자바2

(DB 프로그래밍)

Chapter 12
DB 프로그래밍

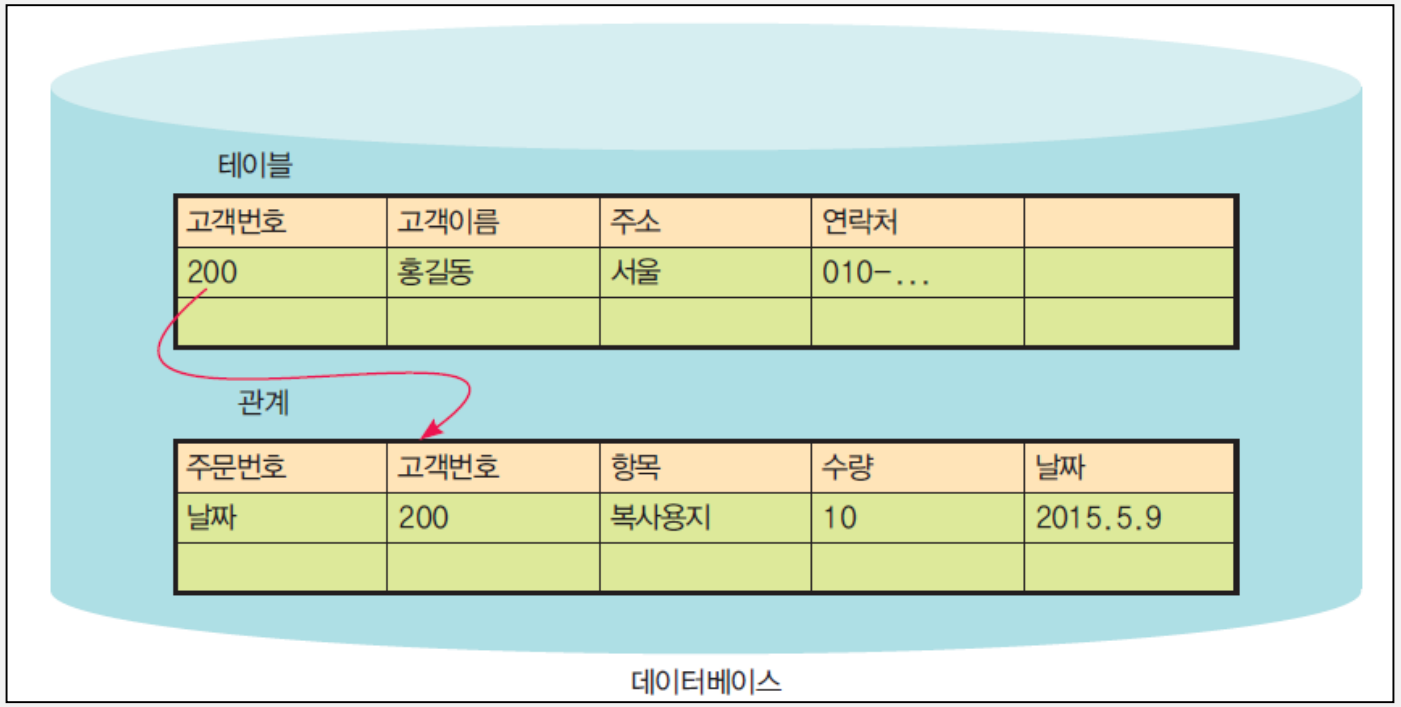
Chapter12의 학습목표

- 자바에서 DB를 연결하는 방법에 대해 알아본다
- 자바를 통해 DB에 데이터를 추가, 수정, 삭제하는 법을 학습한다

DB

데이터베이스 (DataBase)

- 데이터가 빠르게 추출될 수 있도록 데이터를 조직화하여 저장하는 방법
- 열과 행으로 이루어질 수 있음
- DBMS (데이터베이스 관리 시스템) : 다수의 사용자를 위하여 데이터가 저장, 접근, 변경되는 기능을 정의하고 있는 서버
- 관계형 데이터베이스 (relational data base) : 여러 개의 테이블이 존재하고 테이블과 테이블 간에 공통적인 데이터로 어떤 관계를 성립할 수 있는 데이터베이스 시스템



DB

데이터베이스 (DataBase) 프로그램

- 오라클, SQL Server, MySQL, mongoDB, Firebase, MariaDB

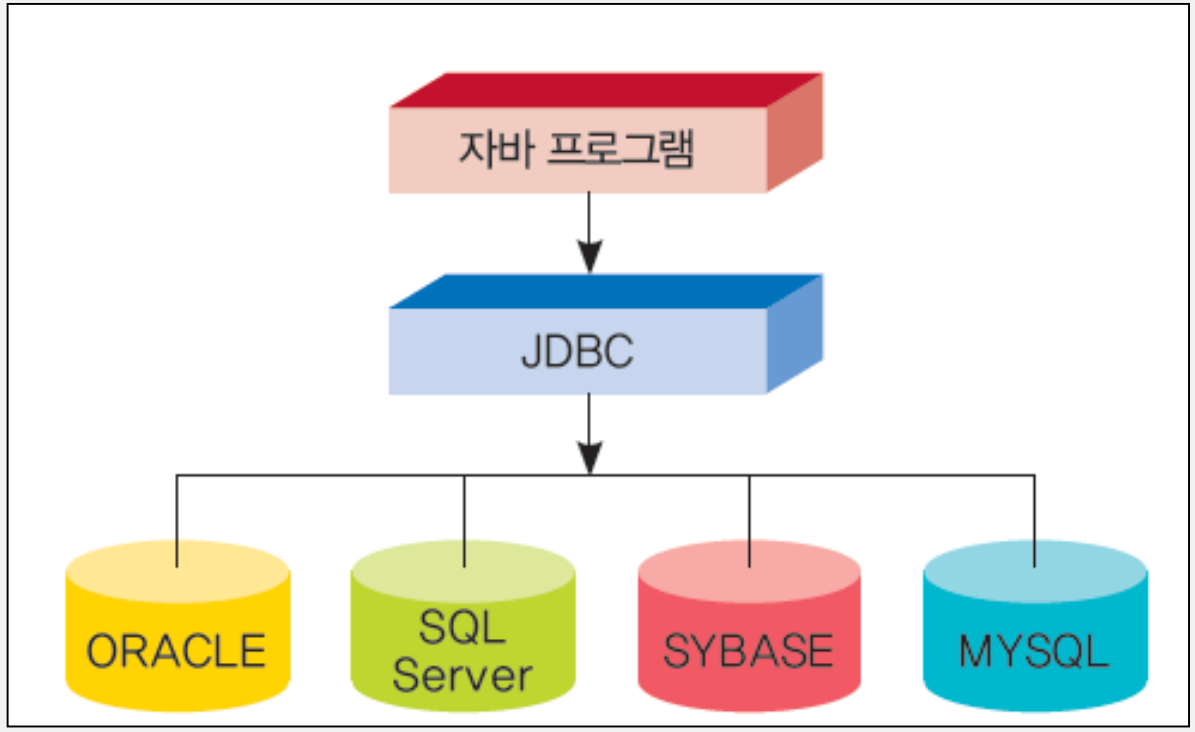
DB 응용 프로그램 개발 절차

1. DBMS 설치
2. DBMS에 필요한 JDBC 드라이버 설치 (“Connector/J”)
3. DB 응용 프로그램을 개발

DB

자바와 데이터베이스

- JDBC (Java Database Connectivity) : 데이터베이스에 연결하여서 데이터베이스 안의 데이터에 대하여 검색하고 변경할 수 있게 하는 자바의 API
- 데이터베이스는 네트워크로 연결된 컴퓨터에 데이터를 제공

