JDBC 미션#2

DAO, VO를 사용해 JDBC로 업무메서드 구현하기

A014손은빈

1. DAO공통부

```
private Connection conn;
public EmployeeDAO() throws ClassNotFoundException, SQLException {
    // 1. driver는 각각 DBMS회사에서 구현 - 기본 제공 x, 수동으로 제공
    Class.forName("oracle.jdbc.driver.OracleDriver");
    System.out.println("1.driver loading OK");

    // 2. DB연결 서버의 정보 및 계정
    String url = "jdbc:oracle:thin:@127.0.0.1:1521:XE";
    String id = "hr";
    String pw = "hr";
    conn=DriverManager.getConnection(url, id, pw);
    System.out.println("2.DBMS 연결 OK");
}
```

업무 메서드를 구현할 DAO클래스를 생성할 때,
드라이버를 로딩하고 DBMS와 연결을 생성함

2. 업무 분석해 쿼리 작성하기

```
/**-- 5. 수강신청 테이블에서
 *       특정일(1998년 3월 1일)에서 특정일(3월3일) 사이에 수강신청 한
 *       강좌번호, 학생번호, 수강신청일을 검색하시오.
    String sql =
            "SELECT" +
            "  lcode" +
            "  , scode" +
            "  , edate" +
            " FROM " +
            "  enrollments" +
            " WHERE edate BETWEEN ? AND ?";
```

가변 입력값에 '?'를 사용함

3. 입력값, 출력값 고려해 메서드 선언하기

```
public Collection<EnrollmentVO> getEnrollments(String startDate, String endDate) {
    Collection<EnrollmentVO> list = new ArrayList();

    return list;
}
```

레코드 행이 다수면 출력타입은 Collection임.
레코드 헹의 속성이 다수면 VO를 사용함.

4. 인터페이스를 사용해 dbms로 쿼리 보내고 결과받기

```
String sql =
        "SELECT" +
        "  lcode" +
        "  , scode" +
        "  , edate" +
        " FROM " +
        "  enrollments" +
        " WHERE edate BETWEEN ? AND ?";

PreparedStatement pstmt=conn.prepareStatement(sql);
pstmt.setString(1, startDate);
pstmt.setString(2, endDate);
ResultSet rs=pstmt.executeQuery();
```

가변 입력값이 있으면 java.sql.PreparedStatement를,
가변 입력값이 없으면 java.sql.Statement를 사용함.
Java.sql.ResultSet으로 SELECT한 레코드행 데이터를 받을 수 있음

5. 결과 레코드행에서 값 추출하기

```
while(rs.next()) {
    list.add(new EnrollmentVO(rs.getString(1)
                              , rs.getString(2)
                              , rs.getDate(3)));
}

return list;
```

레코드의 속성별 값에 인덱스와 자료형으로 접근할 수 있음

6. TEST결과- EnrollmentDAO

```java
/**
 * -- 20. 수강신청한 과목들 중에서
 *         최고 점수를 받은 학생들의 정보를 조회하시오.
 * */
public Collection<EnrollmentVO> getHighestScoredScodes()
        throws SQLException {
    Collection<EnrollmentVO> list = new ArrayList();
    String sql =
            "SELECT" +
            "   scode" +
            "   , lcode" +
            "FROM" +
            "  enrollments" +
            "WHERE (lcode, grade) IN" +
            "  (" +
            "    SELECT" +
            "      lcode" +
            "      , max(grade)" +
            "    FROM enrollments" +
            "    GROUP BY lcode" +
            "  )";
    Statement stmt=conn.createStatement();
    ResultSet rs=stmt.executeQuery(sql);
    while(rs.next()) {
        list.add(new EnrollmentVO(rs.getString(2)
                                  , rs.getString(1)));
    }
    return list;
}
```

```java
21          EnrollmentDAO enDao = new EnrollmentDAO();
22          System.out.println(enDao.getHighestScoredScodes());
```

Console

<terminated> DAOTest (1) [Java Application] C:₩Program Files₩Java₩jdk1.8.0_202₩bin₩javaw.exe (2022. 4. 7. 오전 12:46:23)

```
1.driver loading OK
2.DBMS 연결 OK
[EnrollmentVO [lcode=A109, scode=92414029, edate=null, grade=0]
, EnrollmentVO [lcode=E221, scode=92514009, edate=null, grade=0]
, EnrollmentVO [lcode=C301, scode=92414029, edate=null, grade=0]
, EnrollmentVO [lcode=C401, scode=95414058, edate=null, grade=0]
, EnrollmentVO [lcode=C312, scode=92454018, edate=null, grade=0]
]
```

