

**Due:**

Monday, 19-April-2021 by 23:59

**Deliverables:**

The following Java file should be submitted to Google Classroom by the due date and time specified above. Submissions received after the deadline will be subject to the late policy described in the syllabus.

- Assignment02\_{StudentNumber}.java
- Image files for screenshots of your progress:
  - May be separate image files or all pasted into a .docx or .pdf document
- 1. Assignment02\_{StudentNumber}\_firstCompile
  - This is likely the same as the final compile from the previous assignment
- 2. Assignment02\_{StudentNumber}\_firstCompileNoError
- 3. Assignment02\_{StudentNumber}\_firstTestRun
- 4. Assignment02\_{StudentNumber}\_firstTestRunNoError (optional)
- 5. Assignment02\_{StudentNumber}\_finalCompile
- 6. Assignment02\_{StudentNumber}\_finalRun

**Specifications:**

**Overview:** You will continue the program this semester to maintain the information for a university. Do not forget your headers with @author and @since information.

**Requirements:** Write and modify the following set of classes according to the following specifications:

1. Department
  - a. Attributes
    - i. ID: String
      1. Must be 3 or 4 characters
    - ii. Name: String
    - iii. Chair: Teacher
  - b. Methods
    - i. Constructor that takes the ID and name as a parameter. The chair will be set later as shown below.
    - ii. getID(): String and setID(ID: String)
    - iii. getName(): String and setName(name: String)
    - iv. getChair(): Teacher and setChair(chair: Teacher)
      1. Raises DepartmentMismatchException if Teacher is not in this department

- 2. Course
  - a. Attributes
    - i. Department: Department
    - ii. Teacher: Teacher
    - iii. Other attributes – no change from previous Assignment
  - b. Methods
    - i. Constructor takes department, number, title, AKTS, and teacher
      - 1. Raises DepartmentMismatchException if Teacher and Course are not in the same department
    - ii. getDepartment(): Department and setDepartment(department: Department)
    - iii. getTeacher(): Teacher and setTeacher(teacher: Teacher)
      - 1. Raises DepartmentMismatchException if Teacher and Course are not in the same department
    - iv. courseCode(): String
      - 1. returns the department ID and number with a space between (i.e. "CSE 102")
    - v. toString(): String – "{department ID} {number} - {title} ({AKTS})"
- 3. Person - abstract
  - a. Attributes
    - i. Department: Department
    - ii. Other attributes – no change from previous Assignment
  - b. Methods
    - i. Constructor that takes the name, email, ID, and department
    - ii. getDepartment(): Department and setDepartment(department: Department)
- 4. Teacher – a child of Person
  - a. Attributes
    - i. No change from previous Assignment
  - b. Methods
    - i. Constructor takes name, email, ID, department, and rank
    - ii. setDepartment(department: Department) – override
      - 1. Check that Teacher is not the chair of current department
      - 2. If Teacher is the chair, sets chair of current department to null before assigning Teacher to new department
    - iii. setRank () removed as we only want to change rank through promote() and demote()
      - 1. promote() and demote() now raise InvalidRankException if the rank falls outside of valid range (which is now 1 to 4)
    - iv. getTitle(): String – returns the following based on rank value
      - 1. Lecturer
      - 2. Assistant Professor
      - 3. Associate Professor
      - 4. Professor

## 5. Student – a child of Person

### a. Attributes

- i. Remove AKTS as this will now be calculated based on courses passed.
- ii. Other attributes – no change from previous Assignment

### b. Methods

- i. Constructor that takes the name, email, ID, and department as parameters
- ii. setAKTS() and passCourse() removed
- iii. getAKTS(): int
  1. returns total AKTS of student based on passed courses
- iv. getAttemptedAKTS(): int
  1. returns total AKTS of courses the student has taken (passed and failed)
- v. addCourse(course: Course, grade: double): None
  1. Keeps a record of courses the student has taken and the grades the student has earned
  2. If course has already been taken, replaces existing grade
  3. Raises InvalidGradeException if grade is negative or greater than 100
- vi. courseGPAPoints(course: Course): double  
courseGradeLetter(course: Course): String  
courseResult(course: Course): String
  1. Returns the GPA points, grade letter, or result, respectively of the course for the student according to the following:

<u>Points</u>	<u>GPA Points</u>	<u>Grade Letter</u>	<u>Result</u>
88-100	4.0	AA	Passed
81-87	3.5	BA	Passed
74-80	3.0	BB	Passed
67-73	2.5	CB	Passed
60-66	2.0	CC	Passed
53-59	1.5	DC	Conditionally Passed
46-52	1.0	DD	Conditionally Passed
35-45	0.5	FD	Failed
0-34	0.0	FF	Failed

2. If the course has not been taken, raises CourseNotFoundException
- vii. getGPA(): double
    1. Calculates the GPA of the student based on the AKTS of each course taken and the grades earned
  - viii. toString(): String – toString() of parent + “ – GPA: “ + {GPA}