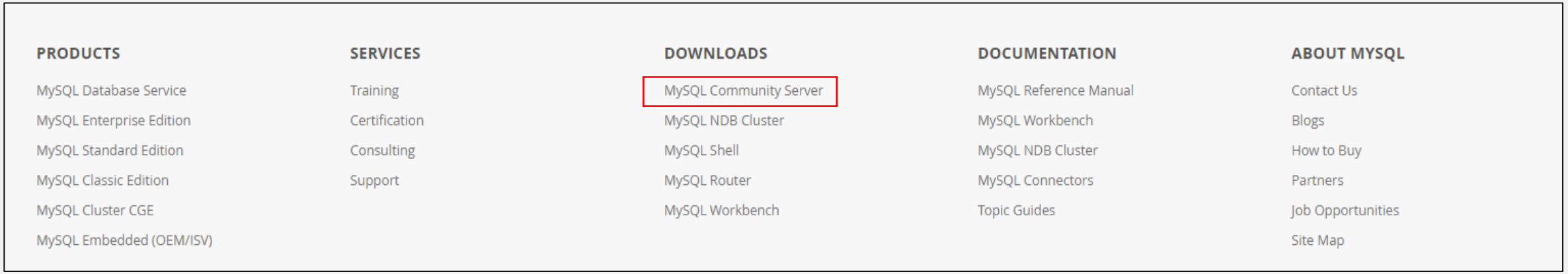


DB

MySQL 설치

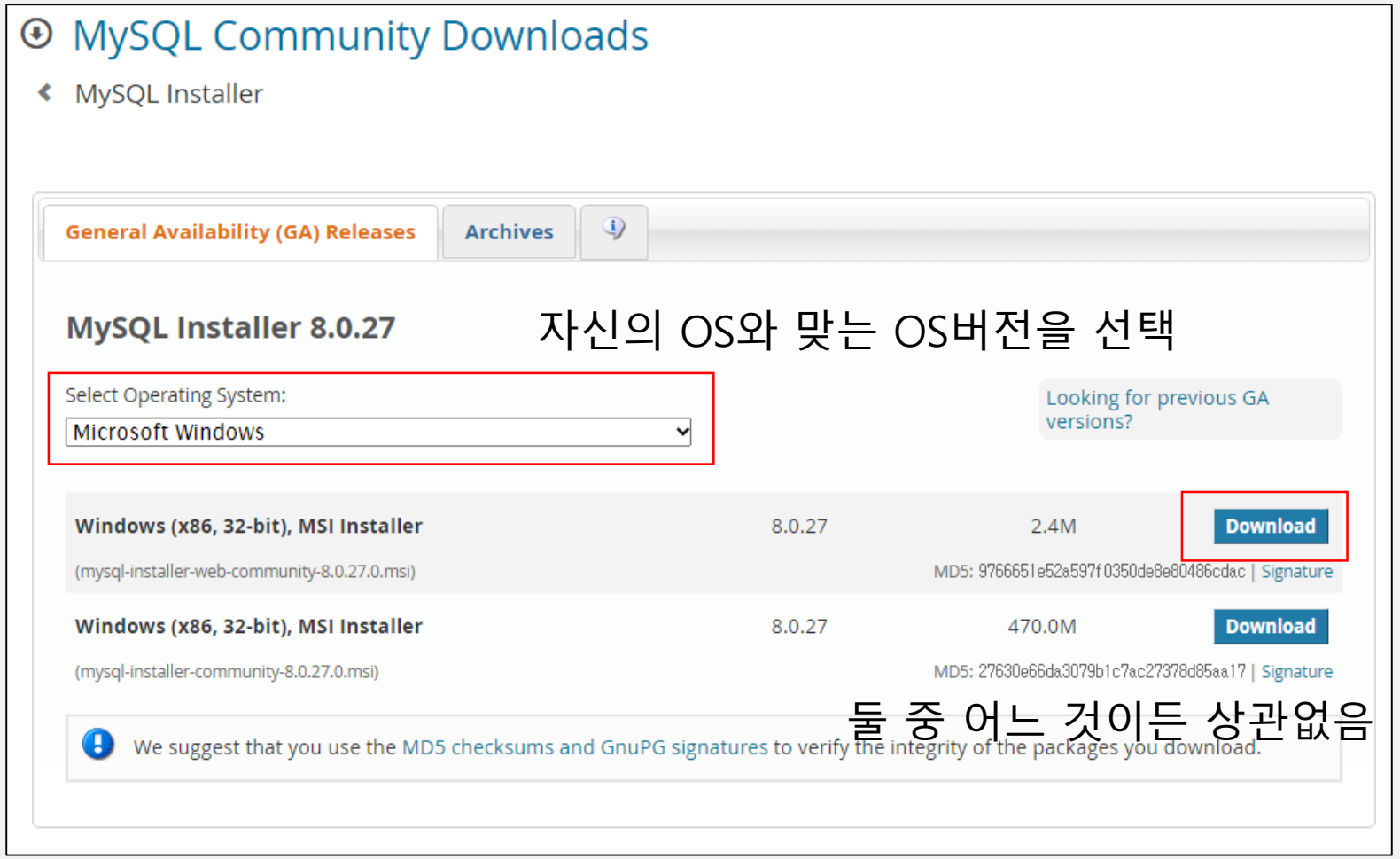
- https://www.mysql.com/

가장 하단으로 이동



