Oracle Database

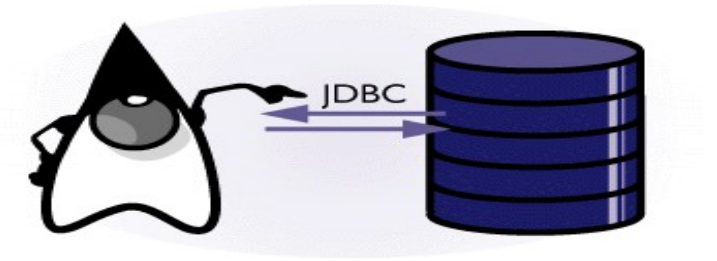
JDBC Programming

JDBC Programming

JDBC 기본 API

JDBC 개요
: 정의

- JDBC (Java Database Connectivity)
  - 자바를 이용한 데이터베이스 접속과 SQL 문장의 실행, 그리고 실행 결과로 얻어진 데이터의 핸들링을 제공하는 방법과 절차에 관한 규약
  - 자바 프로그램 내에서 SQL 문장을 실행하기 위한 자바 API
  - SQL과 프로그래밍 언어의 통합 접근 중 한 형태
- 개발자를 위한 표준 인터페이스인 JDBC API와 데이터베이스 벤더, 또는 기타 서드파티에서 제공하는 드라이브(driver)



JDBC 개요
: 환경 구성

- JDK 설치
  - http://java.sun.com

- JDBC 드라이버 설치
  - 오라클 JDBC 드라이버를 다운로드 받는다 (ojdbc6.jar)
    - http://www.oracle.com/technetwork/apps-tech/jdbc-112010-090769.html
    - 다운로드 받은 JDBC 드라이버를 %JAVA_HOME%\jre\lib\ext에 복사
    - 오라클을 설치한 경우의 위치:
      C:\oraclexe\app\oracle\product\11.2.0\server\jdbc\lib
  - 또는 CLASSPATH에 경로 추가
  - 이클립스의 경우
    - 프로젝트 우클릭 - Properties - Java Build Path - Libraries(탭) - Add External JARs

