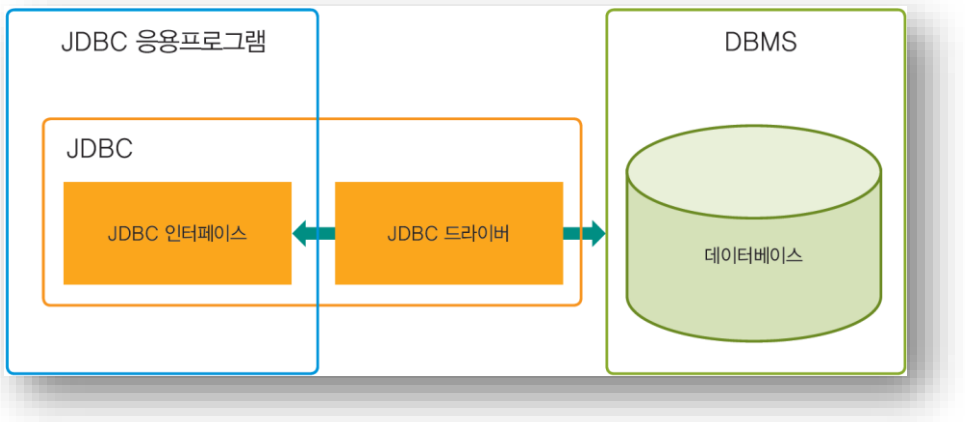
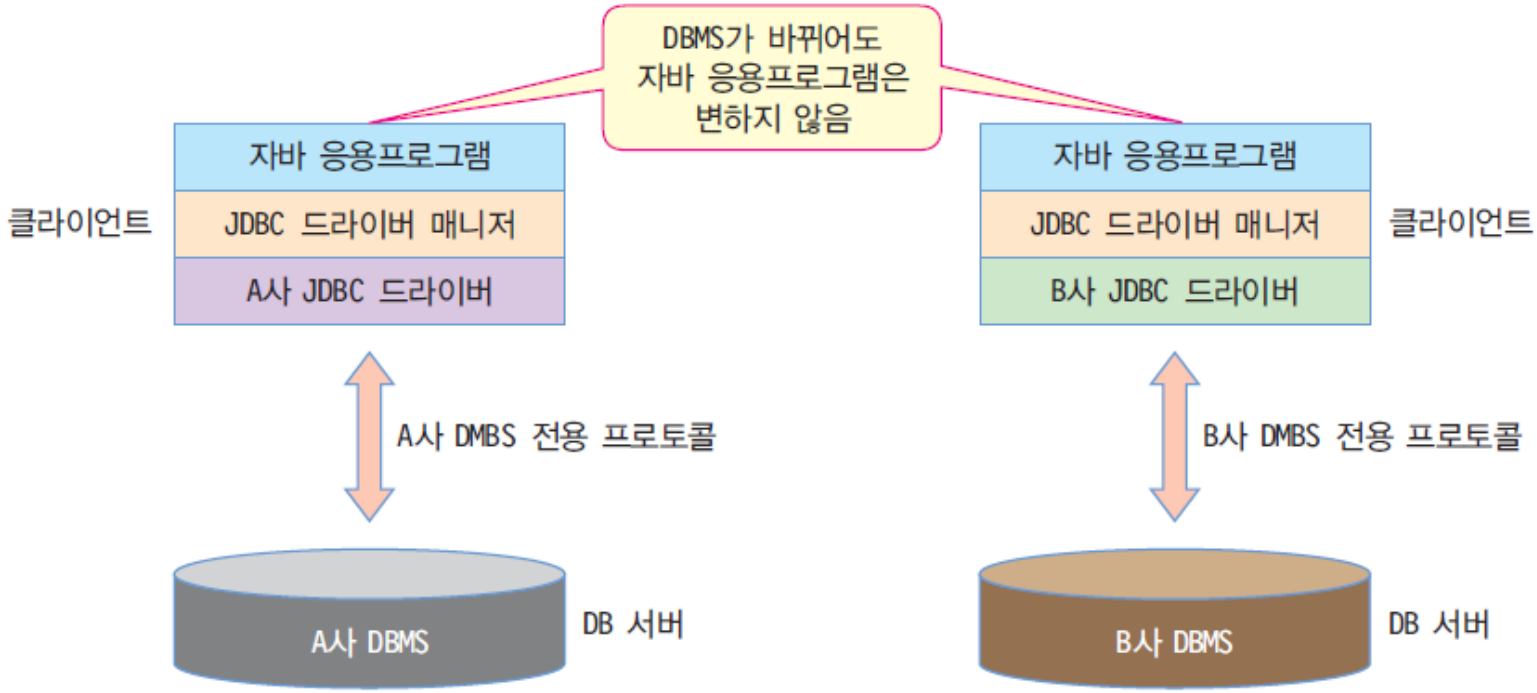

# JDBC 프로그래밍

# JDBC란?

- 자바 언어로 데이터베이스 프로그래밍을 하기 위한 라이브러리
- 특정한 DBMS에 종속되지 않는 관련 API(Application Programming Interface)를 제공



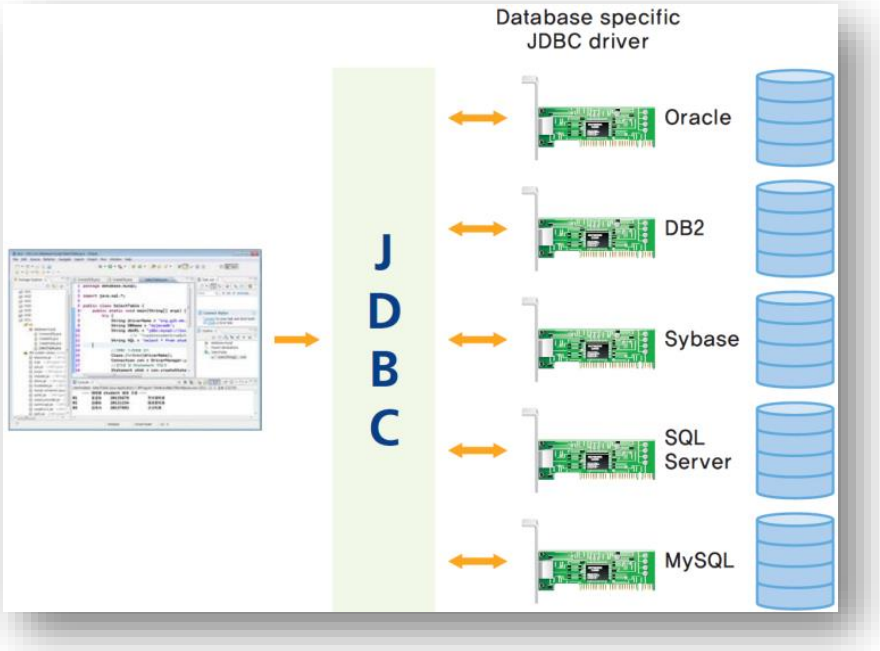
# JDBC 구조




- JDBC 드라이버 매니저
  - 자바 API에서 지원하며 DBMS를 접근할 수 있는 JDBC 드라이버 로드
- JDBC 드라이버
  - DBMS마다 고유한 JDBC 드라이버 제공, JDBC 드라이버와 DBMS는 전용 프로토콜로 데이터베이스 처리
- DBMS
  - 데이터베이스 관리 시스템. 데이터베이스 생성·삭제, 데이터 생성·검색·삭제 등 전담 소프트웨어 시스템

