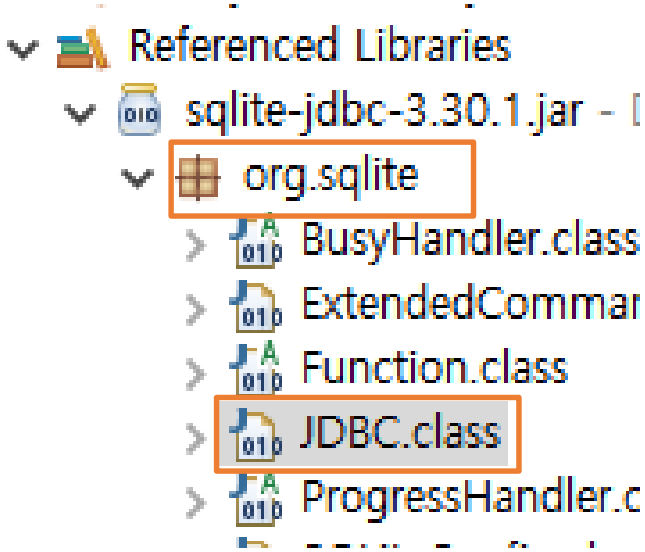
[x] Inherited abstract methods

라이브러리 추가


Properties for ast
type filter text
Resource
Builders
Coverage
Java Build Path
Java Code Style
Java Compiler
Java Editor
Javadoc Location
Project Facets
Project Natures
Project References
Refactoring History
Run/Debug Settings
Server
Task Repository
Task Tags
Validation
WikiText
Java Build Path
Source
Projects
Libraries
Order and Export
Module Dependencies
JARs and class folders on the build path:
Modulepath
JRE System Library [JavaSE-13]
Classpath
Add JARs...
Add External JARs...
Add Variable...
Add Library...
Add Class Folder...
프로젝트 선택 후
Alt + Enter를 눌러
속성으로 이동
Source
Projects
JARs and class folders on the build
Modulepath
sqlite-jdbc-3.30.1.jar
JRE System Library [Java
Classpath
Apply and Close
Cancel

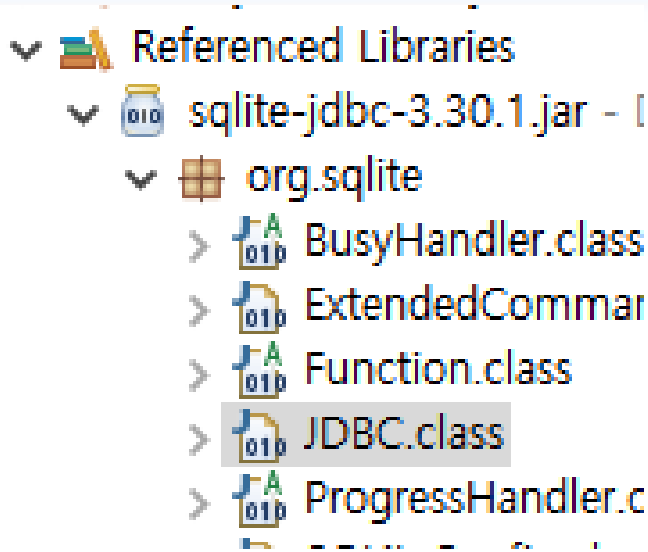
라이브러리 확인

```
public class SQLiteTest {
   public static void main(String[] args) {
      try {
         Class.forName("org.sqlite.JDBC");
      } catch (ClassNotFoundException e) {
         e.printStackTrace();
      }
   }
}
```