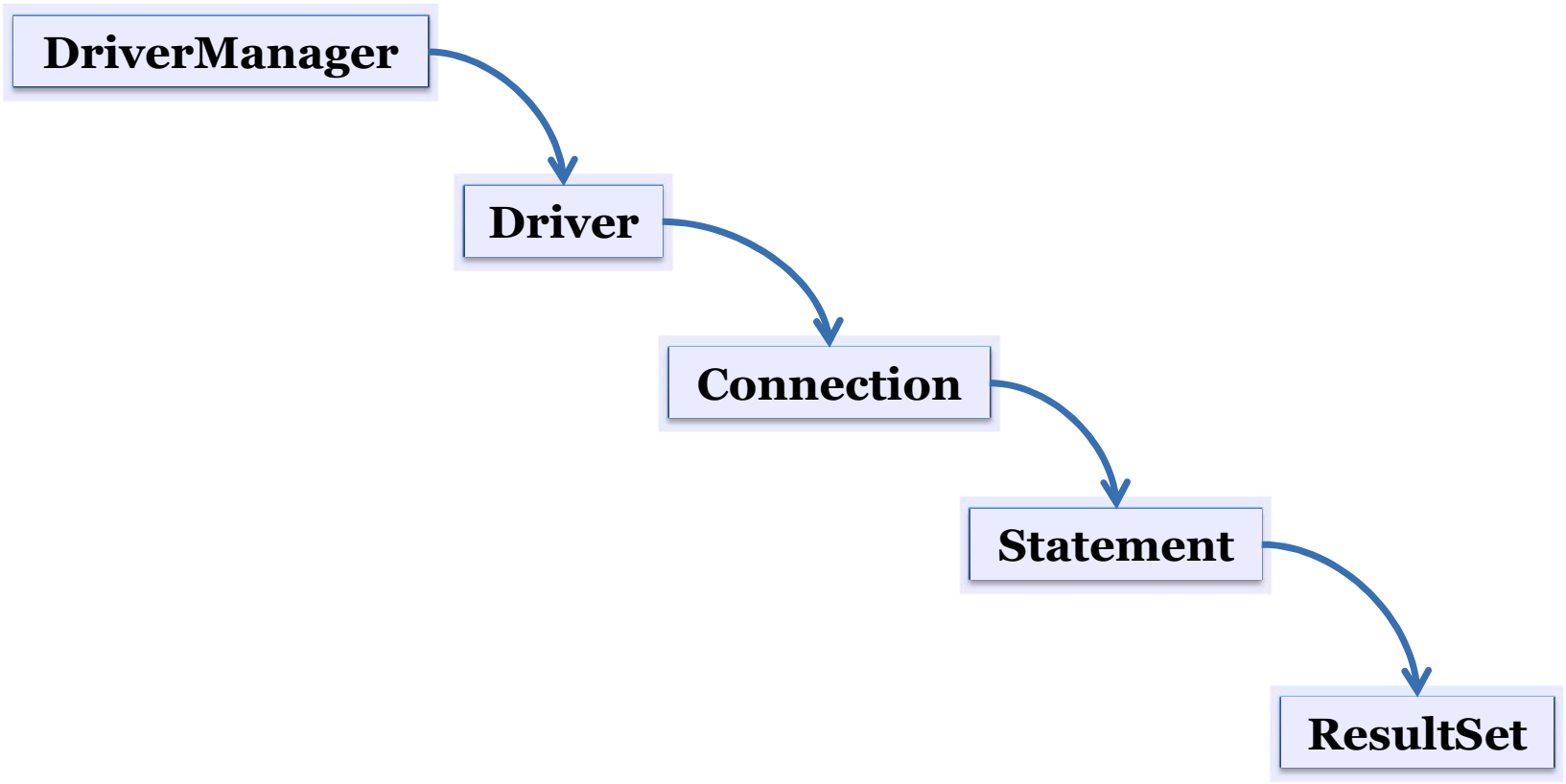
JDBC 개요
: 참고

- Java API Reference
  - https://docs.oracle.com/javase/8/docs/api/
- JDBC Tutorial
  - https://docs.oracle.com/javase/tutorial/jdbc/index.html

JDBC 사용
: 단계별 정리

- JDBC를 이용한 데이터베이스 연결 방법
  - 1단계 : import java.sql.*;
  - 2단계 : 드라이버 로드
  - 3단계 : Connection 객체 생성
  - 4단계 : Statement 객체 생성 및 질의 수행
  - 5단계 : SQL 결과물이 있다면 ResultSet 객체를 생성
  - 6단계 : 모든 객체를 닫는다

JDBC 사용
: 사용 Class



JDBC 사용
: 드라이버 로드와 Connection 얻기

- IMPORT

```
import java.sql.*;
```

- 드라이버 로드

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

- Connection 얻기

```
String dburl = "jdbc:oracle:thin:@localhost:1521:xe";
Connection conn = DriverManager.getConnection(dburl, ID, PWD);
```

JDBC 사용
: SQL 실행 및 결과 얻기

- Statement 생성

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

- 질의 수행 및 결과 얻기

```
ResultSet rs = stmt.executeQuery("SELECT no FROM user");
```

- 질의 수행을 위한 stmt의 메서드
  - stmt.execute(QUERY_STRING);
  - stmt.executeQuery(QUERY_STRING);
  - stmt.executeUpdate(QUERY_STRING);

JDBC 사용
: 질의 수행 메서드 예제

- executeQuery(String sql) - SELECT

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

- executeUpdate(String sql) - INSERT, UPDATE, DELETE

```
Statement stmt = con.createStatement();
int updateCount = stmt.executeUpdate("INSERT INTO test VALUES (1)");
```

