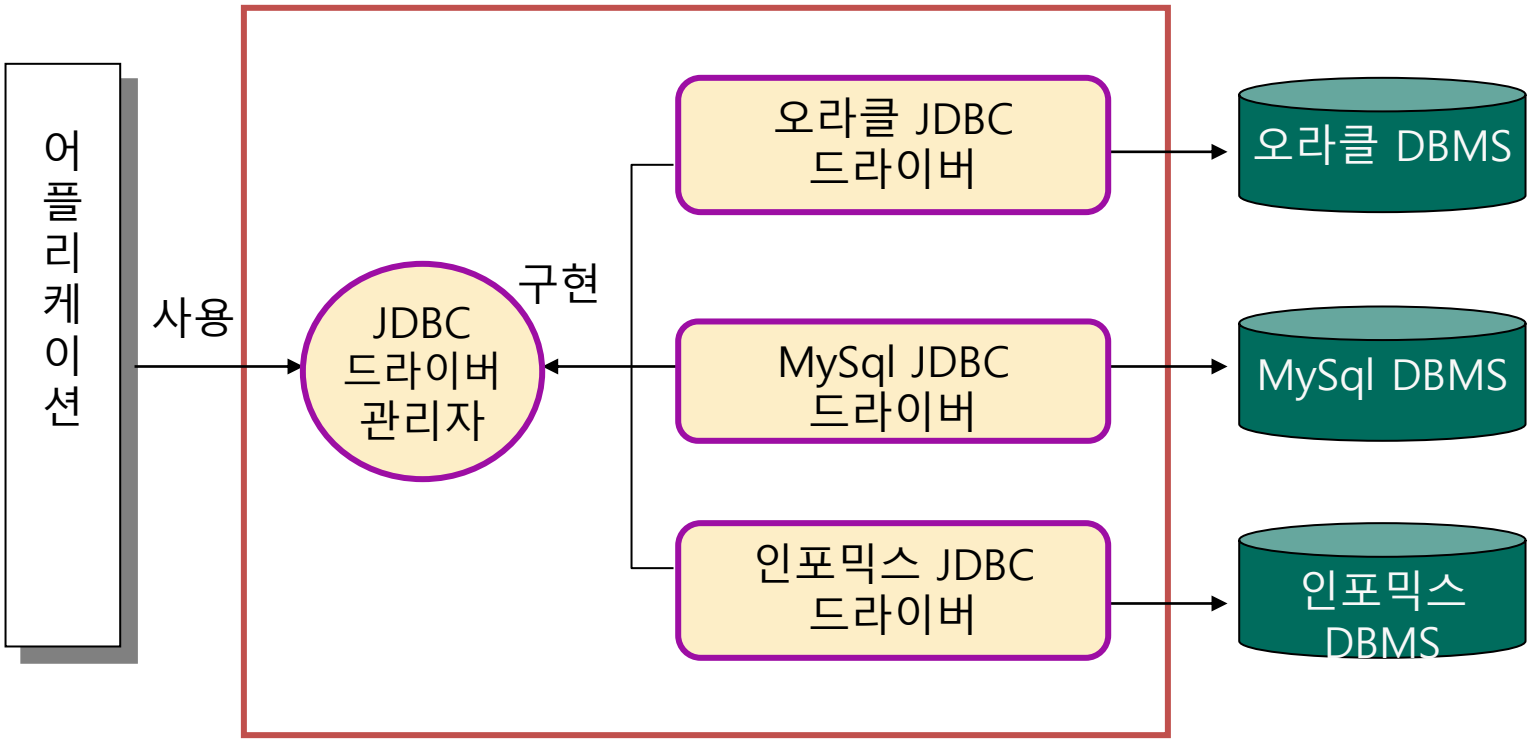
JAVA
- JDBC

■ JDBC 개념 및 역할

- 자바애플리케이션에서 표준화된 데이터베이스 접근 제공.
- 각 데이터베이스 접속에 대한 상세한 정보를 알 필요 없음.
- 이론적으로는 개발된 애플리케이션에서 DB 변경 시
  JDBC 드라이버만 교체하면 됨