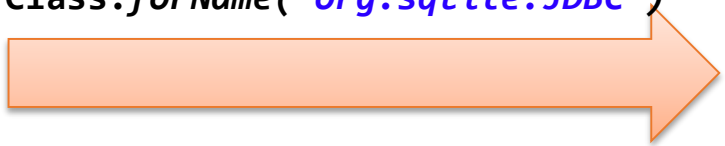


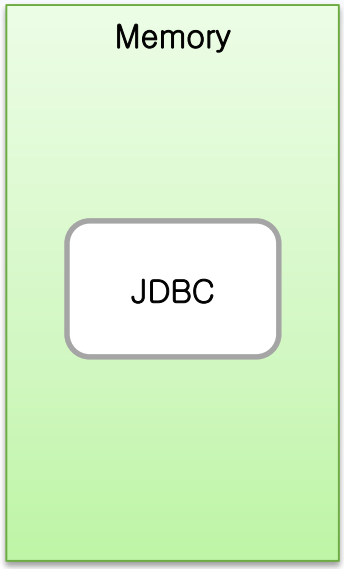
Statement 이해하기

메모리 등록




```
Class.forName("org.sqlite.JDBC")
```

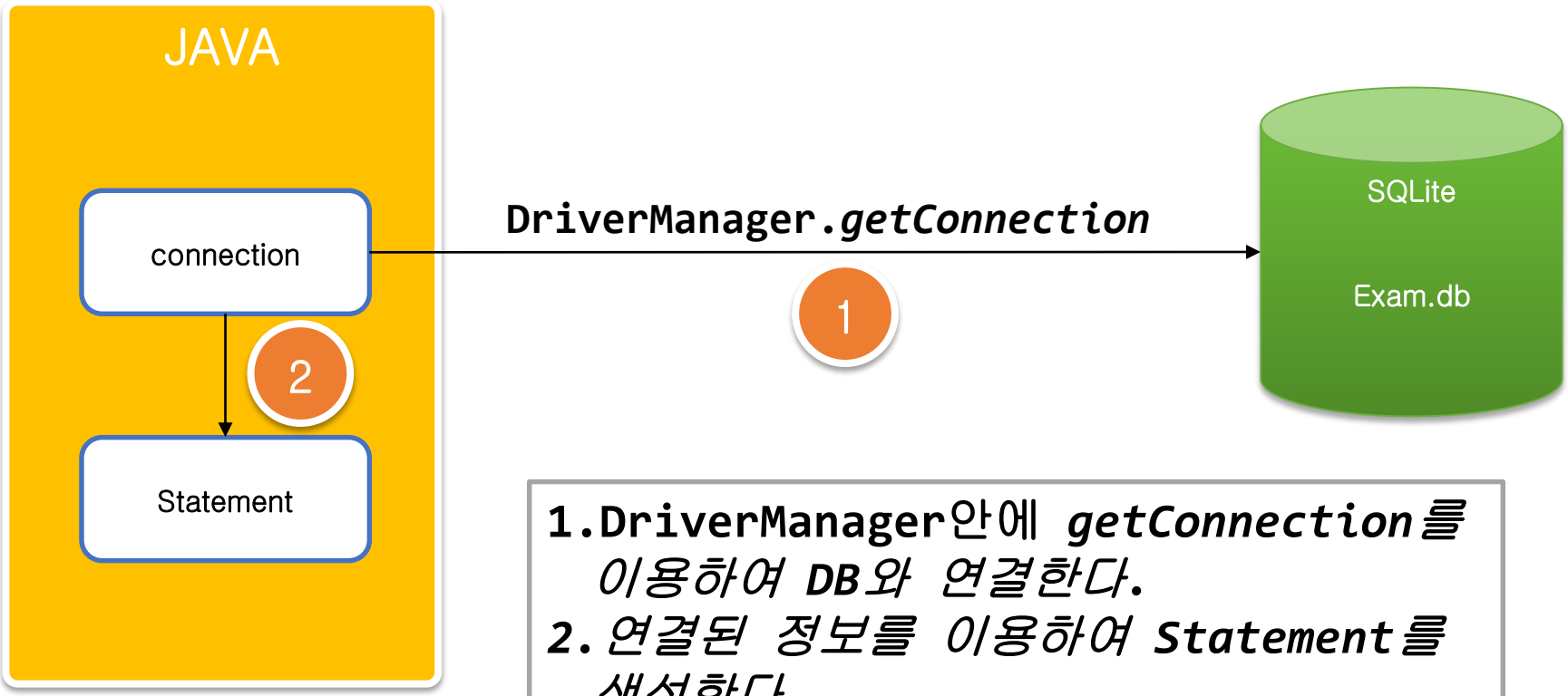




JDBC 클래스를
이용하기 위해
메모리에 업로드 시킴

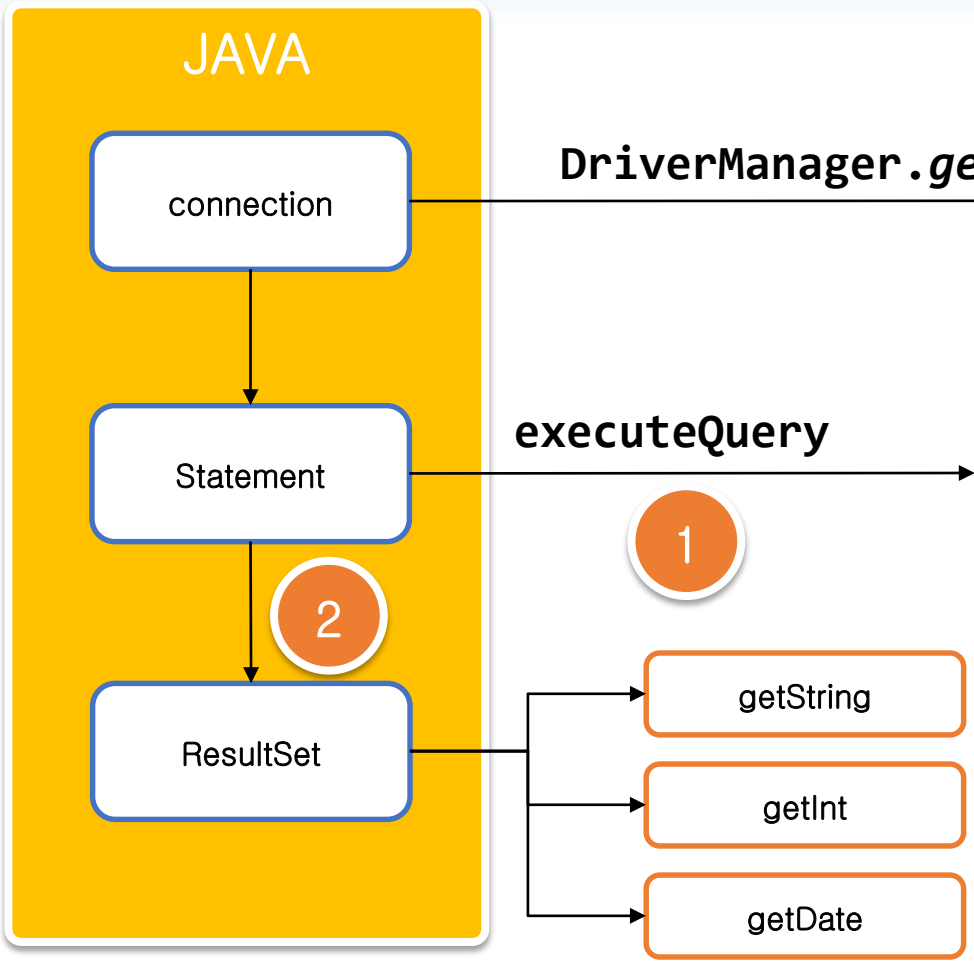
statement

```
DriverManager.getConnection("jdbc:sqlite:[드라이버명][경로][db명]");
```