- execute(String sql) - 기타 SQL 문

```
Statement stmt = con.createStatement();
if (stmt.execute(sql)) {
    ResultSet rs = stmt.getResultSet();
    ...
} else {
    int rowCount = stmt.getUpdateCount();
    ...
}
```

JDBC 사용
: 결과 활용 및 객체 닫기

- ResultSet으로 결과 받기

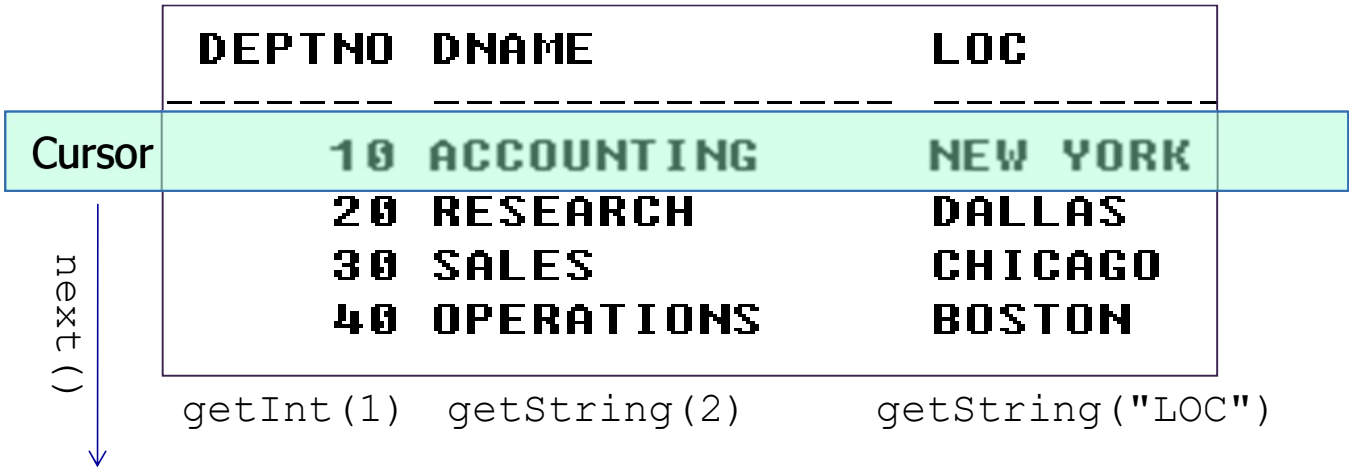
```
ResultSet rs = stmt.executeQuery( "select no from user" );
while ( rs.next() )
    System.out.println( rs.getInt( "no") );
```

- Close

```
rs.close();
stmt.close();
conn.close();
```

JDBC 사용
: ResultSet

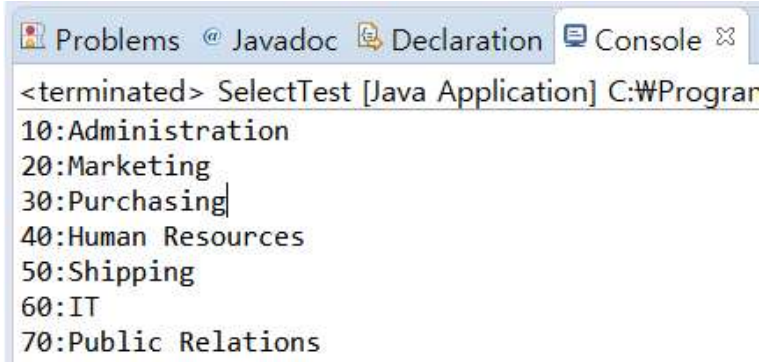
- SQL 문에 대한 결과를 얻는다
  - 이동 함수 : next, previous, first, last, beforeFirst, afterLast, relative, absolute
    - 기본 : TYPE_FORWARD_ONLY & CURSOR_READ_ONLY
      (= next로만 이동 가능 & Read Only)
  - 값 추출 함수 : getXXX(숫자/이름) - 데이터 타입에 따라 알맞게



JDBC 사용

- [실습] ConnectionTest.java
  - hr/hr 계정에 JDBC를 이용하여 Connection을 연결
  - 결과 메시지
    - 접속에 성공했을 때 : "연결 성공!!"
    - JDBC 드라이버를 로드하지 못했을 때 : "JDBC Driver를 찾지 못했습니다."
    - 그외 SQL 에러가 발생했을 경우 : "SQLError!!"