JDBC 구조

- JDBC(Java Database Connectivity)
- 자바에서 데이터베이스를 표준화 된 방법으로 접속할 수 있도록 만든 API 규격
- 데이터베이스 벤더와 상관없이 동일한 개발이 가능함



■ JDBC 드라이버 설치

- **JDBC 드라이버 선택**

  JDBC 드라이버는 사용하고자 하는 데이터베이스 벤더 별로 제공 됨

- **오라클 JDBC드라이버**

  C:₩app₩product₩11.2.0₩dbhome_1₩jdbc₩lib₩ojdbc6.jar

- **설치 디렉터리(다음 중 한 가지를 이용함)**

  1. 프로젝트 라이브러리에 추가하는 방법.
  2. **이클립스 프로젝트의 WebContent₩WEB-INF₩lib 폴더에 복사하는 방법**
  3. 톰캣설치 디렉터리₩common₩lib 폴더에 복사하는 방법

WebContent₩WEB-INF₩lib 폴더에 에 설치

■ JDBC 드라이버 설치

MySQL JDBC 드라이버의 다운로드 https://dev.mysql.com/downloads/

MySQL Connectors

MySQL offers standard database driver connectivity for using MySQL with applications and tools that are compatible with industry standards ODBC and JDBC.

DOWNLOAD

Connector/J

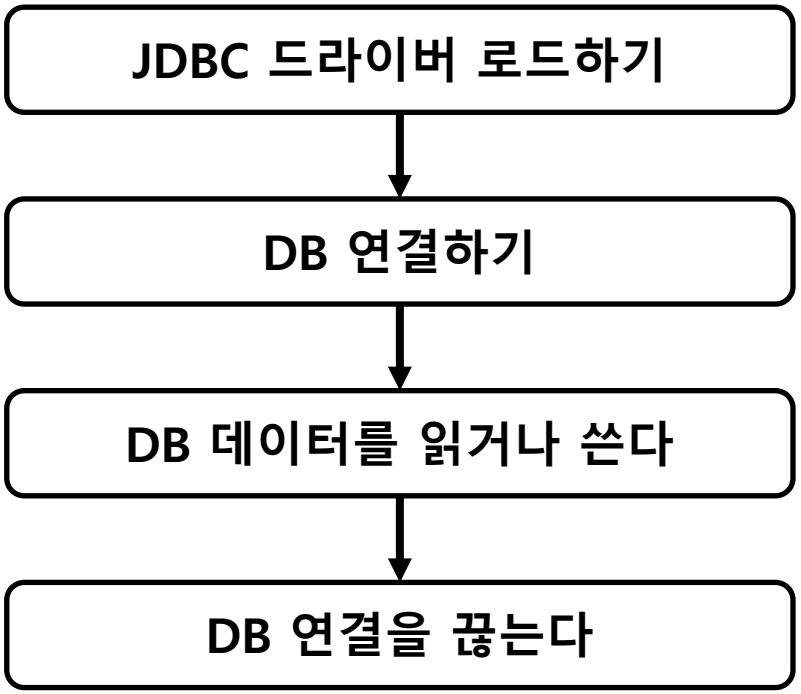
Standardized database driver for Java platforms and development.

