INT204 Server-side Programming II 64130500108

1. Create data for Student, Subject and StudentGrade then write all objects to table in database

Entities in server

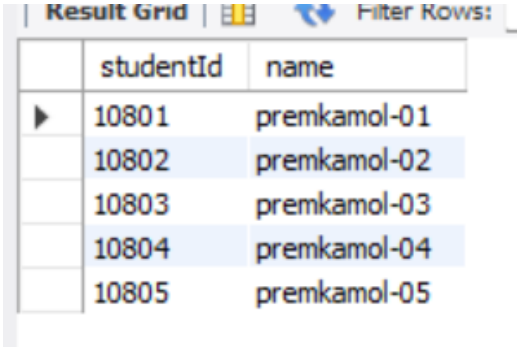
```java
@Getter
@Setter
@Table(name = "students")
public class Student {
    @Id
    @Column(name = "studentId",nullable = false,length = 10)
    private Integer id;
    @Column(name = "name",nullable = false,length = 100)
    private String name;
    @JsonIgnore
    @OneToMany(mappedBy = "StudentGrade")
    private Set<StudentGrade> studentGrades = new LinkedHashSet<>();

}
```

```java
@Getter
@Setter
@Table(name = "subjects")
public class Subject {
    @Id
    @Column(name = "subjectId",nullable = false,length = 10)
    private Integer id;
    @Column(name = "studentCode",nullable = false,length = 100)
    private String code;
    @Column(name = "title",nullable = false,length = 100)
    private String title;
    @Column(name = "credit",nullable = false,length = 100)
    private Double credit;
}
```

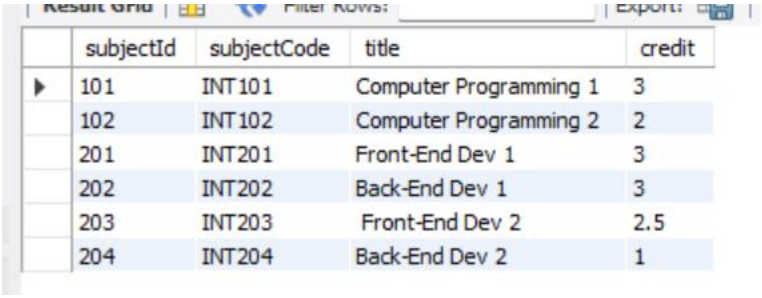
```java
@Getter
@Setter
@Table(name = "studentGrades")
public class StudentGrade {
    @Id
    @Column(name = "gradeId",nullable = false,length = 10)
    private Integer id;
    @Column(name = "grade",nullable = false,length = 100)
    private Double grade;
    @JsonIgnore
    @OneToMany(mappedBy = "Subject")
    private Set<Subject> subjects = new LinkedHashSet<>();

}
```

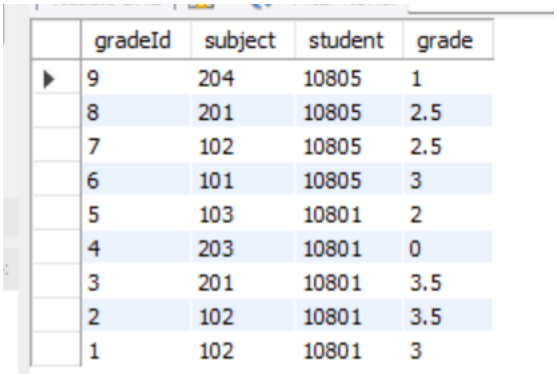


Result Grid | Filter Rows:

	studentId	name
▶	10801	premkamol-01
	10802	premkamol-02
	10803	premkamol-03
	10804	premkamol-04
	10805	premkamol-05



	subjectId	subjectCode	title	credit
▶	101	INT101	Computer Programming 1	3
	102	INT102	Computer Programming 2	2
	201	INT201	Front-End Dev 1	3
	202	INT202	Back-End Dev 1	3
	203	INT203	Front-End Dev 2	2.5
	204	INT204	Back-End Dev 2	1



	gradeId	subject	student	grade
▶	9	204	10805	1
	8	201	10805	2.5
	7	102	10805	2.5
	6	101	10805	3
	5	103	10801	2
	4	203	10801	0
	3	201	10801	3.5
	2	102	10801	3.5
	1	102	10801	3

2. List all student with grade, output format for each student as below

repository

```
2 usages
public interface StudentRepositories extends JpaRepository<Student, Integer> {
}
```

service

```
@Service
public class StudentService {
    1 usage
    @Autowired
    private StudentRepositories repositories;

    1 usage
    public List<Student> getAllStudent(){
    return repositories.findAll();
    }
}
```

Controller

```
@RestController
@RequestMapping("api/students")
public class StudentController {
    1 usage
    @Autowired
    private StudentService service;

    @GetMapping("")
    public List<Student> getStudent(){
        return  service.getAllStudent();
    }
}
```

3. Delete specified Student

* return 400 – Can’t remove student (grade not empty) * when student have grade info in StudentGrade.

* return 404 – Student does not exist * when specific student id does not exist.

```
6 usages
@RestController
public class ItemNotFoundException extends Throwable {
    @ExceptionHandler(ItemNotFoundException.class)
    @ResponseStatus(HttpStatus.NOT_FOUND)
    public ItemNotFoundException handleNoSuchElementFoundException(
            ItemNotFoundException exception) {
        return exception;
    }
    @ExceptionHandler(NumberFormatException.class)
    @ResponseStatus(HttpStatus.BAD_REQUEST)
    public ResponseStatusException handleNumberElementFoundException(RuntimeException exception){
        return new ResponseStatusException(HttpStatus.BAD_REQUEST, "Invalid number ja"+ exception.getMessage());
    }
}
```

```
    @ExceptionHandler(ItemNotFoundException.class)
    @ResponseStatus(HttpStatus.NOT_FOUND)
    public ItemNotFoundException handleNoSuchElementFoundException(
            ItemNotFoundException exception) {
        return exception;
    }
    @ExceptionHandler(NumberFormatException.class)
    @ResponseStatus(HttpStatus.BAD_REQUEST)
    public ResponseStatusException handleNumberElementFoundException(RuntimeException exception){
        return new ResponseStatusException(HttpStatus.BAD_REQUEST, "Invalid number ja"+ exception.getMessage());
    }
```