6. TEST결과- EmployeeDAO

```java
/**
 * -- 40. 관리자 번호 및 해당 관리자에 속한 사원의 최저 급여, 인원수를 표시합니다.
    관리자를 알 수 없는 사원 및 최저 급여가 3000 이상인 그룹 제외시키고
    급여를 기준으로 출력 결과를 내림차순으로 정렬하시오.
 */
public Collection<int[]> getEmployeeDataOfManager(int maxSalary)
        throws SQLException {
    Collection<int[]> c = new ArrayList<int[]>();
    String sql = "SELECT" +
            "  manager_id" +
            "  ,MIN(salary)" +
            "  ,COUNT(employee_id)" +
            " FROM" +
            "  employees" +
            " WHERE manager_id IS NOT NULL" +
            " GROUP BY manager_id" +
            " HAVING MIN(salary) <= ?" +
            " ORDER BY MIN(salary)";
    PreparedStatement pstmt=conn.prepareStatement(sql);
    pstmt.setInt(1, maxSalary);
    ResultSet rs=pstmt.executeQuery();
    while(rs.next()) {
        c.add(new int[]{rs.getInt(1), rs.getInt(2), rs.getInt(3)});
    }
    return c;
}
```

```java
16          EmployeeDAO emDao=new EmployeeDAO();
17          for (int[] arr : emDao.getEmployeeDataOfManager(3000)) {
18          System.out.println(Arrays.toString(arr));
19          }
```

Console

<terminated> DAOTest (1) [Java Application] C:₩Program Files₩Java₩jdk1.8.0_202₩bin₩javaw.exe (2022. 4. 7. 오전 12:50:00)

```
1.driver loading OK
2.DBMS 연결 OK
[121, 2100, 8]
[120, 2200, 8]
[122, 2200, 8]
[114, 2500, 5]
[124, 2500, 8]
[123, 2500, 8]
```

6. TEST결과- EmployeeDAO

```java
/**
 * -- 40. 특정 관리자 번호 및 해당 관리자에 속한 사원의 최저 급여, 인원수를 표시합니다.
    관리자를 알 수 없는 사원 및 최저 급여가 특정액수 이상인 그룹 제외시키고
    급여를 기준으로 출력 결과를 내림차순으로 정렬하시오.
 */
public Map<String, Integer> getEmployeeDataOfManager(int ManagerId, int maxSalary)
        throws SQLException {
    Map<String, Integer> m = null;
    String sql = "SELECT" +
            "  manager_id" +
            "  ,MIN(salary)" +
            "  ,COUNT(employee_id)" +
            " FROM" +
            "  employees" +
            " WHERE manager_id = ?" +
            " GROUP BY manager_id" +
            " HAVING MIN(salary) <= ?" +
            " ORDER BY MIN(salary)";
    PreparedStatement pstmt=conn.prepareStatement(sql);
    pstmt.setInt(1, ManagerId);
    pstmt.setInt(2, maxSalary);
    ResultSet rs=pstmt.executeQuery();
    if(rs.next()) {
        m = new LinkedHashMap<String, Integer>();
        m.put("관리자번호", rs.getInt(1));
        m.put("사원최저급여", rs.getInt(2));
        m.put("사원인원수", rs.getInt(3));
    }
    return m;
}
```

```java
16          EmployeeDAO emDao=new EmployeeDAO();
17
18          System.out.println(emDao.getEmployeeDataOfManager(121, 3000));
```

Console

<terminated> DAOTest (1) [Java Application] C:₩Program Files₩Java₩jdk1.8.0_202₩bin₩javaw.exe (2022. 4. 7. 오전 12:54:13)

```
1.driver loading OK
2.DBMS 연결 OK
{관리자번호=121, 사원최저급여=2100, 사원인원수=8}
```

6. TEST결과- LectureDAO

```java
/**
 * -- 6. 강좌 테이블에서
 *       강좌이름이 특정 단어를 포함하는 강좌의
 *       강좌번호, 강좌이름, 담당교수, 강의시간수를 검색하시오.
 * */
public Collection<LectureVO> getLectures(String includedString)
        throws SQLException{
    Collection<LectureVO> list = new ArrayList<LectureVO>();
    String sql =
            "SELECT" +
            "  lcode" +
            "  , lname" +
            "  , hours" +
            "  , instructor" +
            " FROM" +
            "  lectures" +
            " WHERE " +
            "  lname LIKE ?";
    PreparedStatement pstmt=conn.prepareStatement(sql);
    pstmt.setString(1, "%"+includedString+"%");
    ResultSet rs=pstmt.executeQuery();
    while(rs.next()) {
        list.add(new LectureVO(rs.getString(1)
                               , rs.getString(2)
                               , rs.getInt(3)
                               , rs.getString(4)));
    }
    return list;
}
```

```java
30          LectureDAO leDao = new LectureDAO();
31
32          System.out.println(leDao.getLectures("DBMS"));
33
```