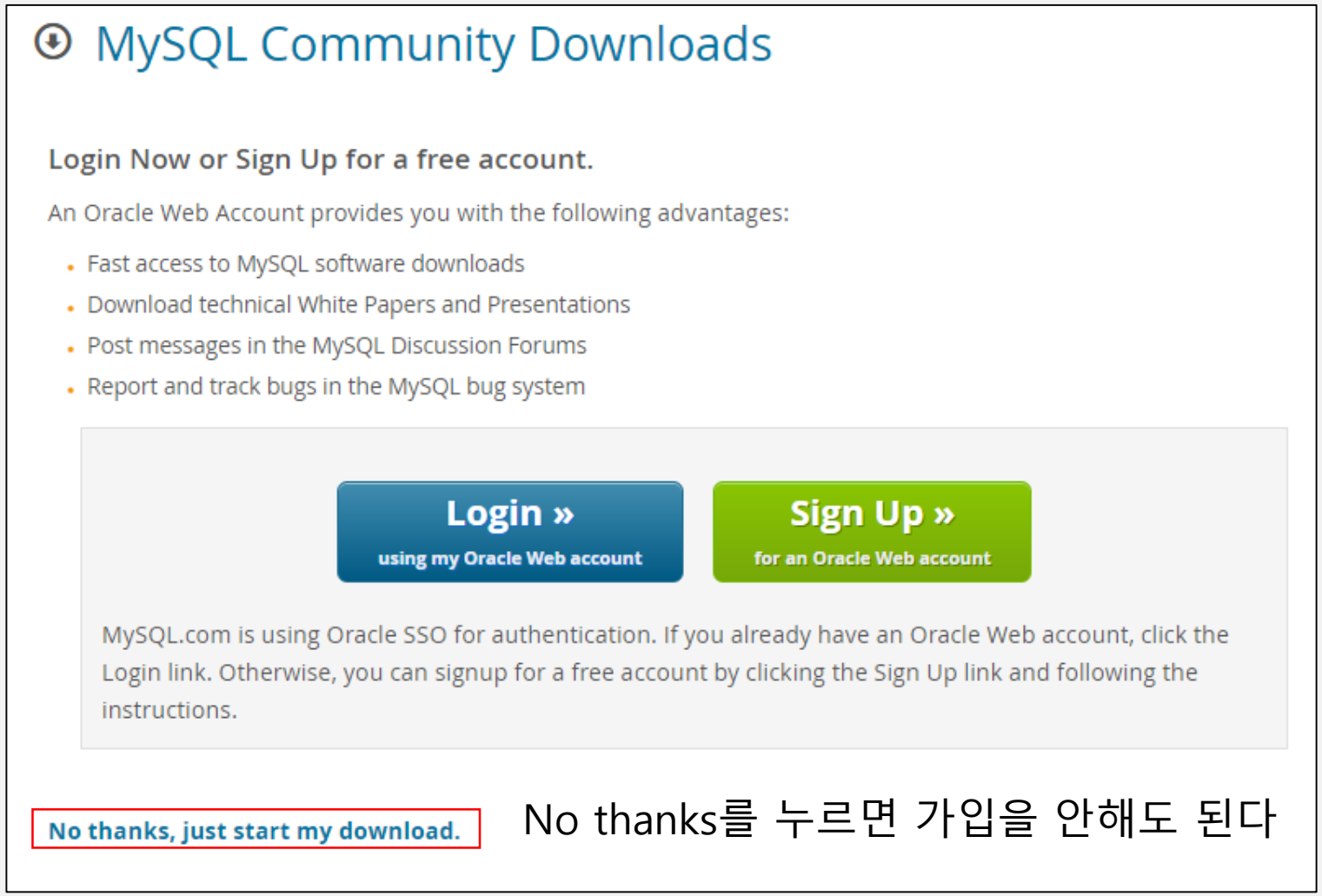
DB

MySQL 설치



DB

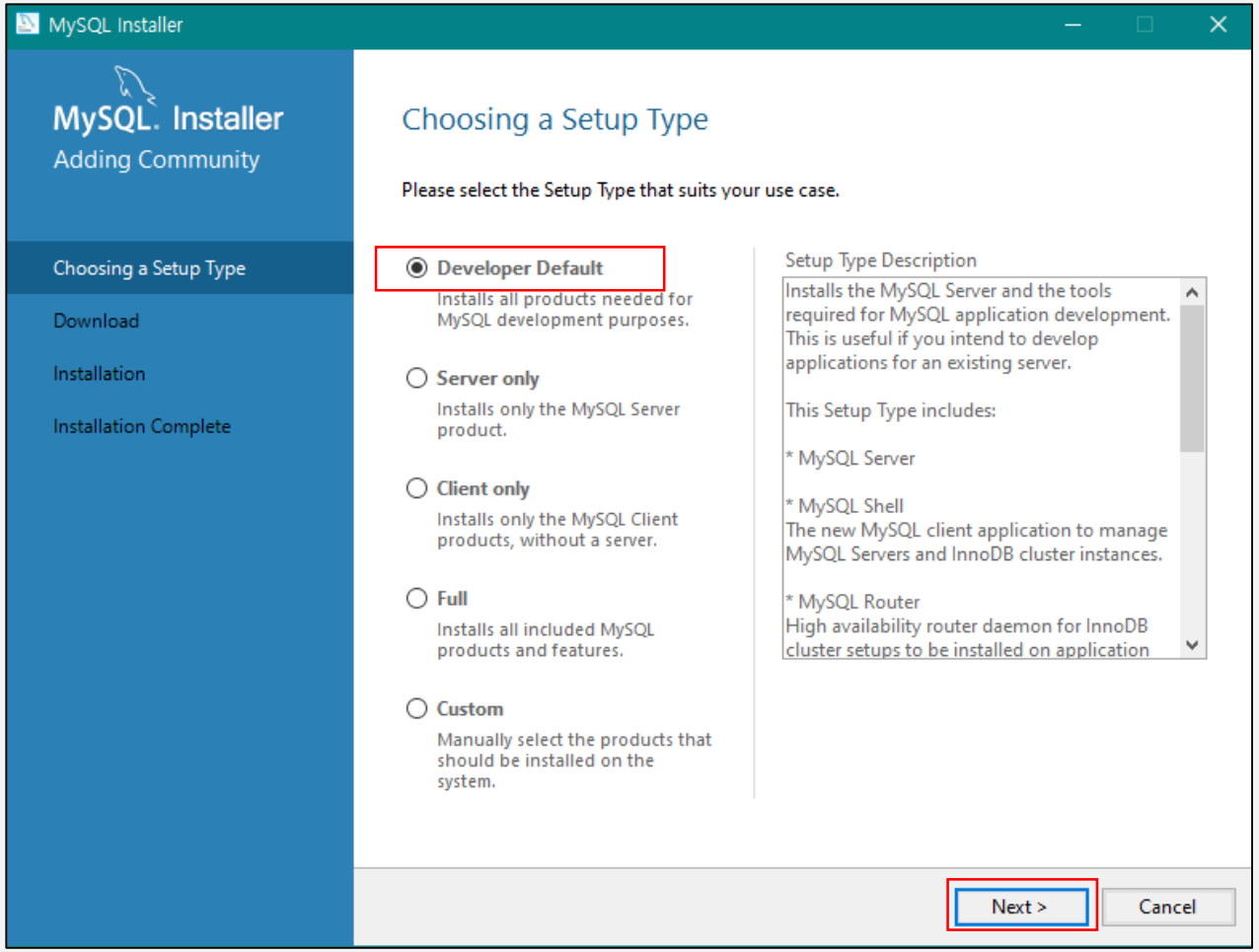
MySQL 설치

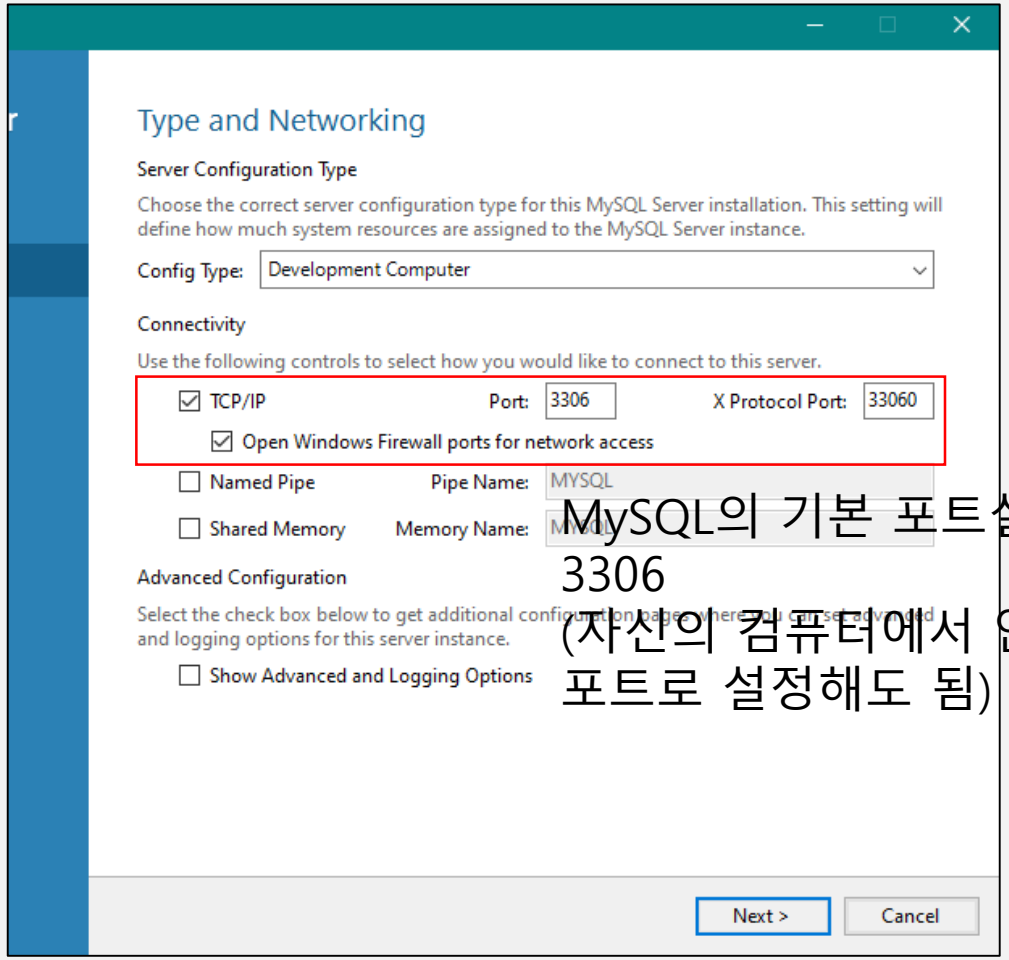


No thanks를 누르면 가입을 안해도 된다