- [실습] SelectTest.java
  - hr/hr 계정에 JDBC를 이용하여 Connection을 연결, departments 테이블로부터 데이터를 불러와서 {department_id}:{department_name} 형태로 출력
  - 결과 메시지
    - {department_id}:{department_name} 의 리스트 형태로 출력
    - JDBC 드라이버를 로드하지 못했을 때 : "JDBC Driver를 찾지 못했습니다."
    - 그외 SQL 에러가 발생했을 경우 : "SQLError!!"



JDBC 사용
: 주의

- 에러시 해결 방법
  - 에러 메시지를 확인
  - 대소문자가 틀렸나 확인 (클래스명, 파일명 등)
  - JDBC는 제대로 로드 되었나?
  - CLASSPATH나 PATH는 잘 설정되어 있는가?
  - DBMS는 제대로 실행되고 있는가?
  - DBMS로의 접근이 가능한가?
  - DB Name이나 USERNAME, Password는 올바른가?

JDBC 사용
: 실습 문제

- [실습 1] hr/hr 계정의 사원 이름과 매니저 이름을 함께 출력해 봅시다
  - 사원 이름은 이름 성 순으로 표기합니다
  - 정렬은 사원 이름 내림차순입니다
  - HREmpList 프로젝트를 만들고 HREmpList 클래스에서 실행되어야 합니다

- [실습 2] 사원 검색 프로그램
  : 클래스 Scanner를 사용하여 사원 이름을 입력 받아 사원 정보를 검색하는 프로그램을 작성해 봅시다
  - 부분 이름 검색이 가능해야 합니다
  - 성, 이름 컬럼에 대해 OR 검색이 되어야 합니다
  - 출력 사원 정보는 이름 성, Email, 전화번호, 입사일입니다
  - HRSearchEmployees 프로젝트를 만들고 HRSearchEmployee 클래스에서 실행되어야 합니다