Console

<terminated> DAOTest (1) [Java Application] C:₩Program Files₩Java₩jdk1.8.0_202₩bin₩javaw.exe (2022. 4. 7. 오전 12:58:50)

```
1.driver loading OK
2.DBMS 연결 OK
[LectureVO [lcode=C401, lname=DBMS, hours=3, instructor=221]
]
```

6. TEST결과- SchoolDAO

```java
/**
 * -- 15. 특정 과목명의 강의를 강의하는
 *        교수명, 이 과목을 수강신청 한 학생들의 학과, 이름, 성적을 검색하시오.
 * */
public Collection<SchoolVO> getSchoolDatas(String lname)
        throws SQLException{
    Collection<SchoolVO> list = new ArrayList();
    String sql =
            "SELECT" +
            "  p.pname" +
            "  , s.department" +
            "  , s.sname" +
            "  , e.grade" +
            " FROM " +
            "  professors p, students s"
           + ", enrollments e, lectures l" +
            " WHERE s.scode = e.scode" +
            " AND e.lcode = l.lcode" +
            " AND l.instructor = p.pcode" +
            " AND l.lname = ?";
    PreparedStatement pstmt=conn.prepareStatement(sql);
    pstmt.setString(1, lname);
    ResultSet rs=pstmt.executeQuery();
    while(rs.next()) {
        list.add(new SchoolVO(rs.getString(1)
                              , rs.getString(2)
                              , rs.getString(3)
                              , rs.getFloat(4)));
    }
    return list;
}
```

```java
34          SchoolDAO scDao = new SchoolDAO();
35
36          System.out.println(scDao.getSchoolDatas("DBMS"));
```

Console

<terminated> DAOTest (1) [Java Application] C:₩Program Files₩Java₩jdk1.8.0_202₩bin₩javaw.exe (2022. 4. 7. 오전 1:03:59)

```
1.driver loading OK
2.DBMS 연결 OK
[SchoolVO [pname=JameLee, department=computer engineering, sname=ChangKim, grade=85.0]
, SchoolVO [pname=JameLee, department=computer engineering, sname=HyePark, grade=90.0]
, SchoolVO [pname=JameLee, department=electronic engineering, sname=MinKim, grade=70.0]
]
```

6. TEST결과- ProfessorDAO

```java
/**
 * -- 37.'HanOh' 교수의 직급을 'assistant professor'에서 'associate professor'로 변경하시오.
 *           - 단, 수행결과화면에 변경한 행수의 결과를 표기하시오. 급여도 500000 추가
 * */
public boolean setProfessor(String updatePosition, float addSalary)
        throws SQLException {
    boolean result = false;

    String sql =
            "UPDATE professors p" +
            "SET p.position = ?"
            + " , p.salary=salary+?" +
            "WHERE p.pname = 'HanOh'";
    PreparedStatement pstmt=conn.prepareStatement(sql);
    pstmt.setString(1, updatePosition);
    pstmt.setFloat(2, addSalary);
    int num=pstmt.executeUpdate();
    result = (num ==1);
    return result;
}
```