DB

MySQL 설치

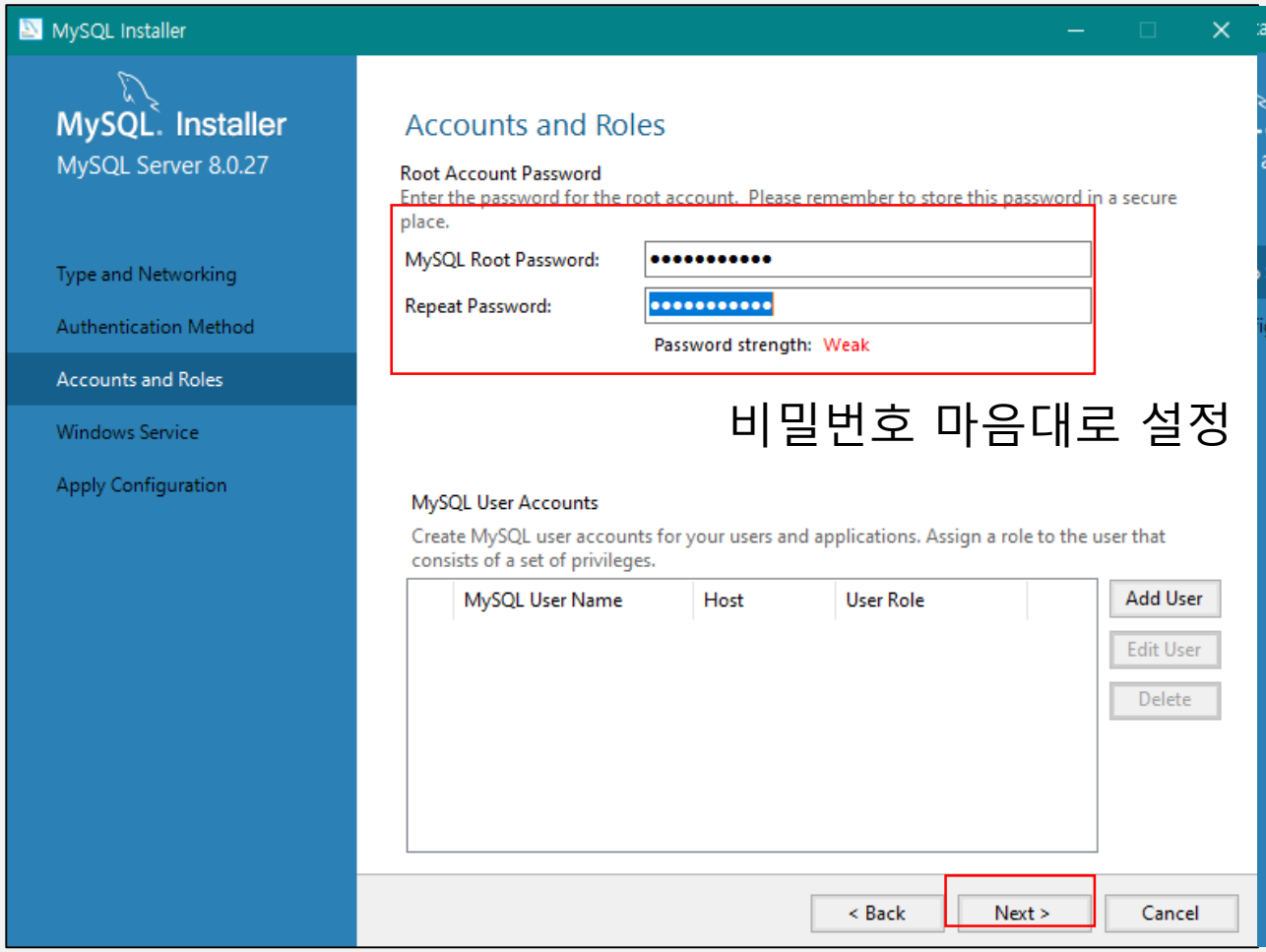


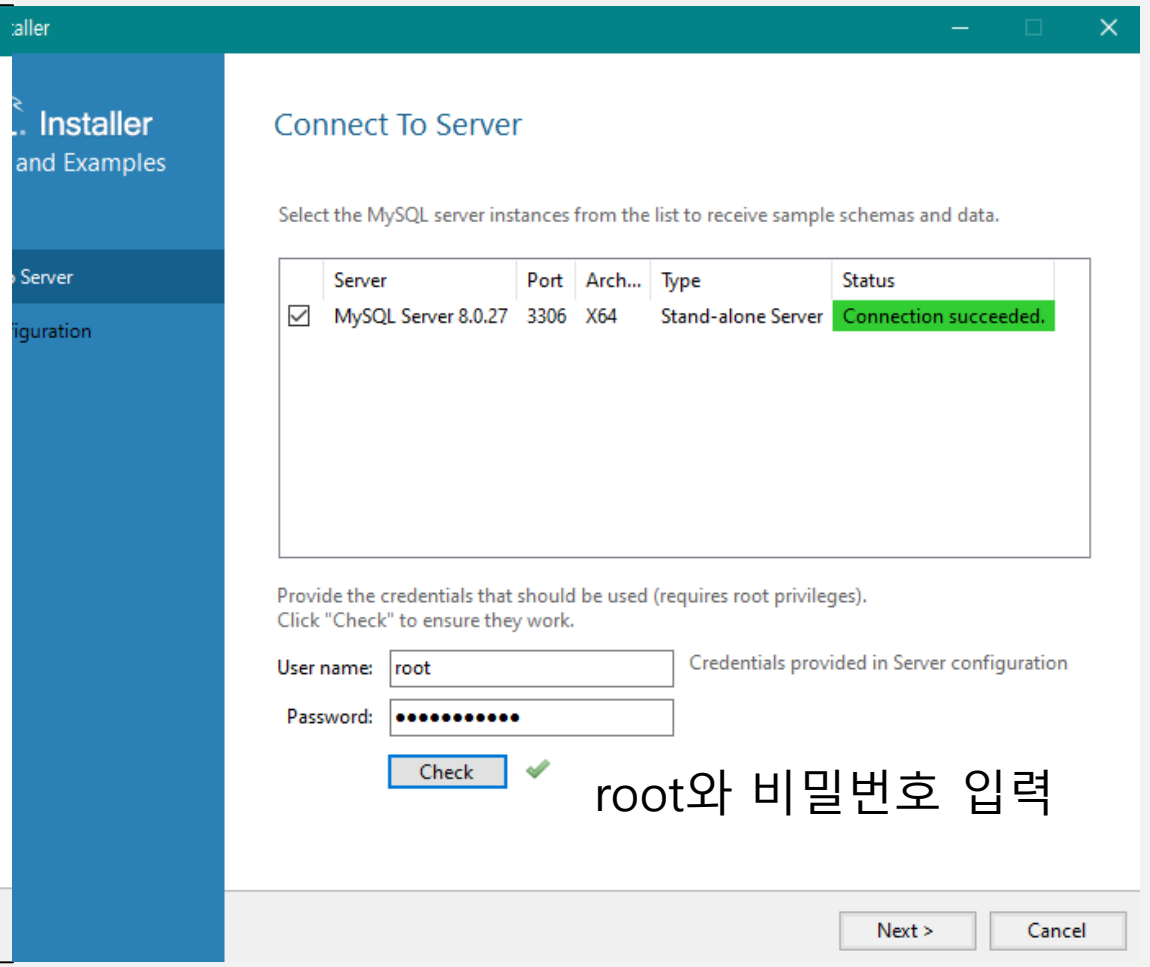


MySQL의 기본 포트설정은 3306
(자신의 컴퓨터에서 안쓰는 포트로 설정해도 됨)