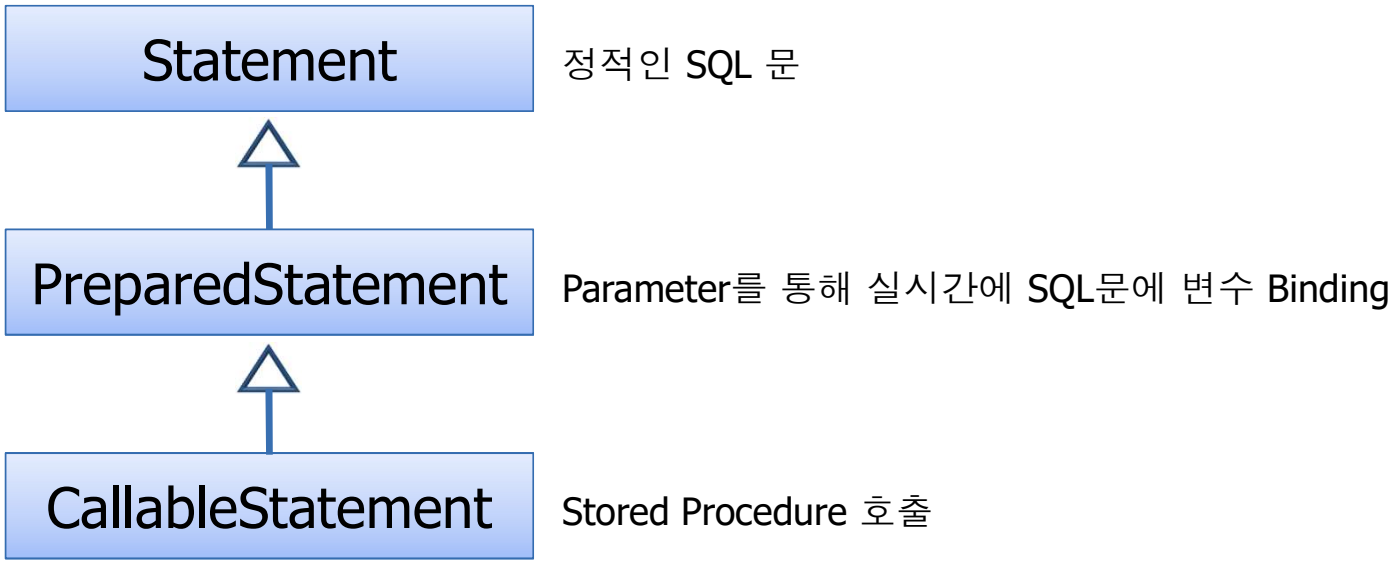
JDBC 사용
: 실습 문제

- [실습 3] 급여 검색 프로그램 작성
  : 사용자로부터 최소 급여와 최대 급여를 입력 받아 급여가 이 범위 내에 속하는 사원의 정보를 출력하는 프로그램을 작성해 봅시다
  - 정렬은 salary 오름차순입니다
  - HRSalary 프로젝트의 HRSalary 클래스에서 실행되어야 합니다
  - 결과 예시

```
2000 10000
============================================================
Kevin Freeney                3000
Donald Oconnell              4000
Pat Fay                      90000
```

Statement 활용
: 상속 관계

- Statement를 상속받아 각 클래스가 정의되어 있음
  - 자식 클래스는 부모 클래스의 기능 + alpha



Statement 활용
: PreparedStatement

- 수행방법
  - SQL 미리 준비 : 파싱, 실행 계획 저장
  - 실시간에 Parameter 바인딩

- 바인딩
  - 바인딩 변수 : ?
  - setXXX 메서드 : 숫자 1부터

- 장점
  - 효율성
  - 보안 (SQL Injection 회피)

Statement 활용
: PreparedStatement

```
...
String sql = "SELECT no FROM user WHERE no = ?";
PreparedStatement pstmt = conn.preparedStatement(sql);
pstmt.setInt(1, 10);
ResultSet rs = pstmt.executeQuery();
...
```

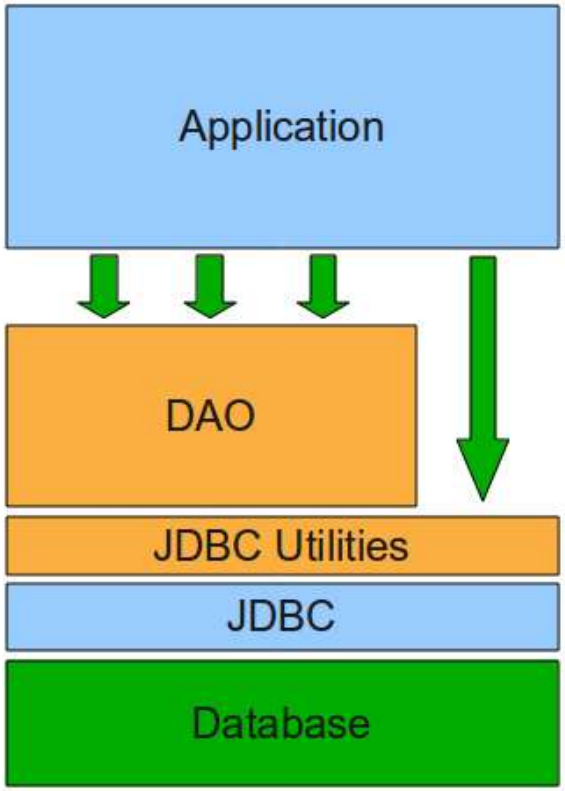
- [실습 4] 실습문제 2, 3을 PreparedStatement를 사용해 다시 작성해 봅시다

JDBC Programming

DAO (Data Access Object)