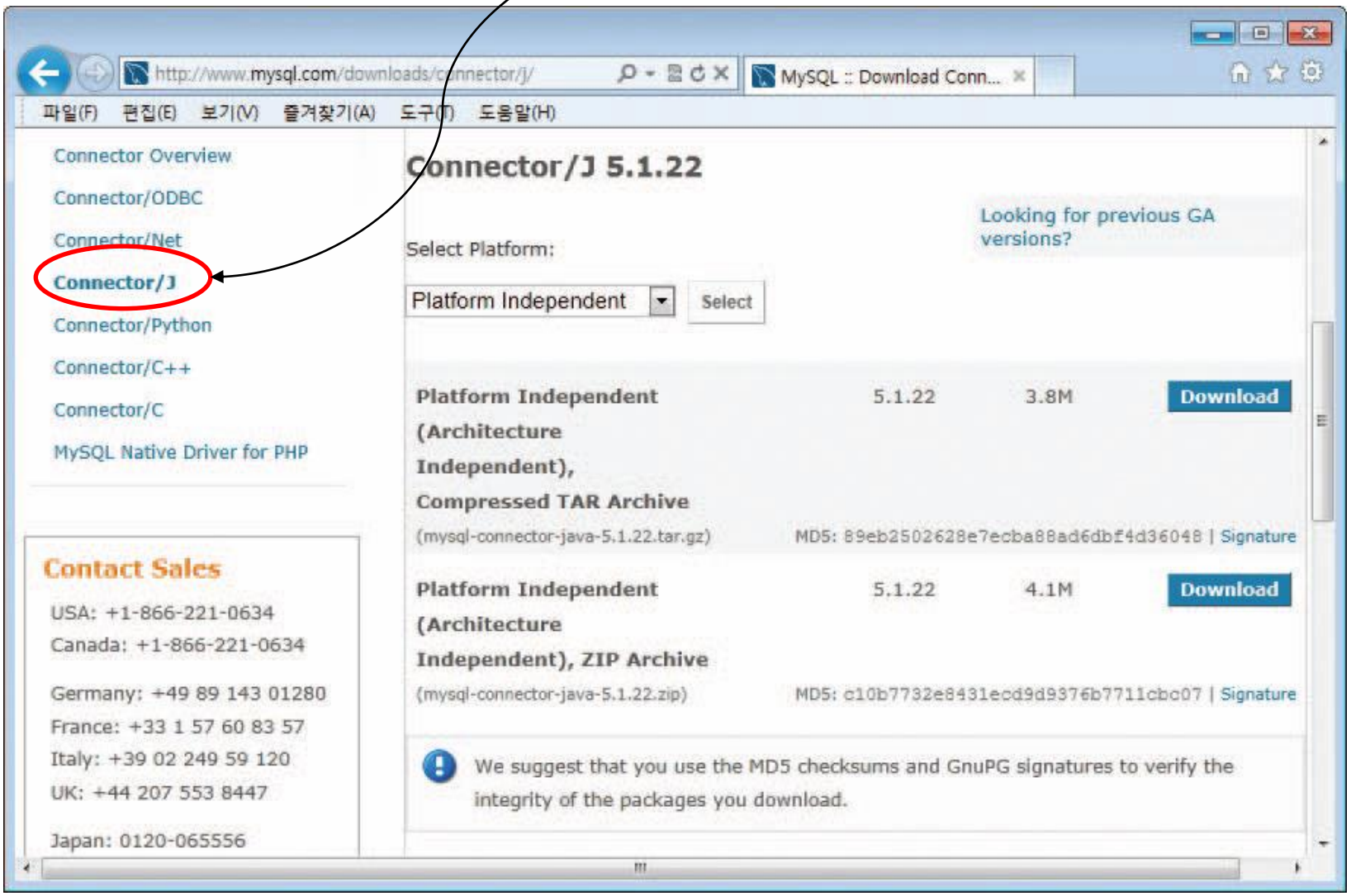
# JDBC 역할

- 다양한 DBMS에 독립적으로 데이터베이스 프로그래밍을 가능하도록 하는 API(application programming interfaces) 규격
  - 오라클(ORACLE), MySQL, SQLServer, DB2 등 어떤 DBMS를 사용하던지 소스의 수정을 최소화하여 바로 실행
  - JDBC와 함께 JDBC 드라이버(JDBC Driver)도 필요