DB

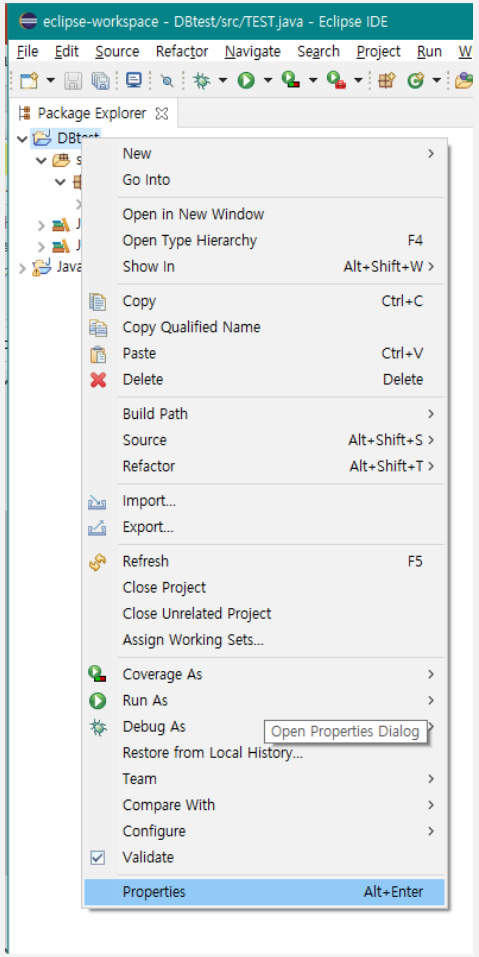
MySQL 설치

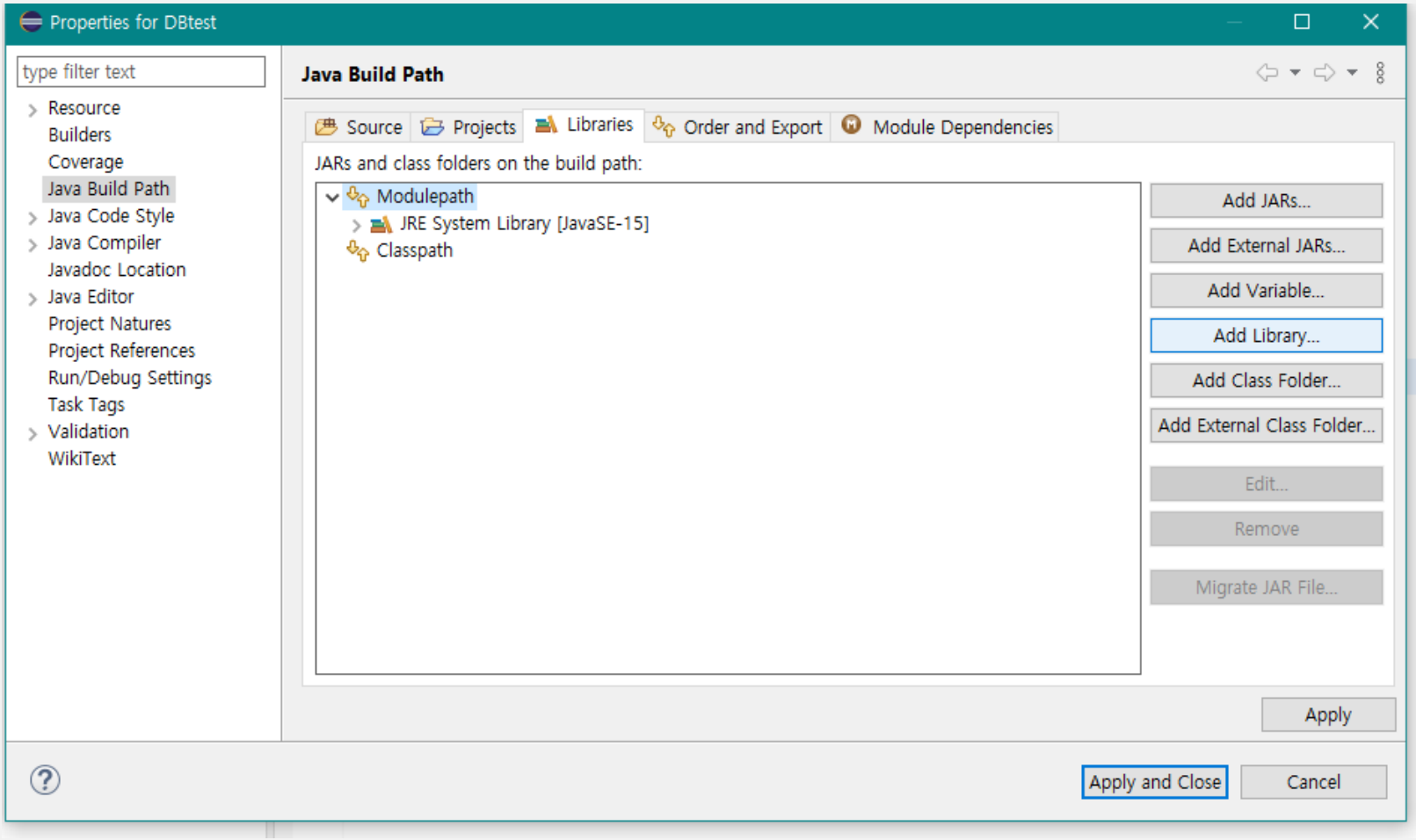




DB

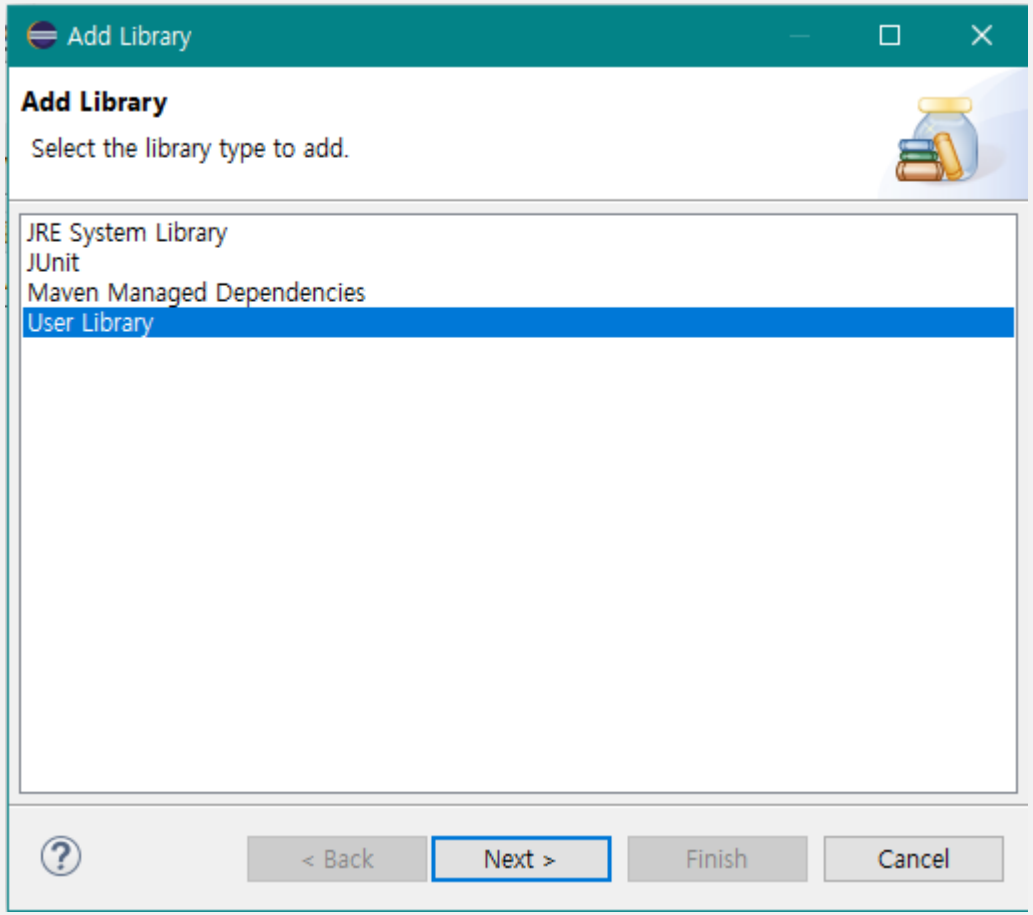
JDBC 연동

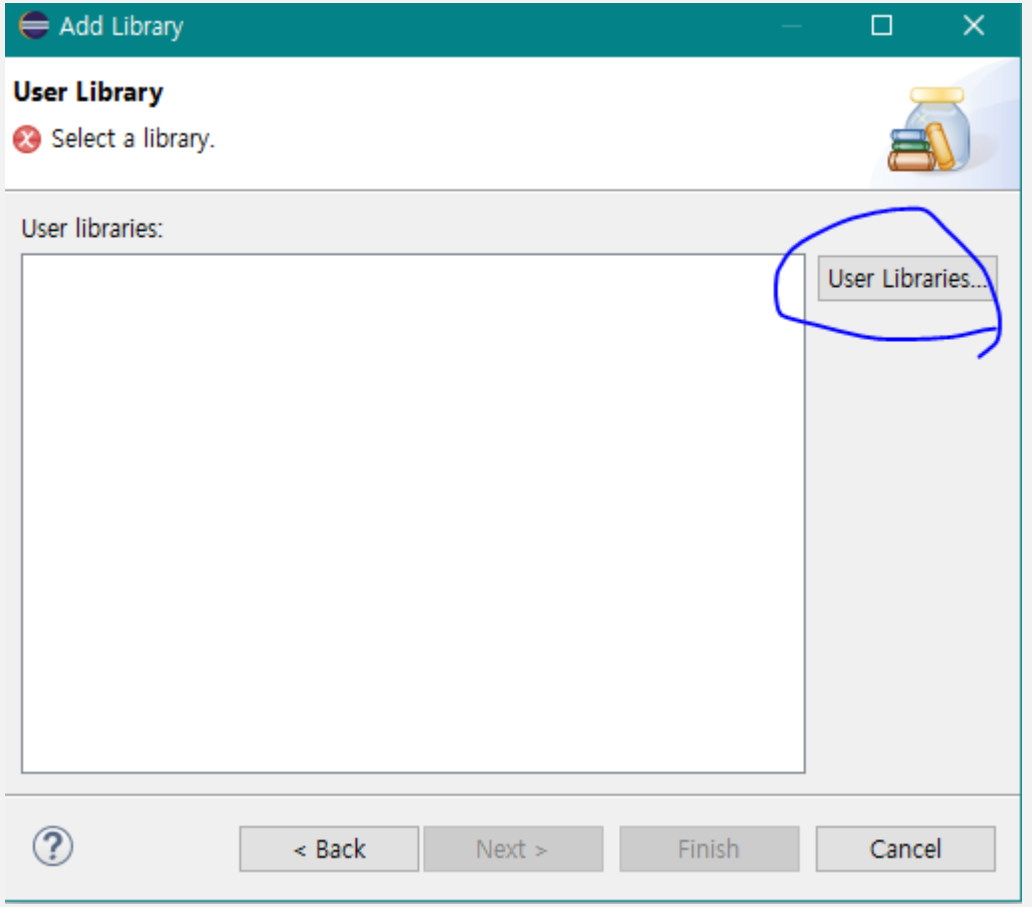




DB

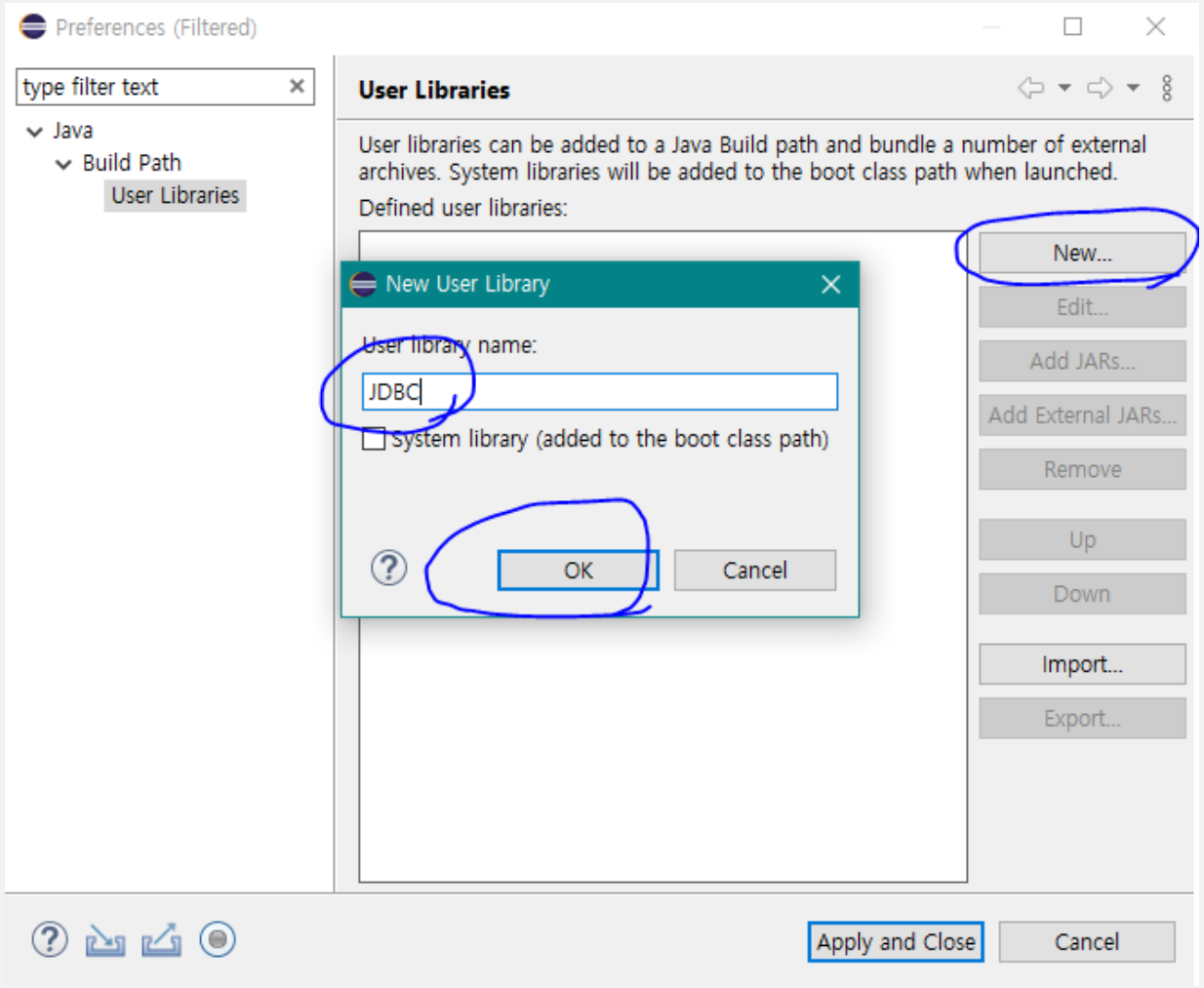
JDBC 연동





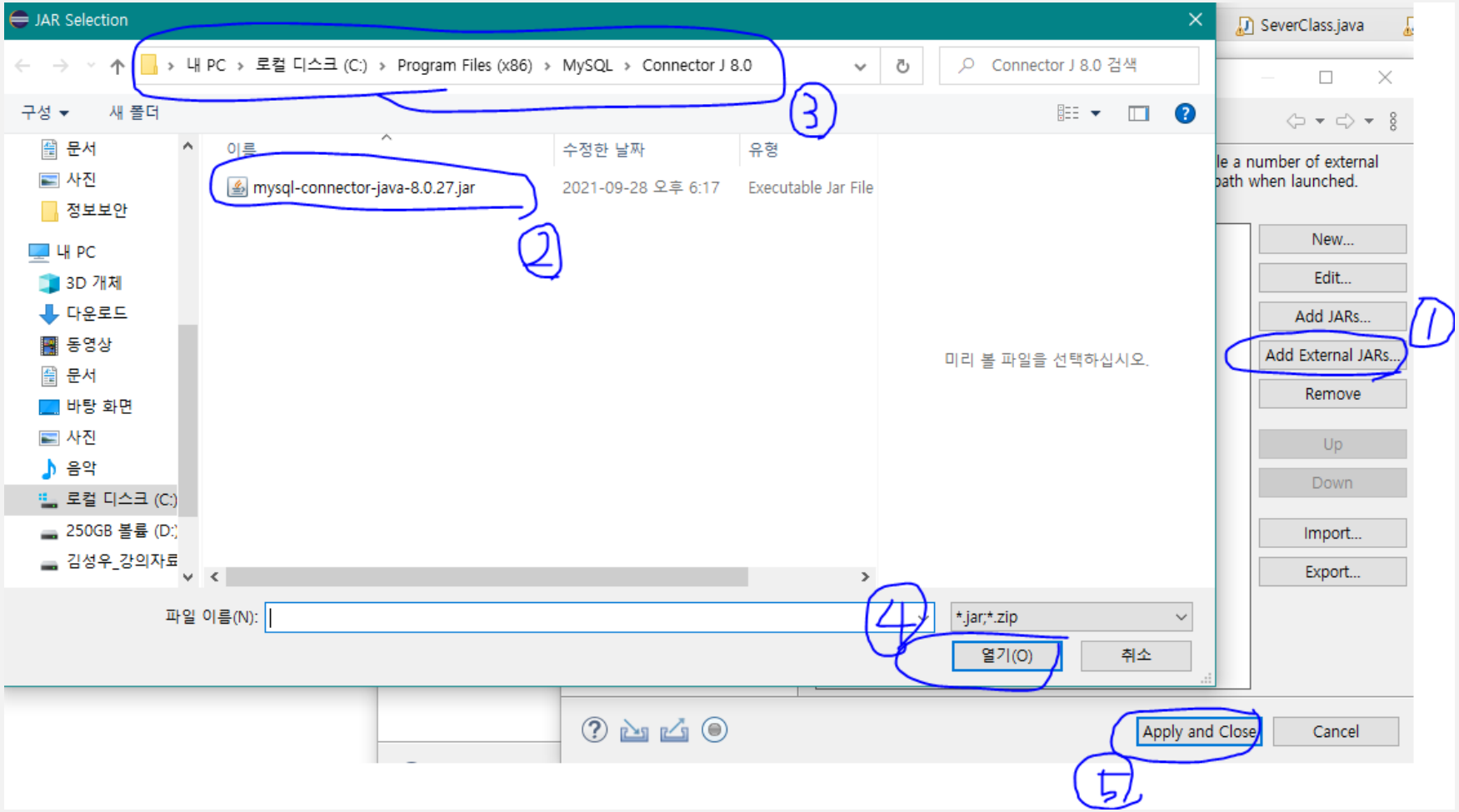
DB

JDBC 연동



DB

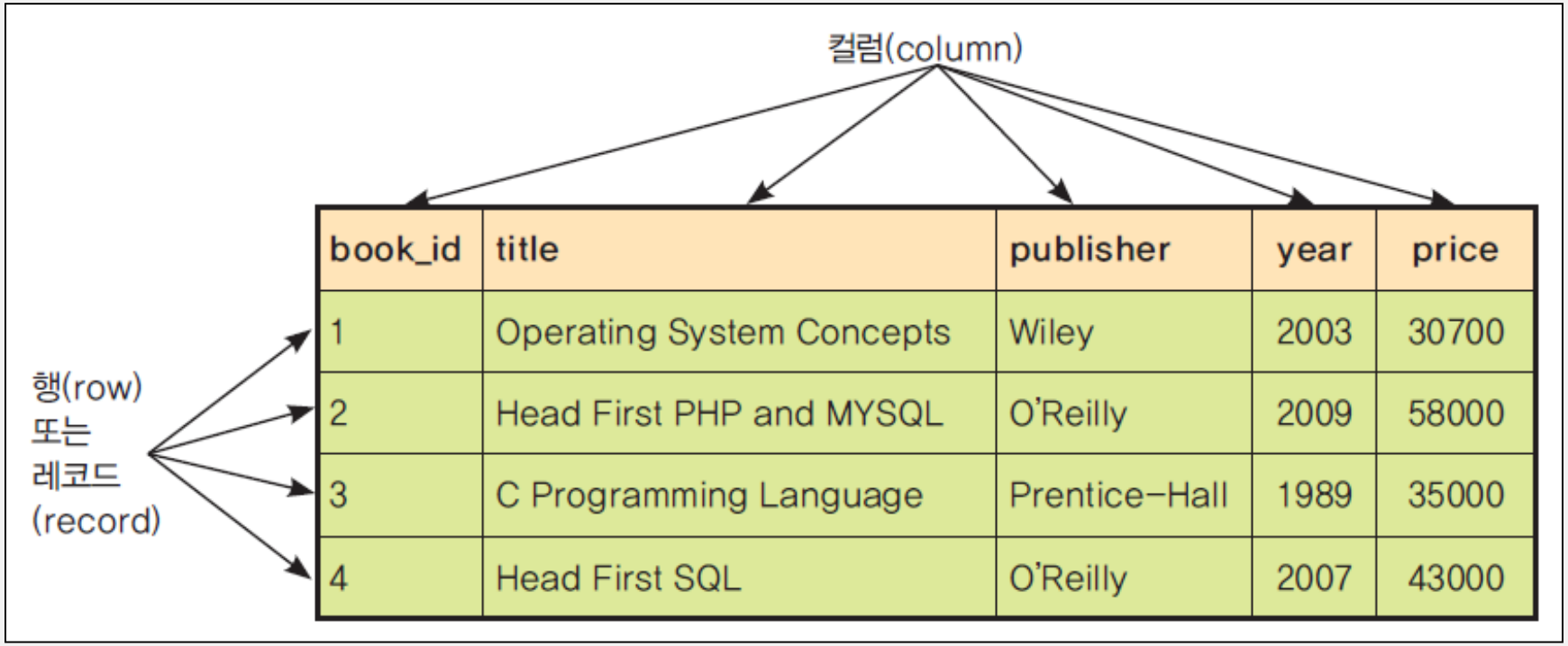
JDBC 연동



DB

테이블

- 레코드(record) : 하나의 행(row)
- 레코드는 여러개의 칼럼(column)으로 이루어짐
- 테이블은 무결성 법칙을 따라서 작성되어야 함
- 주요키(primary key) : 레코드와 레코드를 구분 가능한 하나의 칼럼



book_id	title	publisher	year	price
1	Operating System Concepts	Wiley	2003	30700
2	Head First PHP and MYSQL	O'Reilly	2009	58000
3	C Programming Language	Prentice-Hall	1989	35000
4	Head First SQL	O'Reilly	2007	43000

DB

SQL

- 관계형 데이터베이스에서 사용하기 위하여 설계된 언어

구분	명령어	설명
데이터 정의 명령어 (Data Definition Language)	CREATE	사용자가 제공하는 컬럼 이름을 가지고 테이블을 생성한다. 사용자는 컬럼의 데이터 타입도 지정하여야 한다. 데이터 타입은 데이터베이스에 따라 달라진다. CREATE TABLE은 보통 DML보다 적게 사용된다. 왜냐하면 이미 테이블이 만들어져 있는 경우가 많기 때문이다.
	ALTER	테이블에서 컬럼을 추가하거나 삭제한다.
	DROP	데이블의 모든 레코드를 제거하고 테이블의 정의 자체를 데이터베이스로부터 삭제하는 명령어이다.
	USE	어떤 데이터베이스를 사용하는지를 지정
데이터 조작 명령어 (Data Manipulation Language)	SELECT	데이터베이스로부터 데이터를 쿼리하고 출력한다. SELECT 명령어들은 결과 집합에 포함시킬 컬럼을 지정한다. SQL 명령어 중에서 가장 자주 사용된다.