C:₩Program Files (x86)₩MySQL₩Connector J 8.0

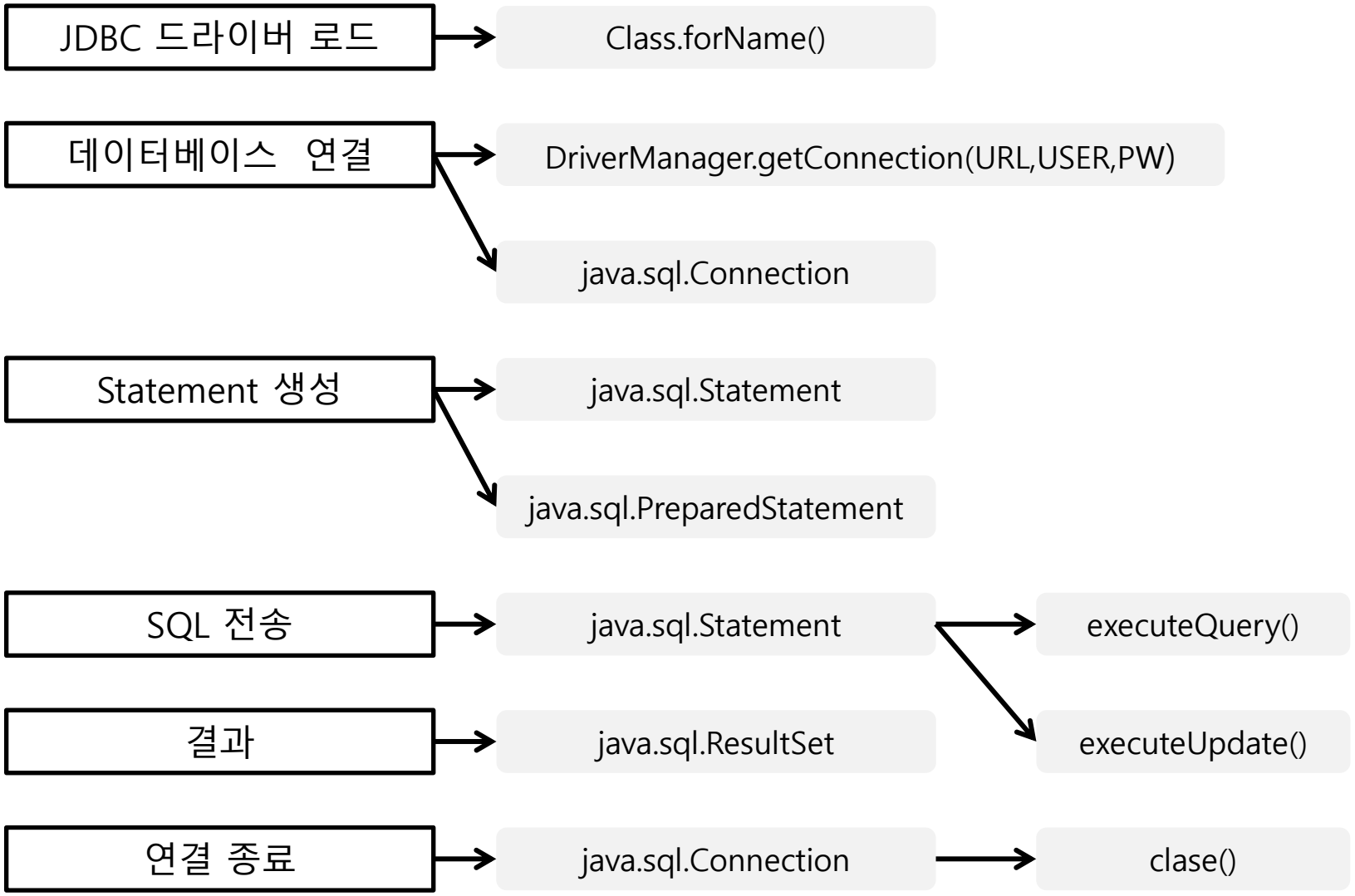
JDBC를 이용한 프로그래밍

■ JDBC를 이용한 프로그램의 작성

자바 프로그램에서 데이터베이스를 사용하는 4단계의 과정



■ JDBC를 이용한 프로그램의 작성



■ JDBC 드라이버의 로드

JDBC 드라이버를 로드하는 방법

```
Class.forName("oracle.jdbc.driver.OracleDriver");
```

oracle 데이터베이스 드라이버 로드

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

Mysql 데이터베이스 드라이버 로드

■ 데이터베이스 연결

데이터베이스 연결 URL의 예

JDBC:oracle:thin:@IP주소:1521:orcl

프로토콜 서브프로토콜 서브네임

jdbc:mysql://IP주소:3306:sid

■ 데이터베이스 연결

데이터베이스로 연결하는 방법

```
Connection conn = DriverManger.getConnection
                                (JDBC_url,"아이디","비밀번호");
```