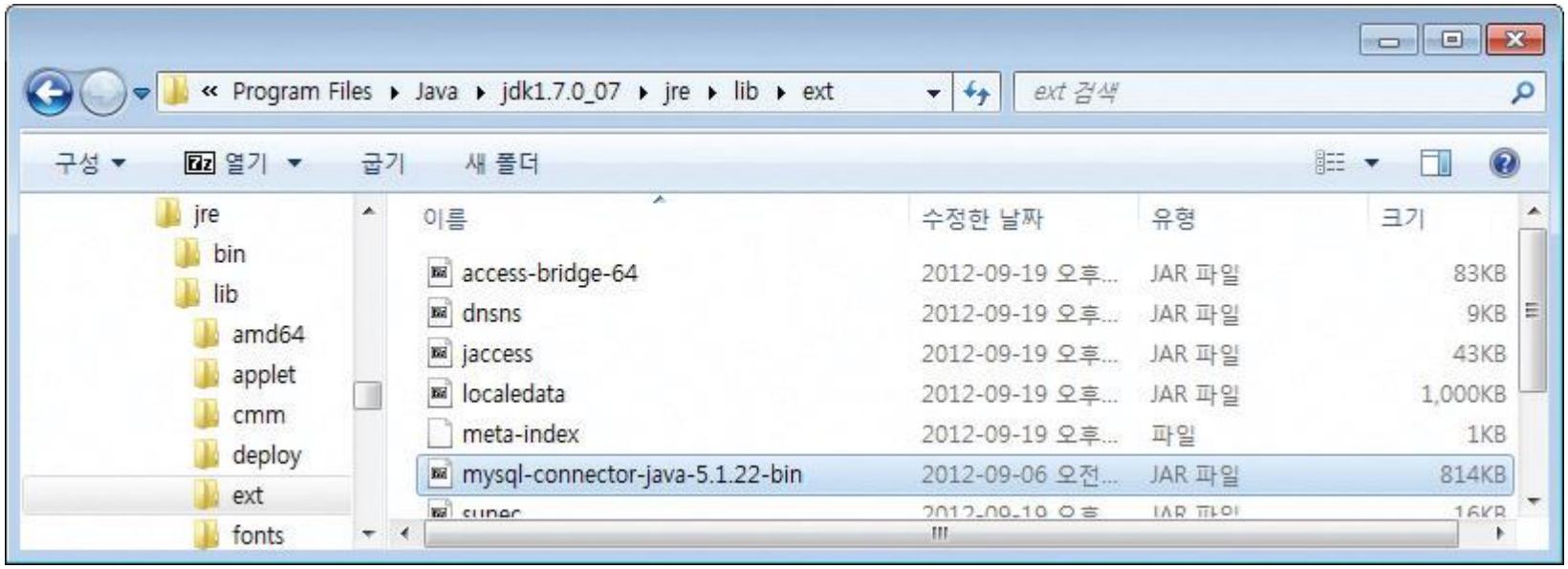
# JDBC 드라이버 설치

MySQL 다운로드 페이지에서
Java용 Connector 선택

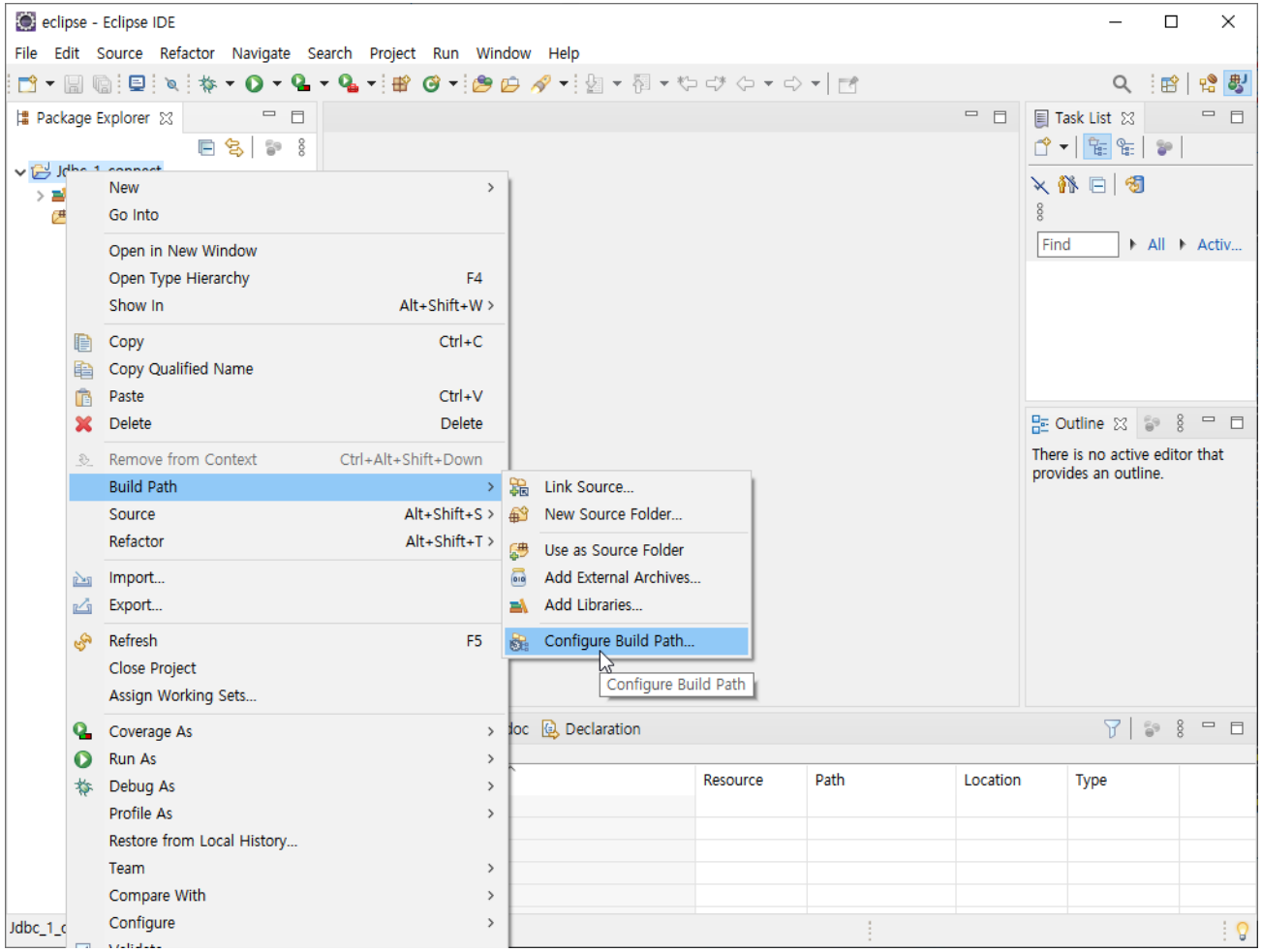


# JDBC 드라이버 설치

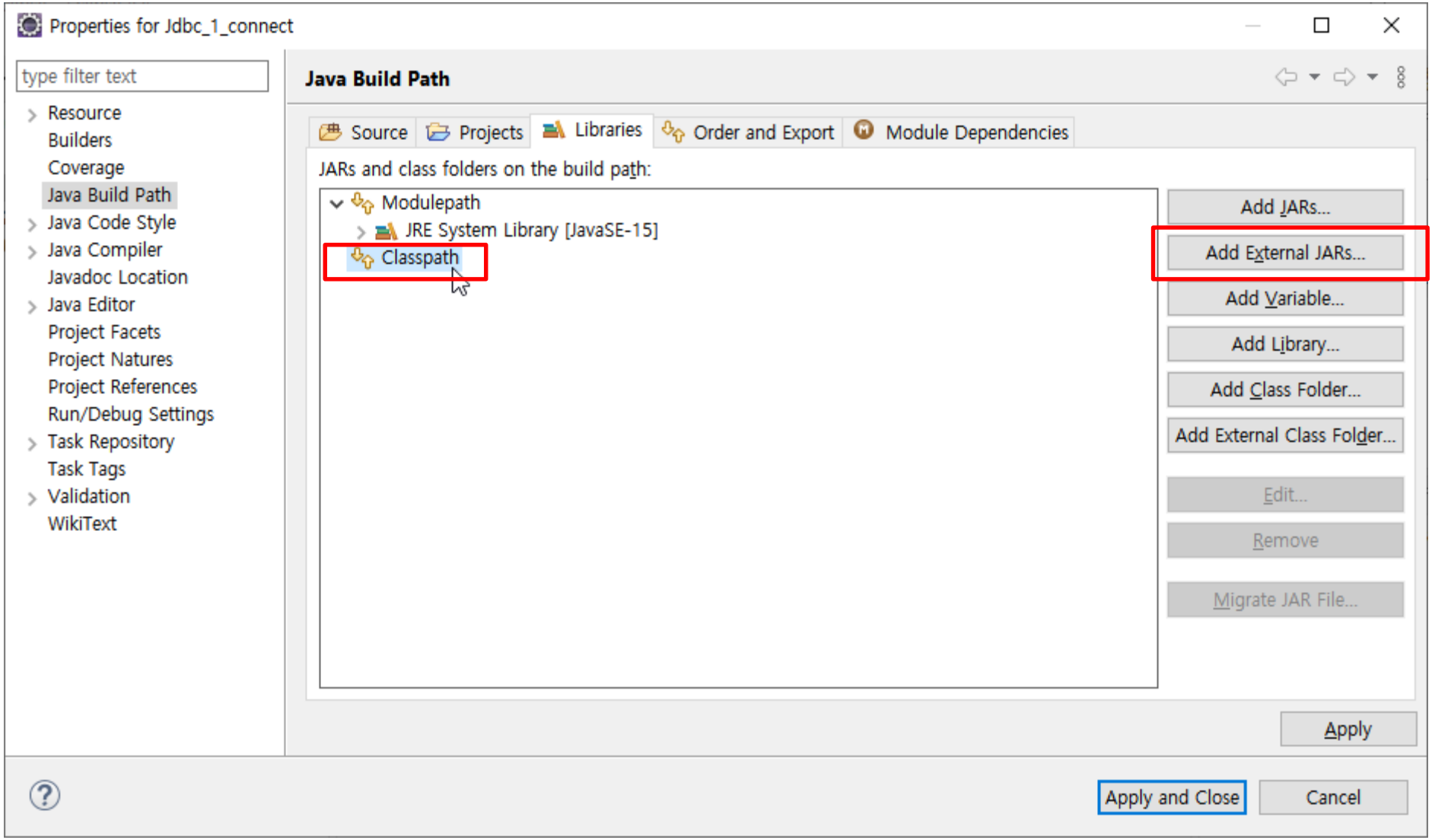
- Java용 드라이버 복사
  - mysql-connector-java-5.1.46.JAR 파일을 JDK 설치 디렉터리 밑의 JRE₩LIB₩EXT 디렉터리에 복사
  - JRE만 설치한 경우는 JRE 설치 디렉터리 밑의 LIB₩EXT 디렉터리에 복사



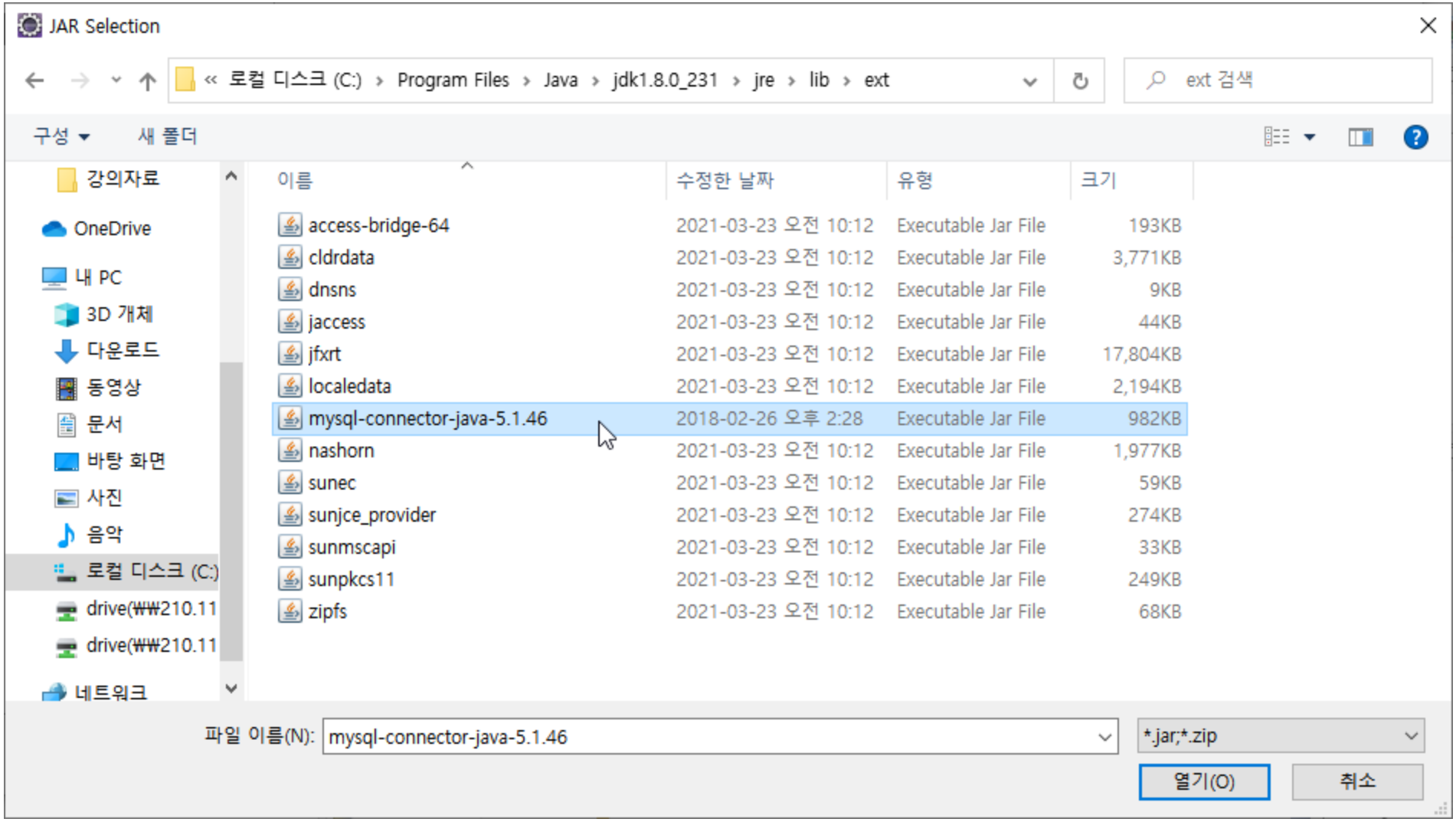
# 이클립스 드라이버 설정


eclipse - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Task List
Find All Activ...
Outline
There is no active editor that provides an outline.
New
Go Into
Open in New Window
Open Type Hierarchy F4
Show In Alt+Shift+W
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Remove from Context Ctrl+Alt+Shift+Down
Build Path
Source Alt+Shift+S
Refactor Alt+Shift+T
Import...
Export...
Refresh F5
Close Project
Assign Working Sets...
Coverage As
Run As
Debug As
Profile As
Restore from Local History...
Team
Compare With
Configure
Link Source...
New Source Folder...
Use as Source Folder
Add External Archives...
Add Libraries...
Configure Build Path...
Configure Build Path
Declaration
Resource Path Location Type