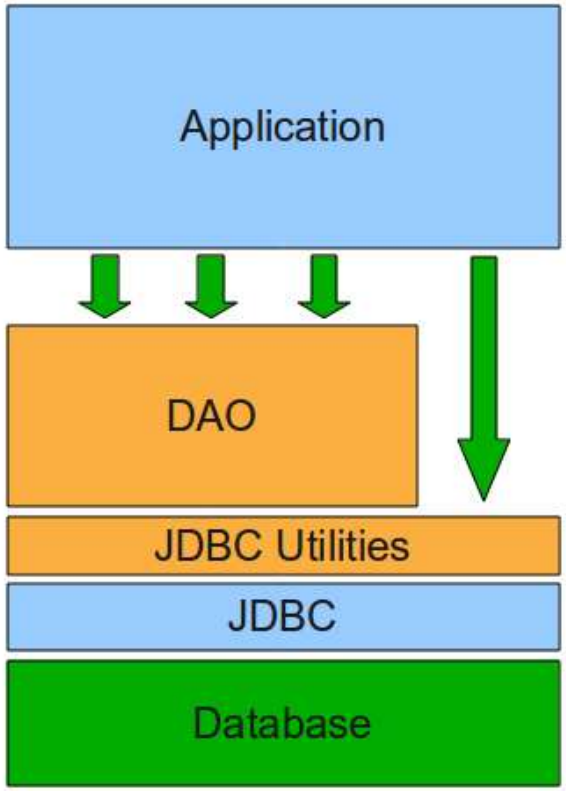
DAO (Data Access Object)
: 개념

- DAO (Data Access Object)
  - DB를 사용하여 데이터를 조회하거나 조작하는 기능을 전담하도록 만든 오브젝트(트랜잭션 객체)
  - 주요 로직에서 Database를 접속, 질의를 수행하고 결과를 반환하는 로직을 DAO에 위임하도록 구현
  - Domain Logic으로부터 Persistence Machanism을 숨기기 위해 사용
    - 적절히 디자인을 하면 모든 Domain Logic을 바꾸는 대신 DAO를 바꾸기만 하면 된다
    - 사용자는 자신이 필요한 Interface를 DAO에 던지고 DAO는 이 인터페이스를 구현한 객체를 사용자에게 반환



