JDBC 종합실습 리뷰

업무 분석 더하고 오버로딩할 것 등등

A014손은빈

0. 기존 문서 수정

생성자 오버라이딩 -> 생성자 오버로딩

상속한 클래스의 메서드를 재정의하는 것이 아니고 매개변수 타입과 개수를 달리 하는 것이므로 오버라이딩이 아니라 오버로딩임

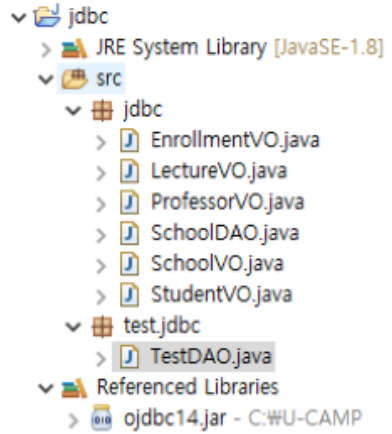
4. VO 구현하기 - 방법1, 통합VO

- SchoolVO.java
- 1개 VO안에서 생성자 오버로딩해 구분할 수 없는 업무들이 있음

```
5  public class SchoolVO {
6      /**enrollments*/
7  //  private String lcode;
8
9      /**professors*/
10     private String pcode;
11     private String department;
12     private Date hireDate;
13
14     /**lectures*/
15     private String lcode;
16     private String lname;
17     private String room;
18     private int capacity;
19
20     /**students*/
21     private String scode;
22     private String sname;
23
24     /**--1. 모든 교수의 번호 확인 */
25     public SchoolVO(String pcode) {
26         this(pcode, null, null, null, null, null, 0, null, null);
27     }
28
29     /**--2. 특정 교수(번호)의 소속학과, 임용일자 확인 */
30     public SchoolVO(String department, Date hireDate) {
31         this(null, department, hireDate, null, null, null, 0, null, null);
32     }
33
34     /**--3. 특정 과목('DBMS') 과목의 강좌 번호 확인 */
35 //  public SchoolVO(String lcode) { -> 이상에서 오버라이딩 구분의 제한으로, 테이블 별로 VO를 구분함
36 //      this(null, department, hireDate, null, null, null, 0, null, null);
37 //  }
```

4. VO 구현하기 - 방법2, 테이블별VO

1개 VO 방식의 오버로딩 불가 문제 -> 테이블별로 4개 VO를 구현함



1. 모든 교수의 번호 확인

다양한 경우를 대비해 VO를 재사용하는 경우와 그렇지 않고 String을 쓰는 경우를 모두 구현해 둘 수 있음.

```
/**--1. 모든 교수의 번호 확인 */
public Collection<ProfessorVO> getProfessors() throws SQLException{
    Collection<ProfessorVO> list = new ArrayList<ProfessorVO>();
    String sql =
            "   SELECT " +
            "     pcode" +
            "   FROM professors";
    Statement stmt=conn.createStatement();
    ResultSet rs=stmt.executeQuery(sql);
    while(rs.next()) {
        list.add(new ProfessorVO(rs.getString(1)));
    }
    return list;
}

// VO를 사용하지 않은 컬렉션을 반환값으로하여 오버로딩
public Collection<String> getProfessorsAsString() throws SQLException{
    Collection<String> list = new ArrayList<String>();
    String sql =
            "   SELECT " +
            "     pcode" +
            "   FROM professors";
    Statement stmt=conn.createStatement();
    ResultSet rs=stmt.executeQuery(sql);
    while(rs.next()) {
        list.add(rs.getString(1));
    }
    return list;
}
```