	INSERT	새로운 레코드를 테이블에 추가한다. INSERT는 새롭게 생성된 테이블을 채우거나 새로운 레코드를 이미 존재하는 테이블에 추가할 때 사용된다.
	DELETE	지정된 레코드를 테이블로부터 삭제한다.
	UPDATE	테이블에서 레코드에 존재하는 값을 변경한다.

DB

데이터베이스 생성하기

- CREATE 구문을 사용하여 테이블 생성

CREATE TABLE 테이블명 (컬럼명1, 자료형1, 컬럼명2, 자료형2, ...)

```
DROP DATABASE book_db;

CREATE DATABASE book_db;

USE book_db;

 CREATE TABLE books(
    -> book_id INT NOT NULL auto_increment,
    -> title VARCHAR(30),
    -> publisher VARCHAR(10),
    -> price INT,
    -> PRIMARY KEY(book_id)
    -> );
```

같은 이름의 db가 있을 수도 있으므로 db삭제

book_db라는 이름의 db 생성

book_db를 사용한다는 의미

book_db 안에 books 이름의 테이블 생성

book_id, title, publisher 와 같은 칼럼들 생성
VARCHAR() : 문자
book_id를 주요키로 설정

DB

레코드 추가하기

- INSERT 구문을 사용하여 테이블 생성

INSERT INTO 테이블명 [(컬럼명1, 컬럼명2, ...)] VALUES (값1, 값2, ...)

```
INSERT INTO books (title, publisher, price)
    -> VALUES('Operationg System Concepts', 'Wiley', 30700);

INSERT INTO books (title, publisher, price)
    -> VALUES('Head First PHP and MYSQL', 'OReilly', 58000)
;
INSERT INTO books (title, publisher, price)
    -> VALUES('C Programming Language', 'Prentice', 35000);

INSERT INTO books (title, publisher, price)
    -> VALUES('Head First SQL', 'OReilly', 43000);