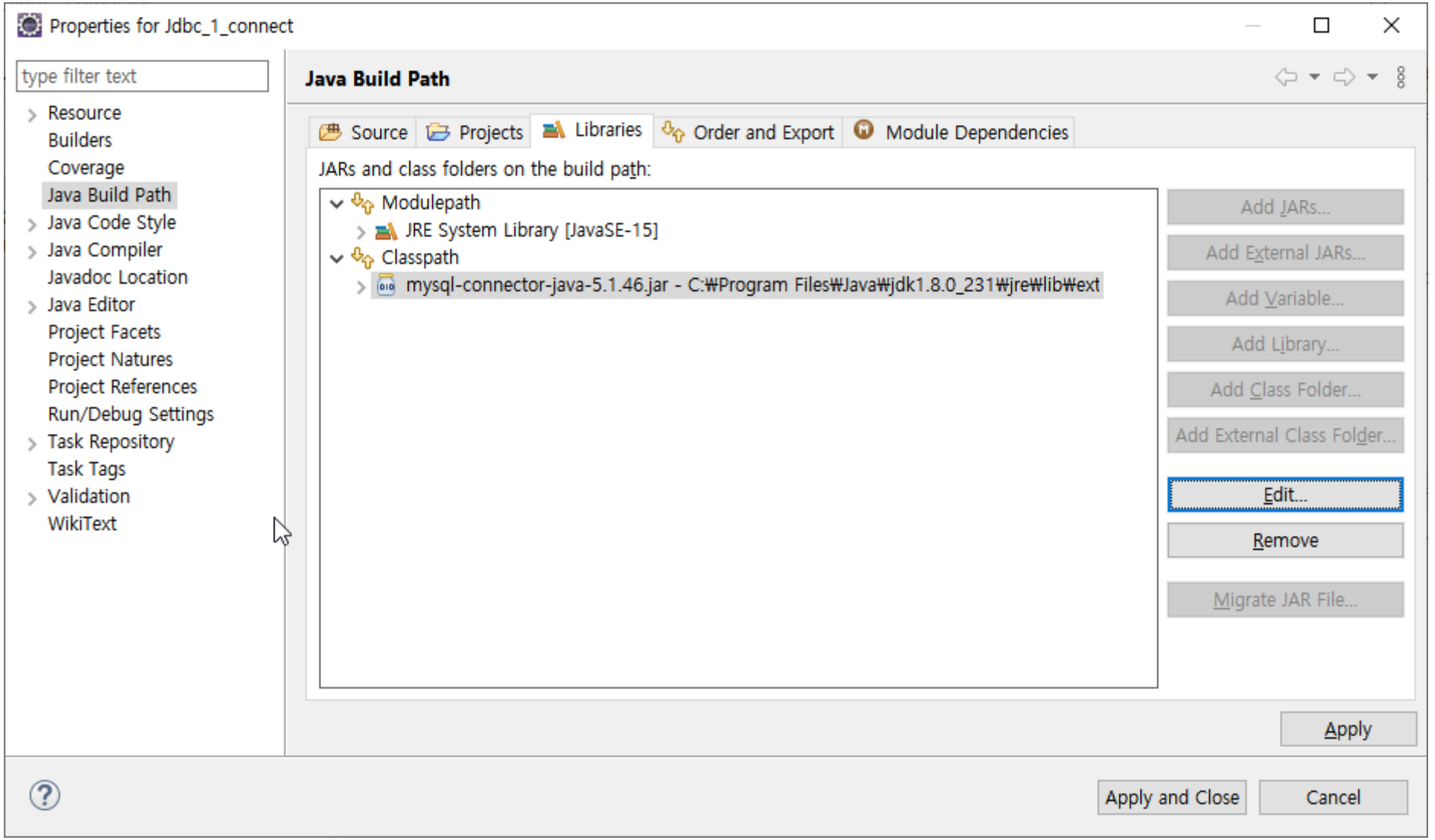
# 이클립스 드라이버 설정


Properties for Jdbc_1_connect
type filter text
Resource
Builders
Coverage
Java Build Path
Java Code Style
Java Compiler
Javadoc Location
Java Editor
Project Facets
Project Natures
Project References
Run/Debug Settings
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-15]
Classpath
Add JARs...
Add External JARs...
Add Variable...
Add Library...
Add Class Folder...
Add External Class Folder...
Edit...
Remove
Migrate JAR File...
Apply
Apply and Close
Cancel

# 이클립스 드라이버 설정


JAR Selection
« 로컬 디스크 (C:) › Program Files › Java › jdk1.8.0_231 › jre › lib › ext
ext 검색
구성
새 폴더
강의자료
OneDrive
내 PC
3D 개체
다운로드
동영상
문서
바탕 화면
사진
음악
로컬 디스크 (C:)
drive(₩₩210.11
drive(₩₩210.11
네트워크
이름
수정한 날짜
유형
크기
access-bridge-64 2021-03-23 오전 10:12 Executable Jar File 193KB
cldrdata 2021-03-23 오전 10:12 Executable Jar File 3,771KB
dnsns 2021-03-23 오전 10:12 Executable Jar File 9KB
jaccess 2021-03-23 오전 10:12 Executable Jar File 44KB
jfxrt 2021-03-23 오전 10:12 Executable Jar File 17,804KB
localedata 2021-03-23 오전 10:12 Executable Jar File 2,194KB
mysql-connector-java-5.1.46 2018-02-26 오후 2:28 Executable Jar File 982KB
nashorn 2021-03-23 오전 10:12 Executable Jar File 1,977KB
sunec 2021-03-23 오전 10:12 Executable Jar File 59KB
sunjce_provider 2021-03-23 오전 10:12 Executable Jar File 274KB
sunmscapi 2021-03-23 오전 10:12 Executable Jar File 33KB
sunpkcs11 2021-03-23 오전 10:12 Executable Jar File 249KB
zipfs 2021-03-23 오전 10:12 Executable Jar File 68KB
파일 이름(N): mysql-connector-java-5.1.46
*.jar;*.zip
열기(O)
취소

# 이클립스 드라이버 설정


Properties for Jdbc_1_connect
type filter text
Resource
Builders
Coverage
Java Build Path
Java Code Style
Java Compiler
Javadoc Location
Java Editor
Project Facets
Project Natures
Project References
Run/Debug Settings
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-15]
Classpath
mysql-connector-java-5.1.46.jar - C:₩Program Files₩Java₩jdk1.8.0_231₩jre₩lib₩ext
Add JARs...
Add External JARs...
Add Variable...
Add Library...
Add Class Folder...
Add External Class Folder...
Edit...
Remove
Migrate JAR File...
Apply
Apply and Close
Cancel

# JDBC 데이터베이스 프로그래밍 절차

- 1 단계 : java.sql 패키지 임포트
- 2 단계 : DBMS 드라이버를 로드
- 3 단계 : DB에 연결 - Connection 객체를 생성
- 4 단계 : SQL문 객체 생성 및 전송 - Statement 객체를 생성
- 5 단계 : SQL문 결과 받기 : ResultSet 객체를 생성
- 6 단계 : 모든 객체를 해제

# JDBC 데이터베이스 프로그래밍 절차

- 1. java.sql 패키지 임포트

```
import java.sql.*;

public class JdbcDriver {
}
```

# JDBC 데이터베이스 프로그래밍 절차

- 2. DBMS 드라이버 로드

```
try {
    Class.forName("com.mysql.jdbc.Driver");
} catch (ClassNotFoundException e) {
    e.printStackTrace();
}
```

  - Class.forName()은 동적으로 자바 클래스 로딩
  - MySQL의 JDBC 드라이버 클래스인 *com.mysql.jdbc.Driver* 로드
  - 드라이버의 클래스 이름은 DB의 드라이버마다 다를 수 있으므로 JDBC 드라이버 문서 참조할 것
  - 자동으로 드라이버 인스턴스를 생성하여 DriverManager에 등록
  - 해당 드라이버가 없으면 ClassNotFoundException 발생
    - 철자 틀린것 없는 지 확인
    - build path에 드라이버 등록했는지 확인

| DBMS 종류 | JDBC 드라이버 로드 문장 |
|---|---|
| ORACLE | Class.forName("oracle.jdbc.driver.OracleDriver"); |
| MS SQLServer | Class.forName("com.microsoft.jdbc.sqlserver.SQLServerDriver"); |
| mSQL | Class.forName("com.imaginay.sql.msql.MsqlDriver"); |
| MySQL | Class.forName("org.gjt.mm.mysql.Driver");<br>Class.forName("com.mysql.jdbc.Driver "); |
| ODBC | Class.forName("sun.jdbc.odbc.JdbcOdbcDriver"); |