■ 데이터베이스 연결 끊기

데이터베이스와 연결을 끊는 방법

```
conn.colse();
```

데이터베이스 연결 종료

■ JDBC 드라이버 로드, 데이터베이스에 연결

```
import java.sql.*;
class JDBCExample1 {
    public static void main(String args[]) {
        Connection conn = null;
        try {
            Class.forName("com.mysql.jdbc.Driver");
            conn = DriverManager.getConnection(
               "jdbc:mysql://localhost:3306/malldb", "root", "");
            System.out.println("데이터베이스에 접속했습니다.");
            conn.close();
        }
        catch (ClassNotFoundException cnfe) {
            System.out.println("해당 클래스를 찾을 수 없습니다." + cnfe.getMessage());
        }
        catch (SQLException se) {
            System.out.println(se.getMessage());
        }
    }
}
```

■ JDBC 드라이버 로드, 데이터베이스에 연결

```
import java.sql.*;
class JDBCExample1 {
    public static void main(String args[]) {
        Connection conn = null;
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            conn = DriverManager.getConnection(
               "jdbc:oracle:thin:@localhost:1521:orcl" , "scott", "tiger");
            System.out.println("데이터베이스에 접속했습니다.");
            conn.close();
        }
        catch (ClassNotFoundException cnfe) {
            System.out.println("해당 클래스를 찾을 수 없습니다." + cnfe.getMessage());
        }
        catch (SQLException se) {
            System.out.println(se.getMessage());
        }
    }
}
```

■ 데이터 조회

테이블의 데이터를 읽어오는 방법

- Statement 객체를 얻는다.

```
Statement stmt = conn.createStatement();
```

■ 데이터 조회

Statement 객체 생성 후 SQL 문장 처리 요청

Statement 객체 생성 후 SQL 문장을 변수 처리부와 함께 문자열로 구현

쿼리가 복잡해질수록 성능저하 및 관리에 어려움이 있음

Statement stmt = conn.createStatement();

stmt.executeUpdate("insert into test values (' " +username+"','"+email+"')");

■ 데이터 조회

PreparedStatement 객체 생성시 SQL 문장을 미리 생성하고 변수 부는 별도의 메서드로 대입하는 방식으로 성능과 관리 면에서 모두 권장 되는 방식임

```
String sql = “insert into test values(?,?)”;
PreparedStatement pstmt = conn.prepareStatement(sql);
pstmt.setString(1, “SCOTT”);
pstmt..setString(2, “scott@test.com”);
pstmt.executeUpdate();
```

■ 데이터 조회

테이블의 데이터를 읽어오는 방법

- executeQuery 메소드를 호출합니다.

```
String sql1 = "select * from dept order by DEPTNO";
ResultSet rs = stmt.executeQuery(sql1);
```

■ 데이터 조회

테이블의 데이터를 읽어오는 방법

– next 메소드를 호출합니다.

```
rs.next()
```

■ 데이터 조회

테이블의 데이터를 읽어오는 방법

– getInt, getString, getFloat 메소드를 호출하여 특정 컬럼의 값을 가져옵니다.

```
String ename = rs.getString("ename")
```

질의 결과 데이터에서 지정 컬럼의 값을 String 타입의 값으로 반환해온다

지정 컬럼 이름

■ 데이터 조회

테이블의 데이터를 읽어오는 방법

```
ResultSet rs = pstmt.executeQuery();

while(rs.next()) {
    name = rs.getString("name");
    age = rs.getInt("email");
}
```

더 이상 읽어올 행이 없을 때 까지 반복

```
//-------------------------------------------------------------

while(rs.next()) {
    name = rs.getString(1);     // or rs.getString("name");
    age = rs.getInt(2);         // or rs.getInt("email");
}


rs.close();
```