2. 특정 교수(번호)의 소속학과, 임용일자 확인

```java
/**--2. 특정 교수(번호)의 소속학과, 임용일자 확인*/
public ProfessorVO getProfessorData(String pcode)
        throws SQLException{
    ProfessorVO v = null;
    String sql=
            "   SELECT " +
            "     department" +
            "     , hiredate" +
            "   FROM professors" +
            "   WHERE pcode = ?";
    PreparedStatement pstmt=conn.prepareStatement(sql);
    pstmt.setString(1, pcode);
    ResultSet rs=pstmt.executeQuery();
    if(rs.next()) {
        v = new ProfessorVO(rs.getString(1)
                , rs.getDate(2));
    }
    return v;
}
```

```java
// 매개인자 없이 오버로딩 -> 특정 매개인자 변수의 값이 정해진 경우
public ProfessorVO getProfessorData()
        throws SQLException{
    ProfessorVO v = null;
    String sql=
            "   SELECT " +
            "     department" +
            "     , hiredate" +
            "   FROM professors" +
            "   WHERE pcode = '221'";
    Statement stmt=conn.createStatement();
    ResultSet rs=stmt.executeQuery(sql);
    if(rs.next()) {
        v = new ProfessorVO(rs.getString(1)
                , rs.getDate(2));
    }
    return v;
}
```

2. 특정 교수(번호)의 소속학과, 임용일자 확인

LinkedHashMap같은 존재하는 메모리 자료구조를 사용하면 안정적이라는 장점이 있음.

```
// 자바 메모리를 반환값으로 한 메서드
public Map<String, Object> getProfessorDataWithMap(String pcode)
        throws SQLException {
    Map<String, Object> m = null;
    String sql=
            "SELECT " +
            "  department" +
            "  , hiredate" +
            " FROM professors" +
            " WHERE pcode = ?";
    PreparedStatement pstmt=conn.prepareStatement(sql);
    pstmt.setString(1, pcode);
    ResultSet rs=pstmt.executeQuery();
    if(rs.next()) {
        m = new LinkedHashMap<String, Object>();
        m.put("소속학과", rs.getString(1));
        m.put("고용일", rs.getDate(2));
    }
    return m;
}
```

3. 특정 과목('DBMS') 과목의 강좌 번호 확인

매개인자가 있는 경우와 매개인자가 없는 경우, 즉, 매개인자 변수 값을 알고 있는 경우를 구분해 메서드를 오버로딩 할 수 있음

```java
/**--3. 특정 과목('DBMS') 과목의 강좌 번호 확인 */
public LectureVO getLcode(String lname) throws SQLException {
    LectureVO v = null;
    String sql =
            "   SELECT" +
            "     lcode" +
            "   FROM lectures" +
            "   WHERE lname = ?";
    PreparedStatement pstmt=conn.prepareStatement(sql);
    pstmt.setString(1, lname);
    ResultSet rs=pstmt.executeQuery();
    if(rs.next()) {
        v = new LectureVO(rs.getString(1));
    }
    return v;
}
// 단일 결과값을 문자열로 반환하는 경우
public String getLcodeAsString(String lname) throws SQLException {
    String v = null;
    String sql =
            "   SELECT" +
            "     lcode" +
            "   FROM lectures" +
            "   WHERE lname = 'DBMS'";
    PreparedStatement pstmt=conn.prepareStatement(sql);
    pstmt.setString(1, lname);
    ResultSet rs=pstmt.executeQuery();
    if(rs.next()) {
        v = rs.getString(1);
    }
    return v;
}
```