# JDBC 데이터베이스 프로그래밍 절차

```
import java.sql.*;

public class JdbcDriver {

    public static void main(String[] args) {

        try {

            Class.forName("com.mysql.jdbc.Driver");
            System.out.println("JDBC driver load success");
        } catch (ClassNotFoundException e) {
            System.out.println("JDBC driver load fail");
        }

    }

}
```

# JDBC 데이터베이스 프로그래밍 절차

- 3. DB에 연결
  - DriverManager : JDBC 드라이버에 연결시켜주는 클래스

| 메소드 | 설명 |
|---|---|
| Connection getConnection(String url, String id, String pw)) | 해당 URL로 DB에 연결 |

- URL 구조

jdbc protocol : subprotocol: ip:port/dbname


```
try {
    Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/jdbc_db?useSSL=false",
"root", "111113333");
} catch (SQLException e) {
    e.printStackTrace();
}
```

# JDBC 데이터베이스 프로그래밍 절차

```
import java.sql.*;

public class JdbcConnect {


    public static void main(String[] args) {

        Connection conn=null;
        String driverName = "com.mysql.jdbc.Driver";
        String DBName = "jdbc_db";
        String dbURL = "jdbc:mysql://localhost:3306/" + DBName;
        String sslStr="?useSSL=false";

        try {

            Class.forName("com.mysql.jdbc.Driver");
            System.out.println("JDBC driver load success");

            conn = DriverManager.getConnection(dbURL+sslStr
                    , "root","1111133333");
            System.out.println("DB connection success");
        } catch (ClassNotFoundException e) {
            System.out.println("JDBC driver load fail !!");
        } catch (SQLException e) {
            System.out.println("DB connection fail !!");
        }
```

DB URL : "jdbc:mysql://localhost:3306/jdbc_db?useSSL=false"

# JDBC 데이터베이스 프로그래밍 절차

```
finally
        {
            try
            {
                if(conn!=null) {
                    conn.close();
                    System.out.println("DB connection close success");
                }
            }
            catch (SQLException e) {
                System.out.println("DB connection close exception !!");
            }
        }

    }

}
```

# 실습(DB연결)

- bookstore_생성 후 연결, 해제

# JDBC 데이터베이스 프로그래밍 절차

- 4. SQL문 설정 및 전송 객체
  - 4-1. Stetement 인터페이스
    - SQL문 설정 및 전송
    - Connection 인터페이스의 createStatement()를 통해 얻음

```
Connection con = DriverManager.getConnection(dbURL+sslStr, "root", "1111133333");

Statement stmt = con.createStatement();

stmt.executeUpdate("insert into User values ('apple', 'orange', 'melon');");
```

| 메소드 | 설명 |
|---|---|
| ResultSet executeQuery(String sql) | 하나의 ResultSet을 만드는 SQL문에서 사용<br>executeQyery 메소드는 ResultSet 객체를 리턴<br>주로 SELECT문을 이용하는 조회에서 사용됨. |
| int executeUpdate(String sql) | INSERT, UPDATE, DELETE 등 (DML), CREATE, DROP 등(DDL)문들을 실행하는데 사용 |
| void close() | Statement 객체의 데이터베이스와 JDBC 리소스를 즉시 반환 |

데이터 검색(select) : executeQuery()
그이외 : executeUpdate()

# JDBC 데이터베이스 프로그래밍 절차

- 4-2. PreparedStatement 인터페이스
  - SQL문의 구조는 Statement와 동일하나 SQL 문의 값들을 변수로 처리함
  - 재사용성 , 보안 및 속도 향상

Statement

```
Connection con = DriverManager.getConnection()

Statement stmt = con.createStatement();

stmt.executeUpdate("insert into User values ('apple', 'orange', 'melon',20000);");
```

PreparedStatement

```
String sql = "insert into user values (?, ?, ?,?)";

con = DriverManager.getConnection()
pstmt = con.prepareStatement(sql);
pstmt.setString(1,“apple”);
pstmt.setString(2,“orange");
pstmt.setString(3,“melon");
pstmt.setInt(4,20000);
retval=pstmt.executeUpdate();
```

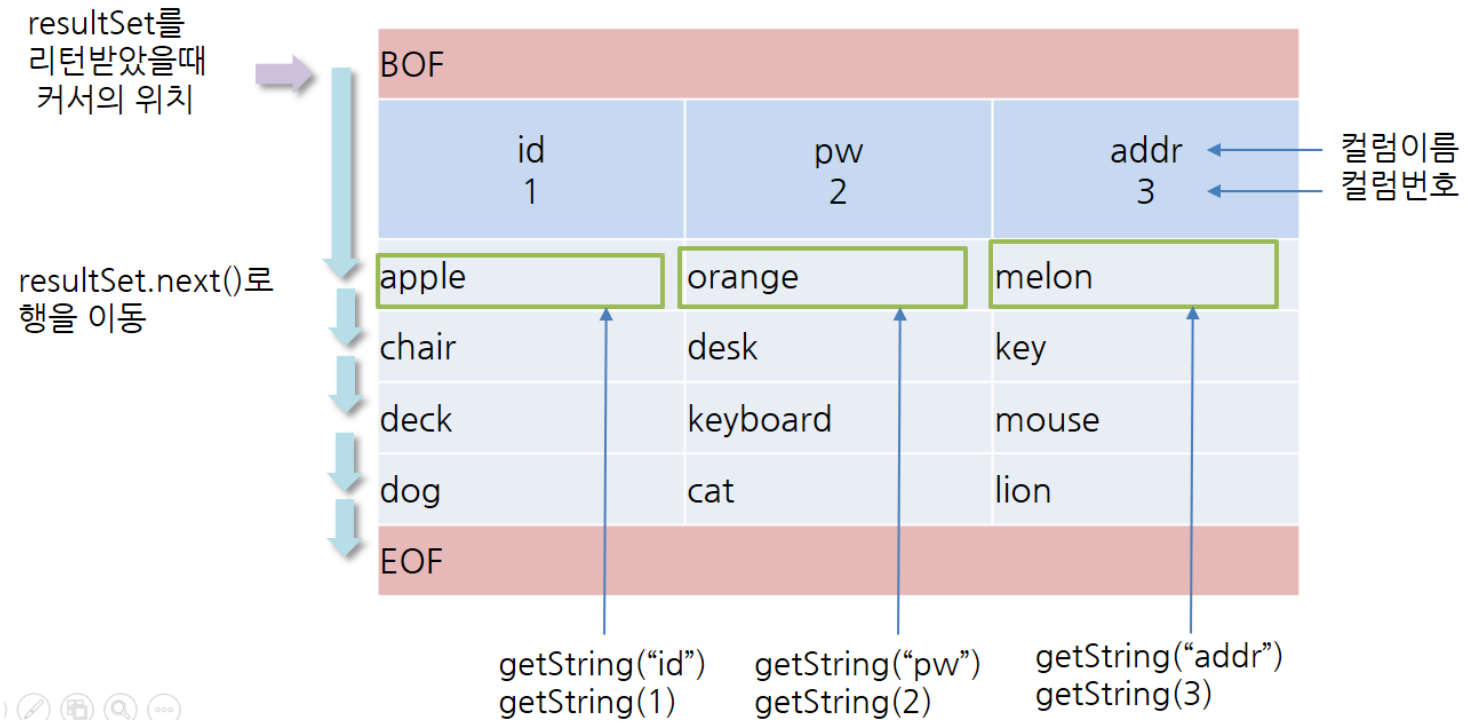
# JDBC 데이터베이스 프로그래밍 절차

- 5. SQL문 결과 받기
  - ResultSet 인터페이스
    - 검색시 SQL문 실행 결과가 저장됨

| 메소드 | 설명 |
|---|---|
| boolean first() | 커서를 첫 번째 행으로 이동 |
| boolean last() | 커서를 마지막 행으로 이동 |
| boolean next() | 커서를 다음 행으로 이동 |
| boolean previous() | 커서를 이전 행으로 이동 |
| boolean absolute(int row) | 커서를 지정된 행으로 이동 |
| boolean isFirst() | 첫 번째 행이면 true 반환 |
| boolean isLast() | 마지막 행이면 true 반환 |
| void close() | ResultSet 객체의 데이터베이스와 JDBC 리소스를 즉시 반환 |
| Xxx getXxx(String columnLable) | Xxx는 해당 데이터 타입을 나타내며 현재 행에서 지정된 열 이름에 해당하는 데이터를 반환. 예를 들어, int형 데이터를 읽는 메소드는 getInt()이다. |
| Xxx getXxx(int columnIndex) | Xxx는 해당 데이터 타입을 나타내며 현재 행에서 지정된 열 인덱스에 해당하는 데이터를 반환. 예를 들어, int형 데이터를 읽는 메소드는 getInt()이다. |