```

DB

레코드 검색/수정/삭제하기

- SELECT / UPDATE / DELETE 구문을 사용하여 테이블 생성

SELECT 컬럼명 FROM 테이블명 [WHERE 조건] [ORDER BY 정렬방식]

UPDATE 테이블명 SET 컬럼명 = 새로운 값, … , [WHERE 조건];

DELETE FROM 테이블명 [WHERE 조건];

SELECT * FROM books;

```
+---------+-----------------------------+-----------+-------+
| book_id | title                       | publisher | price |
+---------+-----------------------------+-----------+-------+
|       1 | Operationg System Concepts  | Wiley     | 30700 |
|       2 | Head First PHP and MYSQL    | OReilly   | 58000 |
|       3 | C Programming Language      | Prentice  | 35000 |
|       4 | Head First SQL              | OReilly   | 43000 |
+---------+-----------------------------+-----------+-------+
4 rows in set (0.00 sec)
```

DB

JDBC를 이용한 데이터베이스 사용 절차

1. URL로 지정된 JDBC 드라이버를 적재

```
Class.forName("com.mysql.cj.jdbc.Driver");
```

2. 데이터베이스 연결 – Connection객체 생성

```
DriverManager.getConnection(url, user, pw);
```

3. SQL 문장 작성 및 전송 – 질의 작업을 처리하는 객체 생성

```
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("SELECT * FROM books");
```

4. 결과 집합 사용 후 연결 해제

```
while(rs.next()) {
    int number = rs.getInt("book_id");
    String name = rs.getString("title");
}
```