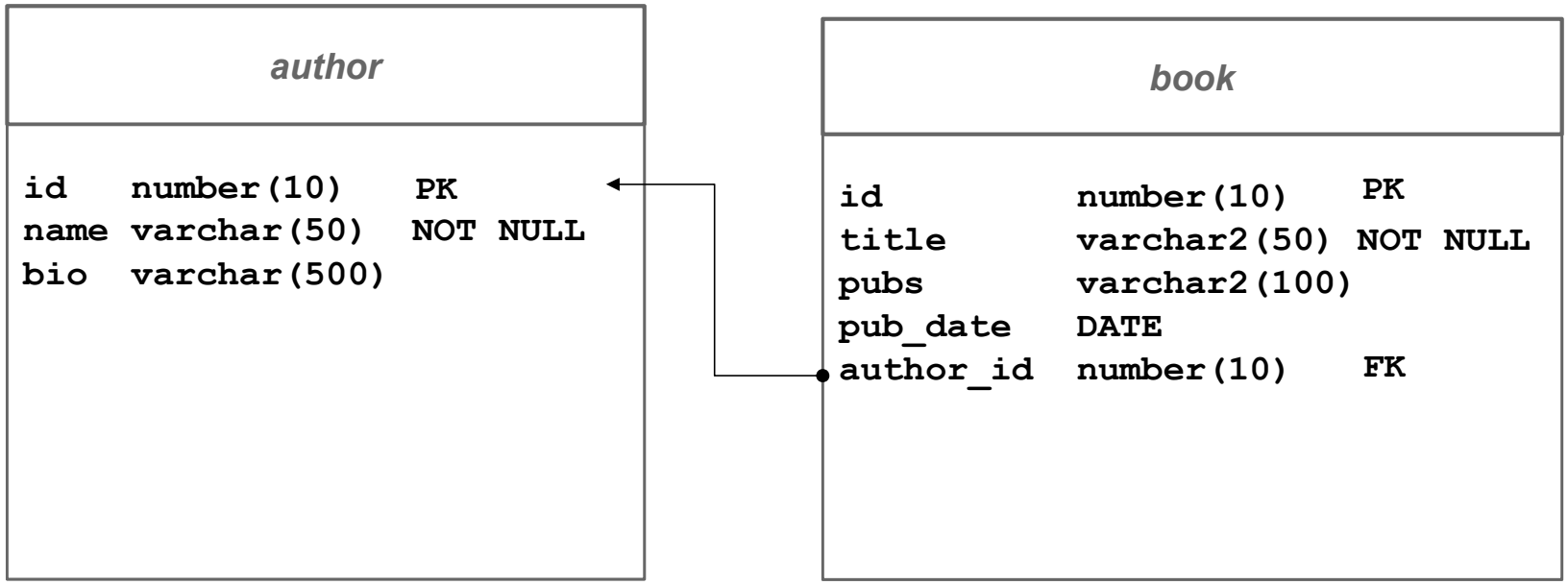
DAO (Data Access Object)
: 개념

- DTO(Data Transfer Object)
  - 계층간 데이터 교환을 위한 자바빈즈
  - 컨트롤러, 뷰, 비즈니스 계층, 퍼시스턴스 계층간 데이터 교환
  - VO(Value Object) 패턴이라고도 함
  - 일반적으로 DTO는 로직을 갖고 있지 않은 순수 데이터 객체이며 속성과 그 속성에 접근하기 위한 getter, setter 메서드만 가진다.
  - 추가적으로 toString(), equals() 등의 Object 클래스 메서드를 작성하기도 한다



DAO
: 예제

- 다음의 book, author 테이블에서 DAO 패턴을 연습하는 예제입니다
  - Scheme



DAO
: 예제

- 실습을 위해 다음 DDL을 검토하고 데이터베이스에 반영합니다

```
DROP TABLE book CASCADE CONSTRAINT;
DROP TABLE author CASCADE CONSTRAINT;
DROP SEQUENCE seq_author;
DROP SEQUENCE seq_book;
```

```
CREATE TABLE author (
  id         NUMBER(10),
  name       VARCHAR2(50) NOT NULL,
  bio        VARCHAR2(500),
  PRIMARY KEY(id)
);
```

```
CREATE TABLE book (
  id          NUMBER(10),
  title      VARCHAR2(50) NOT NULL,
  pubs       VARCHAR2(100),
  pub_date   DATE,
  author_id NUMBER(10),
  PRIMARY KEY(id),
  CONSTRAINT c_book_fk FOREIGN KEY (author_id)
  REFERENCES author(id)
  ON DELETE CASCADE
);
```

DAO
: 예제

- Primary Key 생성을 위해 Sequence를 작성하고 데이터베이스에 반영합니다

```
CREATE SEQUENCE seq_book
    START WITH 1
    INCREMENT BY 1
    MAXVALUE 1000000;

CREATE SEQUENCE seq_author
    START WITH 1
    INCREMENT BY 1
    MAXVALUE 1000000;
```

```
1. select seq_book.nextval from dual
   select seq_author.nextval from dual

2. insert 쿼리를 연습합니다.

3. 연습이 끝나면, 2) DDL를 실행해서 모든 테이블과 시퀀스를 다시 생성합니다.
```

DAO
: 예제

- AuthorVO의 작성
  - 데이터베이스 테이블의 필드와 매칭되는 필드들을 가진 VO 객체를 생서

```
public class AuthorVO {
    private Long id;
    private String name;
    private String bio;

    public Long getId() { return id; }
    public void setId(Long id) {
        this.id = id;
    }

// ...
}
```