# JDBC 데이터베이스 프로그래밍 절차

- 실질적으로 질의 결과의 자료가 있는 영역과 함께 BOF, EOF 제공
  - 첫 행 자료 이전 : (Before the First Row)에BOF(Begin Of File)
  - 마지막 행 자료 이후 : (After the Last Row)에 EOF(End Of File)
- 각각의 행에서 각 칼럼은칼럼이름 또는 번호 순으로 식별
  - 번호는 1번부터 시작
- ResultSet에서 모든 데이터를 다 읽어들인 후에는 close()를 호출하여 자원 해제



# JDBC 데이터베이스 프로그래밍 절차

- 검색된 데이터의 사용

```
while (result.next()) {
        System.out.print(result.getString(1) + " ");
        System.out.print(result.getString(2) + " ");
        System.out.println(result.getString(3));
}
```

# JDBC 데이터베이스 프로그래밍 절차

- 6. 객체 해제
  - 객체가 만들어지는 순서의 역순으로 해제
    - 생성 순서
      1. Connection
      2. Statement or PreparedStatement
      3. ResultSet
    - 해제순서
      1. ResultSet
      2. Statement or PreparedStatement
      3. Connection
  - 해제시 각 객체 마다 예외처리 구문을 넣어 줘야 함

```
if (stmt != null || con != null) {
    try{
        stmt.close();
        con.close();
    }
    catch(SQLException ex) {
    }
}
```

```
if (stmt != null) {
    try{
        stmt.close();
    }
    catch(SQLException ex) {
    }
}
if (con != null) {
    try {
        con.close();
    }
    catch(SQLException ex) {
    }
}
```

# 레코드 삽입

- 테이블을 생성하는 SQL 문장 create
  - 형식 : create table 테이블명(테이블내용)
    - 테이블 User
      - 필드명 : id, pw, addr, point(주 키는 id)

| Column Name | Datatype | PK | NN |
|---|---|---|---|
| id | VARCHAR(20) | ☑ | ☑ |
| pw | VARCHAR(20) | ☐ | ☑ |
| addr | VARCHAR(45) | ☐ | ☑ |
| cash | INT | ☐ | ☑ |

- 테이블에 레코드를 삽입하는 insert 문장

```
insert into User values ('apple','orange','melon‘, 20000);
```

# 레코드 삽입

```
import java.sql.*;

public class JdbcInsertStatement {

    public static void main(String[] args) {
        Connection con = null;
        Statement stmt= null;
        String driverName = "com.mysql.jdbc.Driver";
        String DBName = "jdbc_db";
        String dbURL = "jdbc:mysql://localhost:3306/" + DBName;
        String sslStr="?useSSL=false";


        try {

            Class.forName(driverName);
            con = DriverManager.getConnection(dbURL+sslStr, "root", "1111133333");
            stmt = con.createStatement();
            stmt.executeUpdate("insert into User values ('apple', 'orange', 'melon',20000);");
            stmt.executeUpdate("insert into User values ('chair', 'desk', 'key',1000);");
            stmt.executeUpdate("insert into User values ('deck', 'keyboard', 'mouse',0);");
            System.out.println("3 rows is inserted");

        }
        catch (ClassNotFoundException e) {
                System.out.println("JDBC driver load fail !!");
        } catch (SQLException e) {
            System.out.println("DB SQLException fail !!");

        }
```

# 레코드 삽입

```
        finally
        {
            if (stmt != null) {
                try{
                    stmt.close();
                }
                catch(SQLException ex) {
                    System.out.println("DB stmt close exception !!");
                }
            }
            if (con != null) {
                try {
                    con.close();
                }
                catch(SQLException ex) {
                    System.out.println("DB connection close exception !!");
                }
            }
        }
    }



}
```

# 레코드 삽입

```
import java.sql.*;

public class JdbcInsertPreparedStatement {

    public static void main(String[] args) {
        Connection con = null;
        PreparedStatement pstmt = null;
        int retval=0;
        String driverName = "com.mysql.jdbc.Driver";
        String DBName = "jdbc_db";
        String dbURL = "jdbc:mysql://localhost:3306/" + DBName;
        String sslStr="?useSSL=false";

        String sql = "insert into user values (?, ?, ?,?)";

        try {
            Class.forName(driverName);
            con = DriverManager.getConnection(dbURL+sslStr, "root", "1111133333");
            pstmt = con.prepareStatement(sql);
            pstmt.setString(1,"dog");
            pstmt.setString(2,"cat");
            pstmt.setString(3,"lion");
            pstmt.setInt(4,5000);
            retval=pstmt.executeUpdate();

            System.out.println(retval+" rows is inserted");
        }
```

# 레코드 삽입

```
        catch (ClassNotFoundException e) {
                System.out.println("JDBC driver load fail !!");
        } catch (SQLException e) {
            System.out.println("DB SQLException fail !!");

        }
        finally
        {
            if (stmt != null) {
                try{
                    stmt.close();
                }
                catch(SQLException ex) {
                    System.out.println("DB stmt close exception !!");
                }
            }
            if (con != null) {
                try {
                    con.close();
                }
                catch(SQLException ex) {
                    System.out.println("DB connection close exception !!");
                }
            }
        }
    }
}
```

# 실습(레코드 삽입)

- DB를 만들고 다음과 같이 작성하시오
  - DB명 : bookstore_db
  - Table 명 : book

| Column Name | Datatype | PK | NN |
|---|---|---|---|
| id | VARCHAR(45) | ☑ | ☑ |
| publisher | VARCHAR(45) | ☐ | ☑ |
| name | VARCHAR(45) | ☐ | ☑ |
| price | INT(11) | ☐ | ☑ |

  - 레코드 삽입
    - ‘bk2344’,’재능’,’토끼와사자’,1000
    - ‘bc5431,’우현‘, ‘세종’,1900
    - ‘ty6566’,’생생’, ‘피노키오’,2000
    - ‘ty6788’,’재능’, ‘이순신’,2500

# 데이터 검색

- 테이블의 모든 데이터 검색

```
String SQL = "select * from user";
Statement stmt = conn.createStatement();
ResultSet resultSet = stmt.executeQuery(SQL);
```

- Statement의 executeQuery()는 SQL문의 실행하여 실행 결과를 넘겨줌
- 위의 SQL문의 user 테이블에서 모든 행의 모든 열을 읽어 결과를 rs에 저장

select * from user

| id | pw | addr | cash |
|---|---|---|---|
| apple | orange | melon | 20000 |
| chair | desk | key | 1000 |
| deck | keyboard | mouse | 0 |
| dog | cat | lion | 5000 |