4. 특정 강좌(강좌번호)에 해당하는 강좌이름, 강의실, 최대수강인원 확인

```java
/**--4. 특정 강좌(강좌번호)에 해당하는 강좌이름, 강의실, 최대수강인원 확인 */
public LectureVO getLectureData(String lcode)
        throws SQLException {
    LectureVO v = null;
    String sql =
        "   SELECT" +
        "     lname" +
        "     , room" +
        "     , capacity " +
        "   FROM lectures " +
        "   WHERE lcode = ?";
    PreparedStatement pstmt=conn.prepareStatement(sql);
    pstmt.setString(1, lcode);
    ResultSet rs=pstmt.executeQuery();
    if(rs.next()) {
        v = new LectureVO(rs.getString(1)
                        , rs.getString(2)
                        , rs.getInt(3));
    }
    return v;
}
```

```java
// 매개인자 없이 오버로딩, 매개인자변수 값을 알고 있는 경우
public LectureVO getLectureData()
        throws SQLException {
    LectureVO v = null;
    String sql =
        "   SELECT" +
        "     lname" +
        "     , room" +
        "     , capacity " +
        "   FROM lectures " +
        "   WHERE lcode = 'C301'";
    Statement stmt=conn.createStatement();
    ResultSet rs=stmt.executeQuery(sql);
    if(rs.next()) {
        v = new LectureVO(rs.getString(1)
                        , rs.getString(2)
                        , rs.getInt(3));
    }
    return v;
}
```

5. 학생 이름(ChangKim)으로 수강신청한 강좌번호 확인

```java
/**--5. 학생 이름(ChangKim)으로 수강신청한 강좌번호 확인*/
public Collection<EnrollmentVO> getLcodebySname(String sname)
        throws SQLException {
    Collection<EnrollmentVO> list =
            new ArrayList<EnrollmentVO>();
    String sql=
            "   SELECT  " +
            "     e.lcode " +
            "   FROM enrollments e, students s " +
            "   WHERE e.scode = s.scode " +
            "   AND s.sname = ?";
    PreparedStatement pstmt=conn.prepareStatement(sql);
    pstmt.setString(1, sname);
    ResultSet rs=pstmt.executeQuery();
    while(rs.next()) {
        list.add(new EnrollmentVO(rs.getString(1)));
    }
    return list;
}
```

```java
// 매개인자 없이 오버로딩, 매개인자변수 값을 알고 있는 경우
public Collection<EnrollmentVO> getLcodebySname()
        throws SQLException {
    Collection<EnrollmentVO> list =
            new ArrayList<EnrollmentVO>();
    String sql=
            "   SELECT  " +
            "     e.lcode " +
            "   FROM enrollments e, students s " +
            "   WHERE e.scode = s.scode " +
            "   AND s.sname = 'ChangKim'";
    Statement stmt=conn.createStatement();
    ResultSet rs=stmt.executeQuery(sql);
    while(rs.next()) {
        list.add(new EnrollmentVO(rs.getString(1)));
    }
    return list;
}
```

6. 특정 학과('computer engineering')과 교수들이 지도하는 학생들의 학번,이름 확인

```java
/**--6. 특정 학과('computer engineering')과 교수들이 지도하는 학생들의 학번,이름 확인 */
public Collection<StudentVO> getStudents(String departmentName)
        throws SQLException{
    Collection<StudentVO> list = new ArrayList<StudentVO>();
    String sql=
            "   SELECT  " +
            "     s.scode " +
            "     , s.sname " +
            "   FROM students s, professors p " +
            "   WHERE s.advisor = p.pcode " +
            "   AND p.department = ?";
    PreparedStatement pstmt=conn.prepareStatement(sql);
    pstmt.setString(1, departmentName);
    ResultSet rs=pstmt.executeQuery();
    while(rs.next()) {
        list.add(new StudentVO(rs.getString(1), rs.getString(2)));
    }
    return list;
}
```

```java
// 매개인자 없이 오버로딩, 매개인자변수 값을 알고 있는 경우
public Collection<StudentVO> getStudents()
        throws SQLException{
    Collection<StudentVO> list = new ArrayList<StudentVO>();
    String sql=
            "   SELECT  " +
            "     s.scode " +
            "     , s.sname " +
            "   FROM students s, professors p " +
            "   WHERE s.advisor = p.pcode " +
            "   AND p.department = 'computer engineering'";
    Statement stmt=conn.createStatement();
    ResultSet rs=stmt.executeQuery(sql);
    while(rs.next()) {
        list.add(new StudentVO(rs.getString(1), rs.getString(2)));
    }
    return list;
}
```