DB

JDBC로 데이터베이스에 연결하기

- **URL =>** ***프로토콜 : 서브 프로토콜 : SID***
- **프로토콜 =>** JDBC
- **MySQL의 서브 프로토콜 =>** mysql://DBMS설치IP주소:포트번호
- **SID =>** 데이터베이스의 이름

```
import java.sql.*;

public class TEST {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        String url = "jdbc:mysql://localhost/book_db";
        String user = "root";
        String pw = "password";
        Connection con = null;
```

DB

JDBC로 데이터베이스에 연결하기

```
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            System.out.println("드라이버 적재 성공");

            con = DriverManager.getConnection(url, user, pw);
            System.out.println("데이터베이스 연결 성공");

            Statement stmt = con.createStatement();
        } catch (ClassNotFoundException e) {
            System.out.println("드라이버를 찾을 수 없습니다.");
        } catch (SQLException e) {
            System.out.println("연결에 실패하였습니다.");
        }
    }
}
```

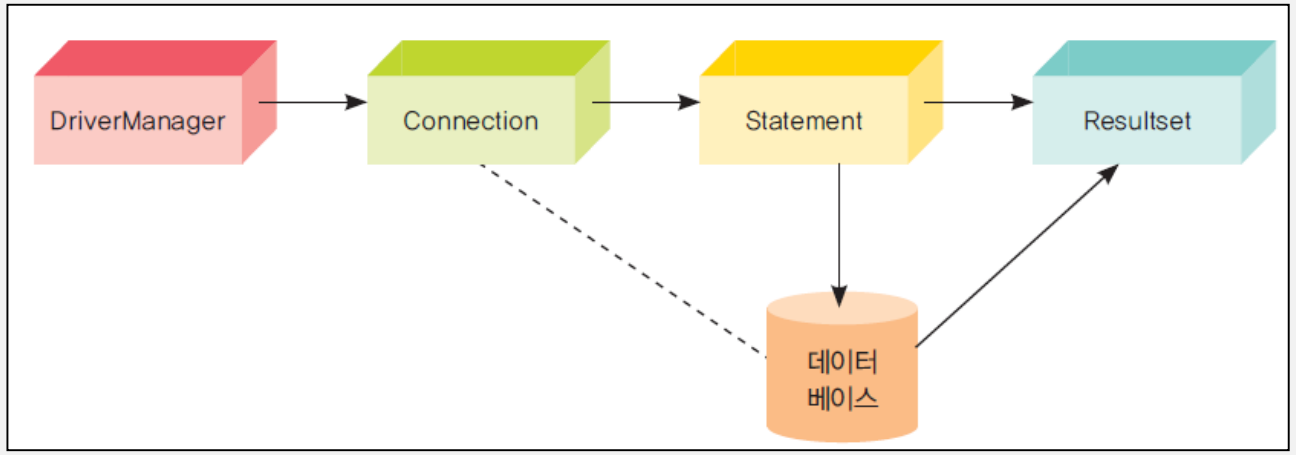
DB

SQL 문장 실행

- Connection, Statement, ResultSet 인터페이스를 사용하여 SQL 문장 실행

```
Statement s = con.createStatement();
String select = “SELECT * FROM books ORDER BY book_id”;
ResultSet rows = s.executeQuery(select);
```

메소드	반환형	담당쿼리
executeQuery()	ResultSet()	select
execute()	int	insert update delete create drop
excuteUpdate()		



DB

결과 집합에서 이동과 처리

- executeQuery 메소드에 의하여 반환된 ResultSet 객체에 쿼리 결과 레코드가 모두 저장되어 있음
- 커서(cursor)를 사용하여 결과 집합에서 레코드를 하나씩 처리
- 결과 집합 (Result Sets) : 쿼리의 조건을 만족하는 레코드들의 집합
- 커서(cursor) : 결과 집합의 레코드들을 포함하고 있는 파일에 대한 포인터. 현재 접근되고 있는 레코드를 가리킴.
- 커서 객체는 previous(), first(), last(), absolute(), relative(), beforeFirst(), afterLast() 등과 같은 이동 메소드를 가짐
- SQL에서는 인덱스 번호가 1부터 시작

커서 이동 예)

```
while ( rows.next() ) {
        //현재 레코드를 처리
        int id = rs.getInt("book_id");
}
```

결과 집합 처리 예) 1- 컬럼명으로, 2- index번호로

```
int id = row.getInt(“id”);
String name = row.getString(“name”)
```

```
int id = row.getInt(1);
String name = row.getString(2)
```

DB

검색결과 출력하기

데이터베이스에서 책을 전부 검색하여 콘솔에 출력

앞의 코드와 동일

```
// TODO Auto-generated method stub
String url = "jdbc:mysql://localhost/book_db";
String user = "root";
String pw = "PASSWORD";
Connection con = null;
```

```
try {
    Class.forName("com.mysql.cj.jdbc.Driver");
    System.out.println("드라이버 적재 성공");

    con = DriverManager.getConnection(url, user, pw);
    System.out.println("데이터베이스 연결 성공");

    Statement stmt = con.createStatement();
    ResultSet rs = stmt.executeQuery("SELECT * FROM books");
    while (rs.next()) {
            int id = rs.getInt("book_id");
            String title = rs.getString("title");
            System.out.println(id + " " + title);
            }
} catch (ClassNotFoundException e) {
    System.out.println("드라이버를 찾을 수 없습니다.");
} catch (SQLException e) {
    System.out.println("연결에 실패하였습니다.");
}
```

DB

데이터베이스 수정, 삭제

```
import java.sql.*;

public class TEST {
    private static Connection makeConnection() {
        String url = "jdbc:mysql://localhost/book_db";
        String id =    id   ;
        String password =       pw      ;
        Connection con = null;
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            System.out.println("드라이버 적재 성공");
            con = DriverManager.getConnection(url, id, password);
            System.out.println("데이터베이스 연결 성공");
        } catch (ClassNotFoundException e) {
            System.out.println("드라이버를 찾을 수 없습니다.");
        } catch (SQLException e) {
            System.out.println("연결에 실패하였습니다.");
        }
        return con;
    }
```

DB

데이터베이스 수정, 삭제

```
private static void addBook(String title, String publisher, int price) {
    Connection con = makeConnection();
    try {
            Statement stmt = con.createStatement();
            String s = "INSERT INTO books (title, publisher, price) VALUES ";
            s += "('" + title + "','" + publisher + "','" + price + "')";
            System.out.println(s);
            int i = stmt.executeUpdate(s);
            if (i == 1)
                System.out.println("레코드 추가 성공");
            else
                System.out.println("레코드 추가 실패");
    }
    catch (SQLException e) {
          System.out.println(e.getMessage());
          System.exit(0);
    }
}

public static void main(String[] args) throws SQLException {
    addBook("Artificial Intellegence", "Addison", 35000);
}
```

```
mysql> select * from books;
+---------+----------------------------+-----------+-------+
| book_id | title                      | publisher | price |
+---------+----------------------------+-----------+-------+
|       1 | Operationg System Concepts | Wiley     | 30700 |
|       2 | Head First PHP and MYSQL   | OReilly   | 58000 |
|       3 | C Programming Language     | Prentice  | 35000 |
|       4 | Head First SQL             | OReilly   | 43000 |
|       5 | Artificial Intellegence    | Addison   | 35000 |
+---------+----------------------------+-----------+-------+
```

DB

Connection Factory

- DB에서 반복적으로 코드를 작성해야 하는 블록을 모듈로 만들어 놓은 것
- 동일한 코드를 찍어낸다고 하여 팩토리(Factory) 패턴이라고 함

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DBConnectionFactory {
    private File file;
    private String driver;
    private String url;
    private String user;
    private String password;
    private Connection connection;

    public DBConnectionFactory() {
        file = new File("setting.txt");
        loadData();
    }
```

```
private void loadData() {
    try {
        BufferedReader fin = new BufferedReader(new FileReader(file));
        driver = fin.readLine();
        url = fin.readLine();
        user = fin.readLine();
        password = fin.readLine();
    }
    catch (Exception e) {
        e.printStackTrace();
    }
}
```

DB

Connection Factory

```
    private Connection getConnection() {
        try {
            Class.forName(driver);
        }catch (Exception e) {
            e.printStackTrace();
            return null;
        }

        try {
            connection = DriverManager.getConnection(url, user, password);
        }catch (Exception e) {
            e.printStackTrace();
            return null;
        }

        return connection;
    }
```

```
    public static void main(String[] args) {
        Connection connection = new DBConnectionFactory().getConnection();
        System.out.println("connection = " + connection);
        try {
            connection.close();
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

}
```

수고하셨습니다.