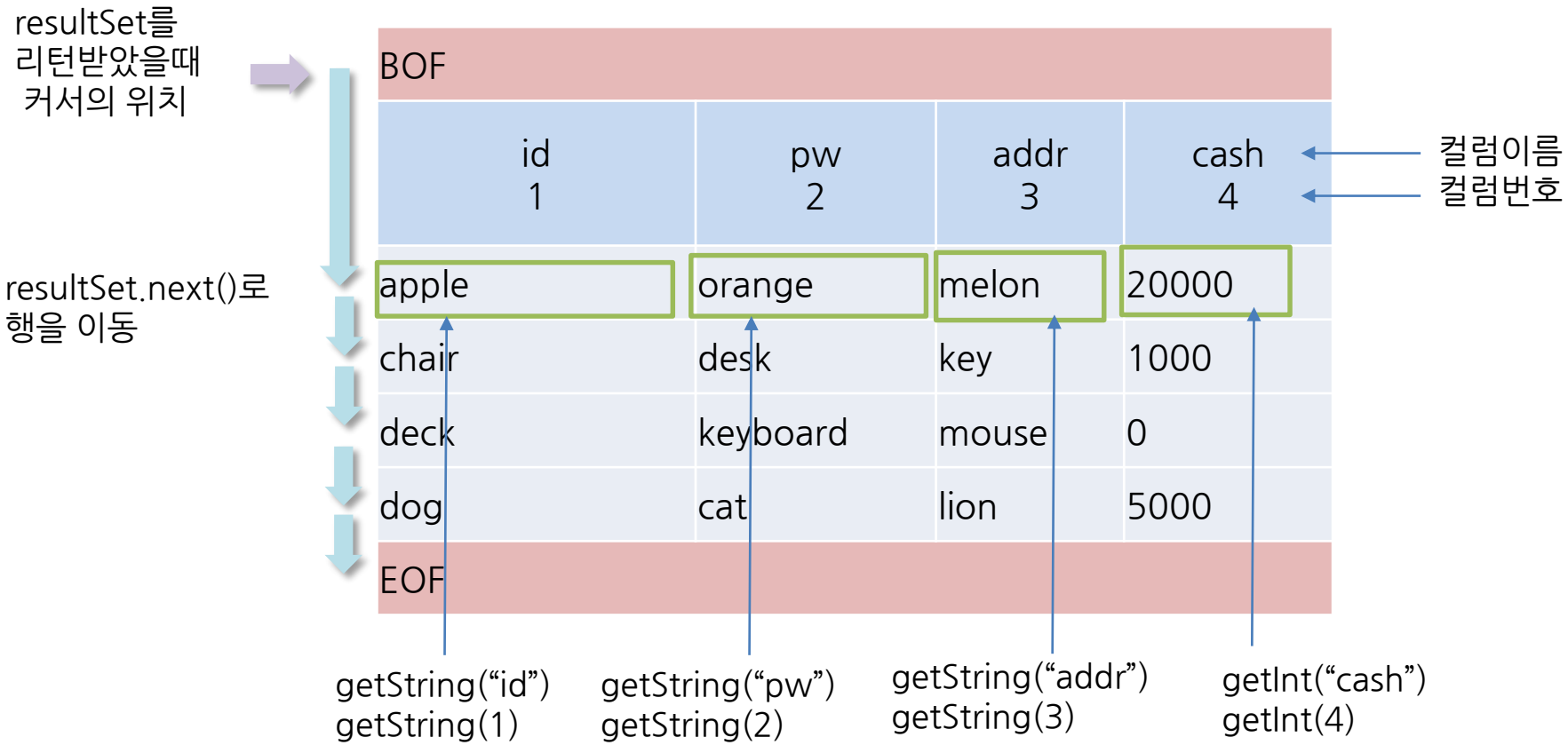
user table

| BOF | | | |
|---|---|---|---|
| id<br>1 | pw<br>2 | addr<br>3 | cash<br>4 |
| apple | orange | melon | 20000 |
| chair | desk | key | 1000 |
| deck | keyboard | mouse | 0 |
| dog | cat | lion | 5000 |
| EOF | | | |

resultSet

# 데이터 검색

- ResultSet에서 데이터를 얻는 방법
  - 행과 열을 지정
    - 행 : ResultSet 의 next()를 이용 이동 및 지정
    - 열 : getXXX()를 이용 컬럼이름이나 컬럼 번호를 지정



# 데이터 검색

- 특정 열만 검색

```
ResultSet rs = stmt.executeQuery("select id,addr from user");
```

  - 특정 열만 읽을 경우는 select문을 이용하여 특정 열의 이름 지정

select id,addr from user

| id | pw | addr | cash |
|---|---|---|---|
| apple | orange | melon | 20000 |
| chair | desk | key | 1000 |
| deck | keyboard | mouse | 0 |
| dog | cat | lion | 5000 |

user table

| BOF | |
|---|---|
| id<br>1 | addr<br>2 |
| apple | melon |
| chair | key |
| deck | mouse |
| dog | lion |
| EOF | |

resultSet

# 데이터 검색

- 조건 검색
  - select문에서 where절을 이용하여 조건에 맞는 데이터 검색

```
rs = stmt.executeQuery("select addr from user where id='apple'");
```

select addr from user where id='apple')

| id | pw | addr | cash |
|---|---|---|---|
| apple | orange | melon | 20000 |
| chair | desk | key | 1000 |
| deck | keyboard | mouse | 0 |
| dog | cat | lion | 5000 |

user table

| BOF |
|---|
| addr<br>1 |
| melon |
| EOF |

resultSet

# 데이터 검색

```
import java.sql.*;

public class JdbcSelectWhere {

    public static void main(String[] args) {
        Connection con = null;
        PreparedStatement pstmt = null;
        ResultSet resultSet=null;

        String driverName = "com.mysql.jdbc.Driver";
        String DBName = "jdbc_db";
        String dbURL = "jdbc:mysql://localhost:3306/" + DBName;
        String sslStr="?useSSL=false";

        String sql = "SELECT  addr FROM user where id=?";
        String UserId="apple";

        try {
            Class.forName(driverName);
            con = DriverManager.getConnection(dbURL+sslStr, "root", "1111133333");
            pstmt = con.prepareStatement(sql);
            pstmt.setString(1, UserId);
            resultSet = pstmt.executeQuery();

            while (resultSet.next())
            {
                System.out.println(resultSet.getString("addr"));
            }

        }
```

# 데이터 검색

```
        catch (ClassNotFoundException e) {
                System.out.println("JDBC driver load fail !!");
        } catch (SQLException e) {
            System.out.println("DB SQLException fail !!");
        }
        finally
        {
            if (resultSet != null) {
                try{
                    resultSet.close();
                }
                catch(SQLException ex) {
                    System.out.println("DB resultSet close exception !!");
                }
            }
            if (pstmt != null) {
                try{
                    pstmt.close();
                }
                catch(SQLException ex) {
                    System.out.println("DB pstmt close exception !!");
                }
            }
            if (con != null) {
                try {
                    con.close();
                }
                catch(SQLException ex) {
                    System.out.println("DB connection close exception !!");
                }
            }
        }
    }
}
```

# 집계함수

# 집계함수

```
import java.sql.*;

public class JdbcCount {

    public static void main(String[] args) {
        Connection con = null;
        PreparedStatement pstmt = null;
        ResultSet resultSet=null;

        String driverName = "com.mysql.jdbc.Driver";
        String DBName = "jdbc_db";
        String dbURL = "jdbc:mysql://localhost:3306/" + DBName;
        String sslStr="?useSSL=false";

        String sql = "select count(*) from mem";

        try {
            Class.forName(driverName);
            con = DriverManager.getConnection(dbURL+sslStr, "root", "1111133333");
            pstmt = con.prepareStatement(sql);
            resultSet = pstmt.executeQuery();

            while (resultSet.next())
            {
                System.out.println("count : "+resultSet.getInt(1));
            }

        }
        catch (ClassNotFoundException e) {
                System.out.println("JDBC driver load fail !!");
        } catch (SQLException e) {
            System.out.println("DB SQLException fail !!");
            e.printStackTrace();
        }
```

# 집계함수

```
        catch (ClassNotFoundException e) {
                System.out.println("JDBC driver load fail !!");
        } catch (SQLException e) {
            System.out.println("DB SQLException fail !!");
        }
        finally
        {
            if (resultSet != null) {
                try{
                    resultSet.close();
                }
                catch(SQLException ex) {
                    System.out.println("DB resultSet close exception !!");
                }
            }
            if (pstmt != null) {
                try{
                    pstmt.close();
                }
                catch(SQLException ex) {
                    System.out.println("DB pstmt close exception !!");
                }
            }
            if (con != null) {
                try {
                    con.close();
                }
                catch(SQLException ex) {
                    System.out.println("DB connection close exception !!");
                }
            }
        }
    }
}
```

# 집계함수

```
import java.sql.*;

public class JdbcCountAs {

    public static void main(String[] args) {
        Connection con = null;
        PreparedStatement pstmt = null;
        ResultSet resultSet=null;

        String driverName = "com.mysql.jdbc.Driver";
        String DBName = "jdbc_db";
        String dbURL = "jdbc:mysql://localhost:3306/" + DBName;
        String sslStr="?useSSL=false";

        String sql = "select count(*) as count from mem";

        try {
            Class.forName(driverName);
            con = DriverManager.getConnection(dbURL+sslStr, "root", "1111133333");
            pstmt = con.prepareStatement(sql);

            resultSet = pstmt.executeQuery();

            while (resultSet.next())
            {
                System.out.println("count : "+resultSet.getString("count"));
            }

        }
        catch (ClassNotFoundException e) {
                System.out.println("JDBC driver load fail !!");
        } catch (SQLException e) {
            System.out.println("DB SQLException fail !!");
            e.printStackTrace();
        }
```

# 집계함수

```
        finally
        {
            if (resultSet != null) {
                try{
                    resultSet.close();
                }
                catch(SQLException ex) {
                    System.out.println("DB resultSet close exception !!");
                }
            }
            if (pstmt != null) {
                try{
                    pstmt.close();
                }
                catch(SQLException ex) {
                    System.out.println("DB pstmt close exception !!");
                }
            }
            if (con != null) {
                try {
                    con.close();
                }
                catch(SQLException ex) {
                    System.out.println("DB connection close exception !!");
                }
            }
        }
    }
}
```