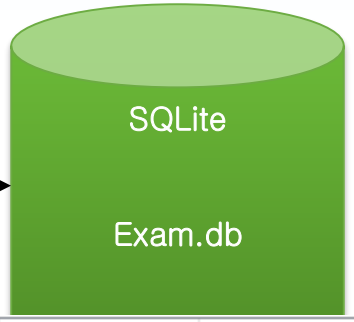


1. DriverManager안에 getConnection를 이용하여 DB와 연결한다.
2. 연결된 정보를 이용하여 Statement를 생성한다.

SELECT



DriverManager.*getConnection*



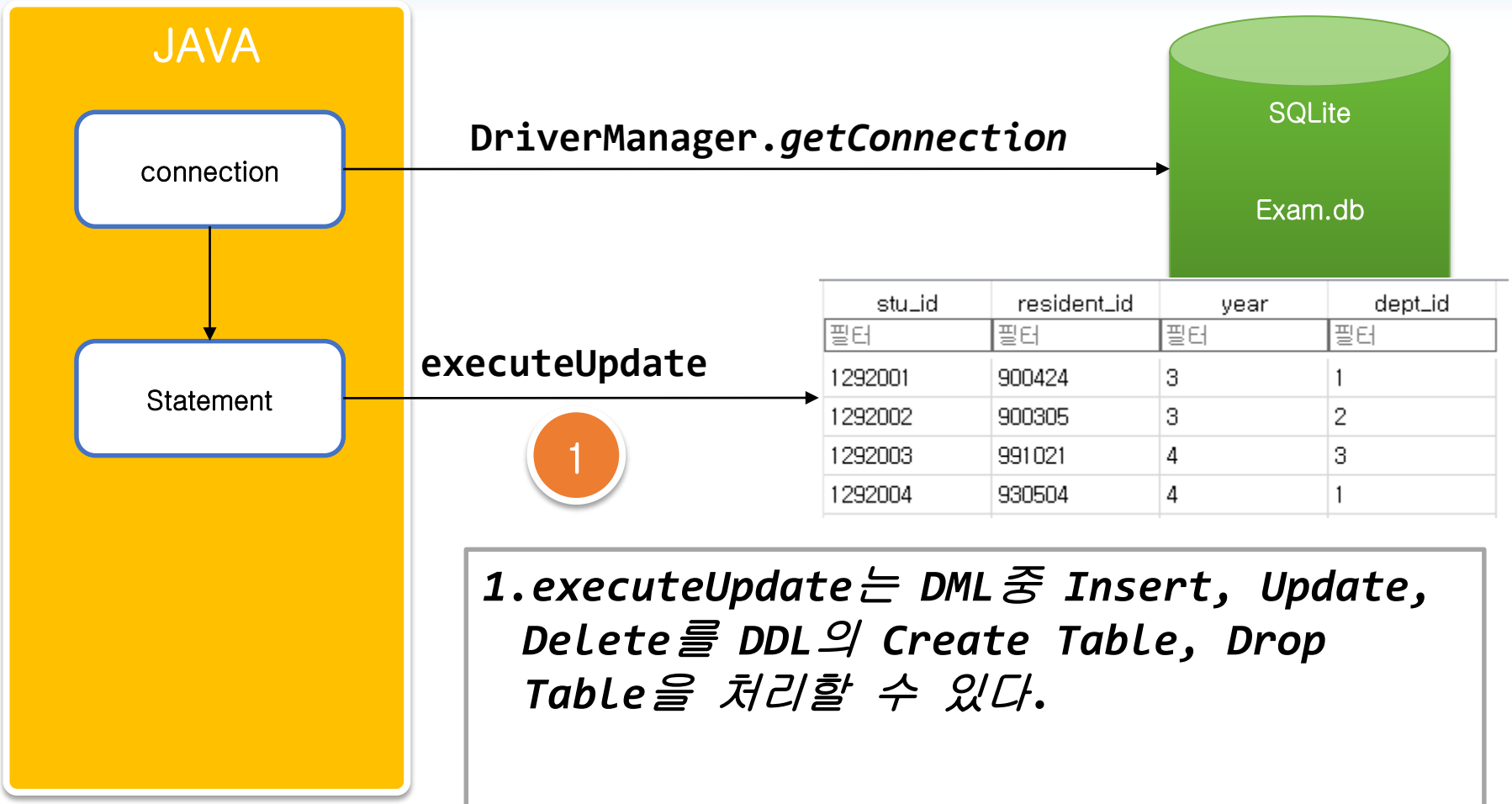
executeQuery

1

stu_id	resident_id	year	dept_id
필터	필터	필터	필터
1292001	900424	3	1
1292002	900305	3	2
1292003	991021	4	3
1292004	930504	4	1

1. executeQuery를 이용하여 query문 전송
2. 결과를 ResultSet으로 전송하며 getXXX를 이용하여 자료형에 맞는 데이터 처리

Update



stu_id	resident_id	year	dept_id
필터	필터	필터	필터
1292001	900424	3	1
1292002	900305	3	2
1292003	991021	4	3
1292004	930504	4	1

1.executeUpdate는 DML중 Insert, Update, Delete를 DDL의 Create Table, Drop Table을 처리할 수 있다.

실습

Java.JDBC.Test

- JDBC01_classforname
- JDBC02_connection
- JDBC03_select
- JDBC04_CreateTable

Quiz

JAVA.JDBC.Quiz

❖ 다음과 같이 두 명의 정보를 입력하시오

id	pw
필터	필터
jin	jin1234
din	jin1234

❖ Din의 패스워드를 din1234로 변경하시오

JAVA.JDBC.Quiz

❖ login table에서 jin 계정을 삭제하시오

id	pw
필터	필터
din	din1234

❖ Login table을 제거 하시오

PreparedStatement 이해하기

PreparedStatement

- 이해하기
  - 가변적인 데이터 처리
- 사용 예

```
PreparedStatement pstmt = conn.prepareStatement(
        "SELECT        count(*) "+
        "FROM          student " +
        "WHERE         year = ?"
        );

pstmt.setInt(1, 2);
```

실습

Select

```
final static String SQL =
    "SELECT   count(*) "+
    "FROM     student " +
    "WHERE    year=?";
public static void main(String[] args) {
    try {
        System.out.println("학년을 입력하세요?");
        int year= System.in.read()-48;

        Class.forName(DRIVER);
        Connection conn = DriverManager.getConnection(DB);
        PreparedStatement pstmt = conn.prepareStatement(SQL);

        pstmt.setInt(1, year);
        ResultSet rs = pstmt.executeQuery();
        while(rs.next()) {
            System.out.println(rs.getInt("count(*)"));
```

Insert

```
final static String SQL =
"INSERT INTO student " +
"VALUES(?, ?, ?, ?);";
public static void main(String[] args) {
   Scanner sc = new Scanner(System.in);
   System.out.println("학번을 입력하세요?");
   String stu_id = sc.next();
   System.out.println("생년월일을 입력하세요?");
   String resident_id = sc.next();
   System.out.println("학년을 입력하세요?");
   int year = sc.nextInt();
   System.out.println("학과를 입력하세요?");
   String dept_id = sc.next();
```

Insert(con' t)

```
try {
    Class.forName(DRIVER);
    Connection conn =
    DriverManager.getConnection(DB);
    PreparedStatement pstmt =
    conn.prepareStatement(SQL);

    pstmt.setString(1, stu_id);
    pstmt.setString(2, resident_id);
    pstmt.setInt(3, year);
    pstmt.setString(4, dept_id);

    pstmt.executeUpdate();
```