■ 데이터 조회

테이블의 데이터를 읽어오는 방법

– ResultSet 객체에 대해 close 메소드를 호출합니다.

```
rs.close();
```

■ 데이터 조회

테이블의 데이터를 읽어오는 방법

– Statement 객체에 대해 close 메소드를 호출합니다.

```
pstmt.close();
```

■ 데이터 입력/수정/삭제

테이블의 데이터를 입력/수정/삭제하는 방법

– Statement 객체를 얻습니다.

```
Statement stmt = conn.createStatement();
```

Statement 객체를 만드는 메소드

■ 데이터 입력/수정/삭제

테이블의 데이터를 입력/수정/삭제하는 방법

– executeUpdate 메소드를 호출합니다.

```
String sql = "insert into member values(1, 'scott', '1111', 'SCOTT', null)";
int rowNum = stmt.excuteUpdate(sql);
```

Sql 문을 실행 하고, 실행 횟수를 반환한다.

데이터베이스연결 예제

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class JDBCExample1 {
        public static void main(String[] args) {
                Connection conn = null;
                  String url = "jdbc:oracle:thin:@localhost:1521:orcl";
                  try {
                   Class.forName("oracle.jdbc.driver.OracleDriver");
                   conn = DriverManager.getConnection(url, "scott", "tiger");
                   System.out.println("데이터베이스에 접속했습니다.");
                   conn.close();
                  } catch (ClassNotFoundException cnfe) {
                   System.out.println("해당 클래스를 찾을 수 없습니다." +
cnfe.getMessage());
                  } catch (SQLException se) {
                   System.out.println(se.getMessage());
                  }
        }

}
```

데이터베이스 처리 예제

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class JDBCExample2 {
        public static void main(String args[]) {
                Connection conn = null;
                String url = "jdbc:oracle:thin:@localhost:1521:orcl";
                try {
                        Class.forName("oracle.jdbc.driver.OracleDriver");
                        conn = DriverManager.getConnection(url, "scott", "tiger");
                        System.out.println("데이터베이스에접속했습니다.");
                       System.out.println("======================================");
```

데이터베이스연결 예제

```
String sql1 = "INSERT INTO dept (DEPTNO, DNAME, Loc)  VALUES (?, ?, ?) ";
```

```
Statement stmt = conn.createStatement();
PreparedStatement pstmt = conn.prepareStatement(sql1);
pstmt.setInt(1, 90);
pstmt.setString(2, "MARKETING");
pstmt.setString(3, "JEJU");
int procNum = pstmt.executeUpdate();
System.out.println(procNum + "개 행이 적용되었습니다.");
```

데이터베이스연결 예제

```
                String sql2 = "select * from dept order by DEPTNO";

                ResultSet rs = stmt.executeQuery(sql2);

                        while (rs.next()) {
                        System.out.print("부서번호 : " + rs.getInt(1) + "\t");
                        System.out.print("부서번호 : " + rs.getString(2) + "\t");
                        System.out.print("부서번호 : " + rs.getString(3) + "\t");
                       System.out.println("\n---------------------------------");
                        }
                        rs.close();
                        pstmt.close();
                        conn.close();
                } catch (ClassNotFoundException cnfe) {
                System.out.println("해당클래스를 찾을 수 없습니다." + cnfe.getMessage());
                } catch (SQLException se) {
                        System.out.println(se.getMessage());
                }
        }
}
```

[문제]

1. EMP 테이블에 새로운 사원 정보를 입력하는 프로그램을 작성해보자.

2. EMP 테이블의 모든 데이터를 출력하는 프로그램을 작성해보자.

3. EMP 테이블에 서 “SCOTT” 사원의 급여(sal) 정보를 1000으로 바꾸는 프로그램을 작성해보자.

4. EMP 테이블에 서 “SCOTT” 이름으로 검색한 결과를 출력하는 프로그램을 작성해보자.

5. 모든 사원정보를 출력하되 부서정보를 함께 출력하는 프로그램을 작성해보자.