7. 특정 학생 정보 등록(단 학생번호는 PK)

중복 값을 인서트 시도하면 무결성 제약 조건 오류가 발생함

```
32 //        7. 특정 학생 정보 등록(단 학생번호는 PK)
33         System.out.println(sDao.addStudent("92414505"
34                 , "NamPark", "computer engineering", "3", "73/11/11", "221"));
```

Markers Properties Servers Data Source Explorer Snippets Console

```
<terminated> TestDAO [Java Application] C:₩Program Files₩Java₩jdk1.8.0_202₩bin₩javaw.exe (2022. 4. 7. 오후 8:27:41)
1.driver loading OK
dbms OK
true
```

```
32 //        7. 특정 학생 정보 등록(단 학생번호는 PK)
33         System.out.println(sDao.addStudent("92414505"
34                 , "NamPark", "computer engineering", "3", "73/11/11", "221"));
```

Markers Properties Servers Data Source Explorer Snippets Console

```
<terminated> TestDAO [Java Application] C:₩Program Files₩Java₩jdk1.8.0_202₩bin₩javaw.exe (2022. 4. 7. 오후 8:28:43)
1.driver loading OK
dbms OK
Exception in thread "main" java.sql.SQLException: ORA-00001: 무결성 제약 조건(HR.PK_STUDENTS_SCODE)에 위배됩니다

        at oracle.jdbc.driver.DatabaseError.throwSqlException(DatabaseError.java:112)
        at oracle.jdbc.driver.T4CTTIoer.processError(T4CTTIoer.java:331)
        at oracle.jdbc.driver.T4CTTIoer.processError(T4CTTIoer.java:288)
        at oracle.jdbc.driver.T4C8Oall.receive(T4C8Oall.java:743)
        at oracle.jdbc.driver.T4CPreparedStatement.doOall8(T4CPreparedStatement.java:216)
        at oracle.jdbc.driver.T4CPreparedStatement.executeForRows(T4CPreparedStatement.java:955)
        at oracle.jdbc.driver.OracleStatement.doExecuteWithTimeout(OracleStatement.java:1169)
        at oracle.jdbc.driver.OraclePreparedStatement.executeInternal(OraclePreparedStatement.java:3285)
        at oracle.jdbc.driver.OraclePreparedStatement.executeUpdate(OraclePreparedStatement.java:3368)
        at jdbc.SchoolDAO.addStudent(SchoolDAO.java:290)
        at test.jdbc.TestDAO.main(TestDAO.java:33)
```

8. 학생정보를 기반으로 수강신청

```
36 //        8. 학생정보를 기반으로 수강신청
37          System.out.println(sDao.addEnrollment("C312", "96414404", "98/03/03", 85));
```

Markers Properties Servers Data Source Explorer Snippets Console

```
<terminated> TestDAO [Java Application] C:₩Program Files₩Java₩jdk1.8.0_202₩bin₩javaw.exe (2022. 4. 7. 오후 8:52:35)
1.driver loading OK
dbms OK
true
```

```
36 //        8. 학생정보를 기반으로 수강신청
37          System.out.println(sDao.addEnrollment("C312", "96414404", "98/03/03", 85));
```