DAO
: 예제

- AuthorDAO의 작성
  - AuthorDAO 클래스를 생성하고 공통 부분이 커넥션 부분을 메서드로 분리

```
public class AuthorDAO {
    private Connection getConnection() throws SQLException {
        Connection conn = null;
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            String dburl = "jdbc:oracle:thin:@localhost:1521:xe";
            conn = DriverManager.getConnection(dburl, ID, PWD);
        } catch( ClassNotFoundException e ) {
             System.out.println( "JDBC Driver 를 찾을 수 없습니다." );
        }
        return conn;
    }
}
```

DAO
: 예제

- AuthorDAO의 작성
  - 데이터베이스 -> DTO로 변환하는 메서드를 작성
  - Connection, Statement, ResultSet 객체를 선언
  - 결과를 반환할 변수 선언

```
    public List<AuthorVO> getList() {
        List<AuthorVO> list = new ArrayList<AuthorVO>();

        Connection conn = null;
        Statement stmt = null;
        ResultSet rs = null;

    }
```

DAO
: 예제

- 쿼리의 실행과 데이터 바인딩 (in AuthorDAO)

```
public List<AuthorVO> getList() {
    // ... Continued
    try {
        conn = getConnection();
        stmt = conn.createStatement();

        String sql = "select id, name, bio from author";
        rs = stmt.executeQuery( sql );

        while( rs.next() ){
            Long no = rs.getLong( 1 );
            String name = rs.getString( 2 );
            String bio = rs.getString( 3 );

            AuthorVO authorVO = new AuthorVO();
            authorVO.setId(id);
            authorVO.setName(name);
            authorVO.setBio(bio);

            list.add( authorVO );
        }
    }
}
```

DAO
: 예제

- 예외 처리와 자원 반환
- 결과값 리턴
  (in AuthorDAO)

```
public List<AuthorVo> getList() {
    // ... Continued
    } catch ( SQLException e ) {
        System.out.println( "error:" + e );
    } finally {
        try {
            if( rs != null ) {
                rs.close();
            }
            if( stmt != null ) {
                stmt.close();
            }
            if(conn != null ) {
                conn.close();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    return list;
}
```

DAO
: 실습

- [실습] AuthorDAO를 완성해 봅시다
- 인터페이스는 다음과 같이 구성합니다.

Interface	설명
`public List<AuthorVO> getList()`	전체 Author의 List를 반환
`public AuthorVO get(Long id)`	개별 Author 객체를 반환
`public boolean insert(AuthorVO authorVO)`	AuthorVO 객체를 받아 author 테이블에 INSERT 성공하면 true, 실패하면 false 반환
`public Boolean delete(Long id)`	개별 Author를 테이블에서 삭제 DELETE 성공하면 true, 실패하면 false 반환
`public boolean update(AuthorVO authorVO)`	AuthorVO를 받아서 UPDATE 성공하면 true, 실패하면 false 반환

DAO
: 실습

- [실습] 사원 정보 검색 프로그램 (DAO 버전)
  : 클래스 Scanner를 사용하여 사원 이름을 입력 받아 사원 정보를 검색하는 프로그램을 작성해 봅시다
  - 부분 이름 검색이 가능해야 합니다
  - 성, 이름 컬럼에 대해 OR 검색이 되어야 합니다
  - 출력 사원 정보는 이름 성, Email, 전화번호, 입사일 입니다
  - HRApp 프로젝트에서 HRMain Class에 main 메서드가 있어야 합니다

DAO
: 실습

- [실습 2] 급여 검색 프로그램 (DAO 버전)
  : 사용자로부터 최소 급여와 최대 급여를 입력 받아 급여가 이 범위 내에 속하는 사원의 정보를 출력하는 프로그램을 작성해 봅시다
  - 정렬은 salary 오름차순입니다
  - HRApp 프로젝트에서 HRSalary 클래스에 main 메서드가 있어야 합니다
  - 결과 예시

```
2000 10000
============================================================
Kevin Freeney                3000
Donald Oconnell              4000
Pat Fay                      90000
```