# 집계함수

```
import java.sql.*;

public class JdbcCountWhere {

    public static void main(String[] args) {
        Connection con = null;
        PreparedStatement pstmt = null;
        ResultSet resultSet=null;

        String driverName = "com.mysql.jdbc.Driver";
        String DBName = "jdbc_db";
        String dbURL = "jdbc:mysql://localhost:3306/" + DBName;
        String sslStr="?useSSL=false";

        String sql = "select avg(age) as avg_age from mem where age >=?;";
        int age=20;

        try {
            Class.forName(driverName);
            con = DriverManager.getConnection(dbURL+sslStr, "root", "1111133333");
            pstmt = con.prepareStatement(sql);
            pstmt.setInt(1, age);
            resultSet = pstmt.executeQuery();

            while (resultSet.next())
            {
                System.out.println("avg : "+resultSet.getString("avg_age"));
            }
        }
        catch (ClassNotFoundException e) {
                System.out.println("JDBC driver load fail !!");
        } catch (SQLException e) {
            System.out.println("DB SQLException fail !!");
        }
```

# 집계함수

```
finally
        {
            if (resultSet != null) {
                try{
                    resultSet.close();
                }
                catch(SQLException ex) {
                    System.out.println("DB resultSet close exception !!");
                }
            }
            if (pstmt != null) {
                try{
                    pstmt.close();
                }
                catch(SQLException ex) {
                    System.out.println("DB pstmt close exception !!");
                }
            }
            if (con != null) {
                try {
                    con.close();
                }
                catch(SQLException ex) {
                    System.out.println("DB connection close exception !!");
                }
            }
        }
    }
}
```

# 실습(데이터 검색)

- DB를 만들고 다음과 같이 작성하시오
  - DB명 : bookstore_db
  - Table 명 : book

| id | publisher | name | price |
|---|---|---|---|
| bk2344 | 재능 | 토끼와사자 | 1000 |
| bc5431 | 우현 | 세종 | 1900 |
| ty6566 | 생생 | 피노키오 | 2000 |
| ty6788 | 재능 | 이순신 | 2500 |

1. 테이블의 모든 내용을 출력하세요
2. 테이블의 publisher과 name을 모두 출력하세요
3. publisher이 '우현'인 레코드의 name, price를 출력하세요

# 데이터 수정

```
import java.sql.*;

public class JdbcUpdate {

    public static void main(String[] args) {
        Connection con = null;
        PreparedStatement pstmt = null;

        int retval=0;

        String driverName = "com.mysql.jdbc.Driver";
        String DBName = "jdbc_db";
        String dbURL = "jdbc:mysql://localhost:3306/" + DBName;
        String sslStr="?useSSL=false";

        String sql = "UPDATE user SET pw=? where id=?";
        String UserId="apple";
        String UserPw="starfruit";

        try {
            Class.forName(driverName);
            con = DriverManager.getConnection(dbURL+sslStr, "root", "1111133333");
            pstmt = con.prepareStatement(sql);

            pstmt.setString(1, UserPw);
            pstmt.setString(2, UserId);

            retval = pstmt.executeUpdate();
            System.out.println(retval+" is updated");

        }
```

# 데이터 수정

```
}
        catch (ClassNotFoundException e) {
                System.out.println("JDBC driver load fail !!");
        } catch (SQLException e) {
            System.out.println("DB SQLException fail !!");
            e.printStackTrace();

        }
        finally
        {

            if (pstmt != null) {
                try{
                    pstmt.close();
                }
                catch(SQLException ex) {
                    System.out.println("DB pstmt close exception !!");
                }
            }
            if (con != null) {
                try {
                    con.close();
                }
                catch(SQLException ex) {
                    System.out.println("DB connection close exception !!");
                }
            }
        }
    }
}
```

# 실습(데이터 검색)

- DB를 만들고 다음과 같이 작성하시오
  - DB명 : bookstore_db
  - Table 명 : book

| id | publisher | name | price |
|---|---|---|---|
| bk2344 | 재능 | 토끼와사자 | 1000 |
| bc5431 | 우현 | 세종 | 1900 |
| ty6566 | 생생 | 피노키오 | 2000 |
| ty6788 | 재능 | 이순신 | 2500 |

1. name가 '세종'인 레코드의 price를 900으로 변경하세요
2. 테이블의 모든 데이터를 출력하세요

# 데이터 삭제

```
import java.sql.*;

public class JdbcSelectWhere {

    public static void main(String[] args) {
        Connection con = null;
        PreparedStatement pstmt = null;
        ResultSet resultSet=null;

        String driverName = "com.mysql.jdbc.Driver";
        String DBName = "jdbc_db";
        String dbURL = "jdbc:mysql://localhost:3306/" + DBName;
        String sslStr="?useSSL=false";

        String sql = "SELECT  addr FROM user where id=?";
        String UserId="apple";

        try {
            Class.forName(driverName);
            con = DriverManager.getConnection(dbURL+sslStr, "root", "1111133333");
            pstmt = con.prepareStatement(sql);
            pstmt.setString(1, UserId);
            resultSet = pstmt.executeQuery();

            while (resultSet.next())
            {
                System.out.println(resultSet.getString("addr"));
            }

        }
```

# 데이터 삭제

```
        catch (ClassNotFoundException e) {
                System.out.println("JDBC driver load fail !!");
        } catch (SQLException e) {
            System.out.println("DB SQLException fail !!");
            e.printStackTrace();

        }
        finally
        {

            if (pstmt != null) {
                try{
                    pstmt.close();
                }
                catch(SQLException ex) {
                    System.out.println("DB pstmt close exception !!");
                }
            }
            if (con != null) {
                try {
                    con.close();
                }
                catch(SQLException ex) {
                    System.out.println("DB connection close exception !!");
                }
            }
        }
    }
}
```

# 실습(데이터 삭제)

- DB를 만들고 다음과 같이 작성하시오
  - DB명 : bookstore_db
  - Table 명 : book

| id | publisher | name | price |
|---|---|---|---|
| bk2344 | 재능 | 토끼와사자 | 1000 |
| bc5431 | 우현 | 세종 | 1900 |
| ty6566 | 생생 | 피노키오 | 2000 |
| ty6788 | 재능 | 이순신 | 2500 |

1. publishe이 '재능'인 레코드를 삭제하세요
2. 테이블의 모든 데이터를 출력하세요

# 날짜 함수

```
import java.sql.*;

public class JdbcInsertDateTime {

    public static void main(String[] args) {
        Connection con = null;
        PreparedStatement pstmt = null;
        int retval=0;
        String driverName = "com.mysql.jdbc.Driver";
        String DBName = "jdbc_db";
        String dbURL = "jdbc:mysql://localhost:3306/" + DBName;
        String sslStr="?useSSL=false";

        String sql = "insert into mem_time values(null,now(),curdate(),curtime())";

        try {
            Class.forName(driverName);
            con = DriverManager.getConnection(dbURL+sslStr, "root", "1111133333");
            pstmt = con.prepareStatement(sql);
            retval=pstmt.executeUpdate();
            System.out.println(retval+" rows is inserted");
        }
        catch (ClassNotFoundException e) {
                System.out.println("JDBC driver load fail !!");
        } catch (SQLException e) {
            System.out.println("DB SQLException fail !!");
        }
```

# 날짜 함수

```
finally
        {
            if (pstmt != null) {
                try{
                    pstmt.close();
                }
                catch(SQLException ex) {
                    System.out.println("DB stmt close exception !!");
                }
            }
            if (con != null) {
                try {
                    con.close();
                }
                catch(SQLException ex) {
                    System.out.println("DB connection close exception !!");
                }
            }
        }
    }
}
```

# 날짜 함수

```
import java.sql.*;

public class JdbcSelectDateTime {

    public static void main(String[] args) {
        Connection con = null;
        PreparedStatement pstmt = null;
        ResultSet resultSet=null;

        String driverName = "com.mysql.jdbc.Driver";
        String DBName = "jdbc_db";
        String dbURL = "jdbc:mysql://localhost:3306/" + DBName;
        String sslStr="?useSSL=false";

        String sql = "SELECT  id, addr FROM user";

        try {
            Class.forName(driverName);
            con = DriverManager.getConnection(dbURL+sslStr, "root", "1111133333");
            pstmt = con.prepareStatement(sql);
            resultSet = pstmt.executeQuery();

            while (resultSet.next())
            {
                System.out.print(resultSet.getString("id")+"₩t");
                System.out.println(resultSet.getString("addr")+"₩t");
            }
        }
        catch (ClassNotFoundException e) {
                System.out.println("JDBC driver load fail !!");
        } catch (SQLException e) {
            System.out.println("DB SQLException fail !!");
        }
```

# 날짜 함수

```
finally
        {
            if (resultSet != null) {
                try{
                    resultSet.close();
                }
                catch(SQLException ex) {
                    System.out.println("DB resultSet close exception !!");
                }
            }
            if (pstmt != null) {
                try{
                    pstmt.close();
                }
                catch(SQLException ex) {
                    System.out.println("DB pstmt close exception !!");
                }
            }
            if (con != null) {
                try {
                    con.close();
                }
                catch(SQLException ex) {
                    System.out.println("DB connection close exception !!");
                }
            }
        }
    }
}
```