Tutorial 5

Hal yang saya pelajari adalah untuk memgambil dua buah objek yang berbeda pada mapper dapat digunakan results dan result. Sehingga memungkinkan untuk melakukan operasi sekaligus pada 2 objek yang berbeda.

1. Ubah method selectAllStudents pada kelas StudentMapper agar halaman viewall menampilkan semua student beserta daftar kuliah yang diambil.

   Menambahkan pada viewall.html

```
<h3 th:text="'Name = ' + ${student.name}">Student Name</h3>
<h3 th:text="'GPA = ' + ${student.gpa}">Student GPA</h3>
<h3>Kuliah yang diambil</h3>
<ul th:each="course,iterationStatus: ${student.courses}">
    <li th:text="${course.name} + '-' + ${course.credits} + ' sks'" >
        Nama kuliah-X SKS
    </li>
</ul>
```

   Mengubah pada studentMapper

```
@Select("select npm, name, gpa from student")
@Results(value = {
    @Result(property="npm", column="npm"),
    @Result(property="name", column="name"),
    @Result(property="gpa", column="gpa"),
    @Result(property="courses", column="npm",
        javaType = List.class,
        many=@Many(select="selectCourses"))
})
List<StudentModel> selectAllStudents ();
```

2. Buatlah view pada halaman http://localhost:8080/course/view/{id} untuk Course sehingga dapat menampilkan data course beserta Student yang mengambil.

   Membuat file view-course.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>View Course by ID</title>
    </head>
    <body>
        <h3 th:text="'ID = ' + ${course.idCourse}">ID</h3>
        <h3 th:text="'Nama = ' + ${course.name}">Nama</h3>
        <h3 th:text="'SKS = ' + ${course.credits}">SKS</h3>

        <h3>Mahasiswa yang Mengambil</h3>
        <ul th:each="student, iterartionStatus: ${course.students}">
            <li th:text="${student.npm} + ' - ' + ${student.name}">
                Nama Student yang mengambil
            </li>
        </ul>
    </body>
</html>
```

Membuat courseModel

```java
package com.example.model;
import java.util.List;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class CourseModel {
    private String idCourse;
    private String name;
    private Integer credits;
    private List<StudentModel> students;
}
```

Menambah method selectCourse pada studentService

```java
CourseModel selectCourse (String id);
```

Menambah method override pada StudentServiceDatabase

```java
@Override
public CourseModel selectCourse(String id) {
    CourseModel course = studentMapper.selectCourse(id);
    log.info ("select course with id " + course.getIdCourse());
    return course;
}
```

Menambah method pada studentMapper

```java
@Select("select id_course, name, credits from course where id_course = #{id}")
@Results(value = {
        @Result(property="idCourse", column="id_course"),
        @Result(property="name", column="name"),
        @Result(property="credits", column="credits"),
        @Result(property="students", column="id_course",
                javaType = List.class,
                many = @Many(select="selectStudentCourses"))
})
CourseModel selectCourse(@Param("id") String id);

@Select("select student.npm, name, gpa " +
        "from studentcourse join student " +
        "on studentcourse.npm = student.npm " +
        "where studentcourse.id_course = #{id}")
List<StudentModel> selectStudentCourses(@Param("id") String id);
```

Menambah method pada studentController

```
@RequestMapping("/course/view/{id}")
public String viewCourse (Model model,
        @PathVariable(value = "id") String id)
{
    CourseModel course = studentDAO.selectCourse (id);

    if (course != null) {
        model.addAttribute ("course", course);
        return "view-course";
    } else {
        model.addAttribute ("course", course);
        return "not-found";
    }
}
```