Markers Properties Servers Data Source Explorer Snippets Console

```
<terminated> TestDAO [Java Application] C:₩Program Files₩Java₩jdk1.8.0_202₩bin₩javaw.exe (2022. 4. 7. 오후 8:53:02)
1.driver loading OK
dbms OK
Exception in thread "main" java.sql.SQLException: ORA-00001: 무결성 제약 조건(HR.PK_ENROLLMENTS_LCODE)에 위배됩니다

        at oracle.jdbc.driver.DatabaseError.throwSqlException(DatabaseError.java:112)
        at oracle.jdbc.driver.T4CTTIoer.processError(T4CTTIoer.java:331)
        at oracle.jdbc.driver.T4CTTIoer.processError(T4CTTIoer.java:288)
        at oracle.jdbc.driver.T4C8Oall.receive(T4C8Oall.java:743)
        at oracle.jdbc.driver.T4CPreparedStatement.doOall8(T4CPreparedStatement.java:216)
        at oracle.jdbc.driver.T4CPreparedStatement.executeForRows(T4CPreparedStatement.java:955)
        at oracle.jdbc.driver.OracleStatement.doExecuteWithTimeout(OracleStatement.java:1169)
        at oracle.jdbc.driver.OraclePreparedStatement.executeInternal(OraclePreparedStatement.java:3285)
        at oracle.jdbc.driver.OraclePreparedStatement.executeUpdate(OraclePreparedStatement.java:3368)
        at jdbc.SchoolDAO.addEnrollment(SchoolDAO.java:312)
        at test.jdbc.TestDAO.main(TestDAO.java:37)
```

9. 학번과 수강신청한 과목을 보고 수강 신청 변경

```java
/**--9. 학번과 수강신청한 과목을 보고 수강 신청 변경 */
public boolean setEnrollment(String updateLcode
                           , String scode
                           , String lcode)
                                  throws SQLException {
    boolean result = false;
    String sql=
            "   UPDATE enrollments " +
            "   SET lcode =  ? " +
            "   WHERE scode=? AND lcode=?";
    PreparedStatement pstmt=conn.prepareStatement(sql);
    pstmt.setString(1, updateLcode);
    pstmt.setString(2, scode);
    pstmt.setString(3, lcode);
    int num=pstmt.executeUpdate();
    result = (num==1);
    return result;
}
```

```java
// 추가업무분석 - 수강신청 변경하면, 성적은 없을 것임
// 그리고 변경은 동일에 했다고 가정함 -> 현재일시가 테이블 데이터와 괴리가 크므로
public boolean setEnrollment2(String updateLcode
        , String scode
        , String lcode)
                throws SQLException {
    boolean result = false;
    String sql=
    "UPDATE enrollments" +
    " SET lcode= ?, grade=null" +
    " WHERE scode= ? AND lcode= ?";
    PreparedStatement pstmt=conn.prepareStatement(sql);
    pstmt.setString(1, updateLcode);
    pstmt.setString(2, scode);
    pstmt.setString(3, lcode);
    int num=pstmt.executeUpdate();
    result = (num==1);
    return result;
}
```

수강신청 정보 변경시, 성적을 null로 할 수 있을 것임.
수강신청 정보 변경시, 날짜는 다를 수 있음.
본인은 테이블 데이터의 날짜와 현재날짜의 괴리가 크므로 동일에 변경했다고 가정했음.

10. 학생이 수강 신청한 특정 과목 수강 신청 삭제

존제하지 않는 레코드 삭제를 시도하면executeUpdate()의 반환값이 0임

```
42 //        10. 학생이 수강 신청한 특정 과목 수강 신청 삭제
43          System.out.println(sDao.removeEnrollment("92414033", "C401"));
```

Markers Properties Servers Data Source Explorer Snippets Console

<terminated> TestDAO [Java Application] C:₩Program Files₩Java₩jdk1.8.0_202₩bin₩javaw.exe (2022. 4. 7. 오후 9:10:41)

```
1.driver loading OK
dbms OK
true
```

```
42 //        10. 학생이 수강 신청한 특정 과목 수강 신청 삭제
43          System.out.println(sDao.removeEnrollment("92414033", "C401"));
```

Markers Properties Servers Data Source Explorer Snippets Console

<terminated> TestDAO [Java Application] C:₩Program Files₩Java₩jdk1.8.0_202₩bin₩javaw.exe (2022. 4. 7. 오후 9:11:07)

```
1.driver loading OK
dbms OK
false
```