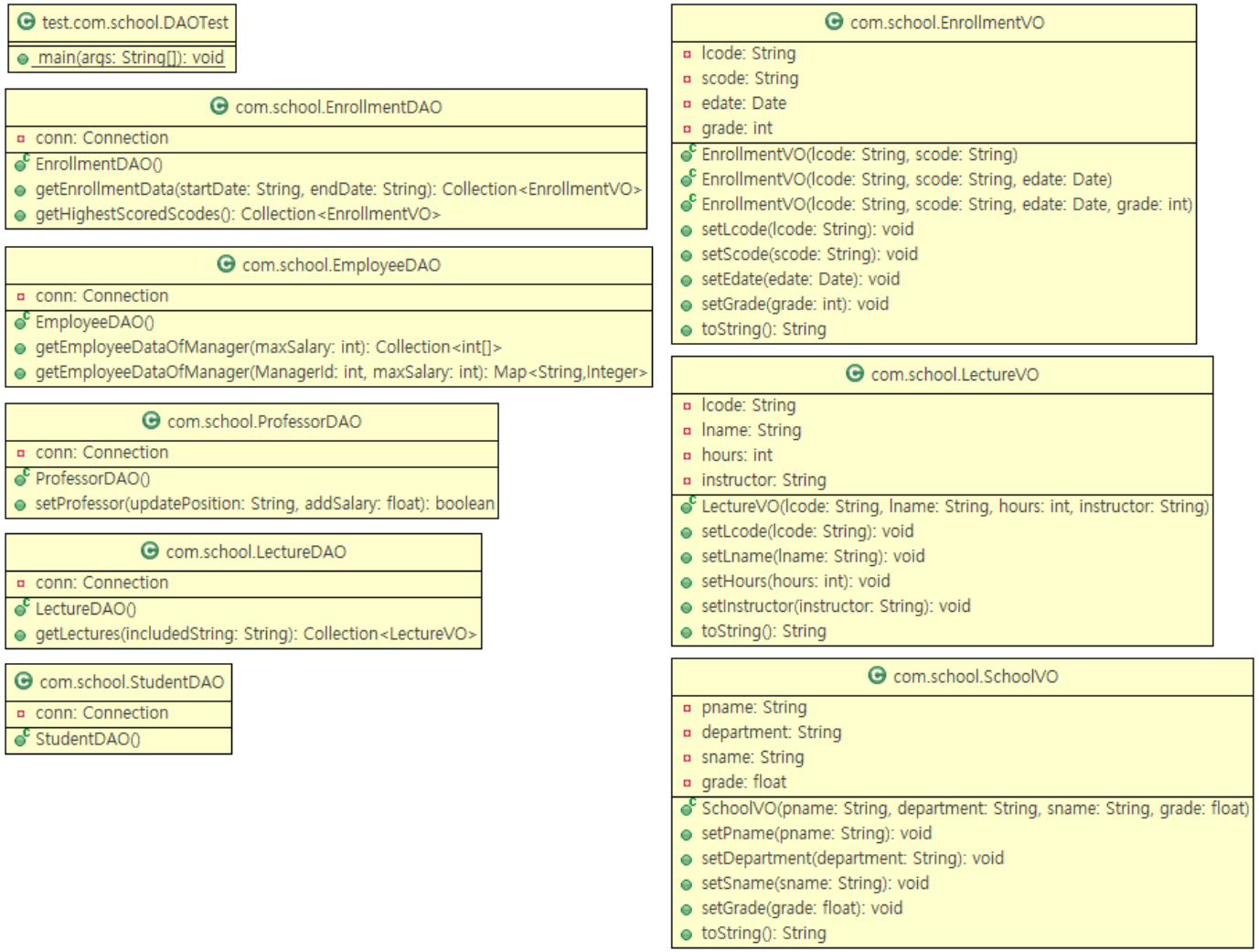
```java
34          ProfessorDAO prDao = new ProfessorDAO();
35          System.out.println(prDao.setProfessor("associate professor", 500000));
36      }
```

Console

<terminated> DAOTest (1) [Java Application] C:₩Program Files₩Java₩jdk1.8.0_202₩bin₩javaw.exe (2022. 4. 7. 오전 1:10:43)

```
1.driver loading OK
2.DBMS 연결 OK
true
```

7. 클래스 다이어그램


test.com.school.DAOTest
main(args: String[]): void
com.school.EnrollmentDAO
conn: Connection
EnrollmentDAO()
getEnrollmentData(startDate: String, endDate: String): Collection<EnrollmentVO>
getHighestScoredScodes(): Collection<EnrollmentVO>
com.school.EmployeeDAO
conn: Connection
EmployeeDAO()
getEmployeeDataOfManager(maxSalary: int): Collection<int[]>
getEmployeeDataOfManager(ManagerId: int, maxSalary: int): Map<String,Integer>
com.school.ProfessorDAO
conn: Connection
ProfessorDAO()
setProfessor(updatePosition: String, addSalary: float): boolean
com.school.LectureDAO
conn: Connection
LectureDAO()
getLectures(includedString: String): Collection<LectureVO>
com.school.StudentDAO
conn: Connection
StudentDAO()
com.school.EnrollmentVO
lcode: String
scode: String
edate: Date
grade: int
EnrollmentVO(lcode: String, scode: String)
EnrollmentVO(lcode: String, scode: String, edate: Date)
EnrollmentVO(lcode: String, scode: String, edate: Date, grade: int)
setLcode(lcode: String): void
setScode(scode: String): void
setEdate(edate: Date): void
setGrade(grade: int): void
toString(): String
com.school.LectureVO
lcode: String
lname: String
hours: int
instructor: String
LectureVO(lcode: String, lname: String, hours: int, instructor: String)
setLcode(lcode: String): void
setLname(lname: String): void
setHours(hours: int): void
setInstructor(instructor: String): void
toString(): String
com.school.SchoolVO
pname: String
department: String
sname: String
grade: float
SchoolVO(pname: String, department: String, sname: String, grade: float)
setPname(pname: String): void
setDepartment(department: String): void
setSname(sname: String): void
setGrade(grade: float): void
toString(): String