- 6. GradStudent – a child of Student
  - a. Attributes
    - i. No change from previous Assignment
  - b. Methods
    - i. Constructor that takes the name, email, ID, department, and thesis
    - ii. Use the following table to override courseGPAPoints(), courseGradeLetter(), and courseResult()

<u>Points</u>	<u>GPA Points</u>	<u>Grade Letter</u>	<u>Result</u>
90-100	4.0	AA	Passed
85-89	3.5	BA	Passed
80-84	3.0	BB	Passed
75-79	2.5	CB	Passed
70-74	2.0	CC	Passed
0-69	1.5	DC	Failed

- 7. Custom Exceptions (All must be instances of RuntimeException)
  - a. CourseNotFoundException
    - i. Additional Attributes – student: Student, course: Course
    - ii. toString() “CourseNotFoundException: “ + {student ID} + “ has not yet taken “ + {course ID}
  - b. DepartmentMismatchException
    - i. Additional Attributes – department:Department, person:Teacher, course:Course
    - ii. Two constructors:
      - 1. Takes course and person but sets department to null
      - 2. Takes department and person but sets course to null.
    - iii. toString()
      - 1. if course is null: “DepartmentMismatchException: “ + {person name} + “(“ + {person ID} + “) cannot be chair of “ + {department ID} + “ because he/she is currently assigned to “ + {person department ID}
      - 2. if course is not null: “DepartmentMismatchException: “ + {person name} + “(“ + {person ID} + “) cannot teach “ + {course code} + “ because he/she is currently assigned to “ + {person department ID}
  - c. InvalidGradeException
    - i. Additional Attribute – grade: double
    - ii. toString() “InvalidGradeException: “ + grade
  - d. InvalidRankException
    - i. Additional Attributes – rank: int
    - ii. toString() “InvalidRankException: “ + rank
  - e. An Exception you create that will be raised anytime an invalid value is attempted to be assigned to a class variable
    - i. Additional Attribute(s) – those you think are necessary
    - ii. toString() {Your exception class name}: + {class and attribute name} + {invalid value attempted} + {valid values}

**Design:** Your program does not require a main method. You are only responsible for creating the six (6) classes and four (4) Exceptions described above.

```

1 public class Assignment02_123456789 {
2     public static void main(String[] args) {
3         Department d = new Department("CSE", "Computer Engineering");
4         Teacher t = new Teacher("Joseph LEDET", "josephledet@akdeniz.edu.tr", 123L, d, 1);
5         System.out.println(t);
6
7         Course c1 = new Course(d, 101, "Programming 1", 6, t);
8         Course c2 = new Course(d, 102, "Programming 2", 4, t);
9         System.out.println(c1.courseCode() + " - " + c1.getTitle());
10        System.out.println(c2);
11
12        Student s = new Student("Test STUDENT", "me@somewhere.com", 456L, d);
13        System.out.println(s);
14
15        s.addCourse(c1, 78);
16        s.addCourse(c2, 25);
17        System.out.println(s.getAKTS());
18        System.out.println(s.getAttemptedAKTS());
19        System.out.println(s.getGPA());
20    }
21 }

```

Run IO:

```

----jGRASP exec: java Assignment02_123456789
Lecturer Joseph LEDET (123) - josephledet@akdeniz.edu.tr
CSE 101 - Programming 1
CSE 102 - Programming 2 (4)
Test STUDENT (456) - me@somewhere.com
6
10
1.8

```

**Example Exception:**

```

1 public class Assignment02_123456789 {
2     public static void main(String[] args) {
3         Department d = new Department("CSE", "Computer Engineering");
4         Department d2 = new Department("CSE2", "Computer Engineering Secrets");
5         Teacher t = new Teacher("Joseph LEDET", "josephledet@akdeniz.edu.tr", 123L, d, 1);
6         System.out.println(t);
7
8         Course c1 = new Course(d, 101, "Programming 1", 6, t);
9         Course c2 = new Course(d2, 102, "Programming 2", 4, t);
10        System.out.println(c1.courseCode() + " - " + c1.getTitle());
11        System.out.println(c2);
12
13        Student s = new Student("Test STUDENT", "me@somewhere.com", 456L, d);
14        System.out.println(s);
15
16        s.addCourse(c1, 78);
17        s.addCourse(c2, 25);
18        System.out.println(s.getAKTS());
19        System.out.println(s.getAttemptedAKTS());
20        System.out.println(s.getGPA());
21    }

```

Run IO:

```

----jGRASP exec: java Assignment02_123456789
Lecturer Joseph LEDET (123) - josephledet@akdeniz.edu.tr
Exception in thread "main" DepartmentMismatchException: Joseph LEDET(123) cannot teach CSE2 102 because he/she is currently assigned
at Course.setTeacher(Assignment02_123456789.java:18)
at Course.<init>(Assignment02_123456789.java:74)
at Assignment02_123456789.main(Assignment02_123456789.java:9)
----jGRASP wedge2: exit code for process is 1.
----jGRASP: operation complete.

```

**Code:** The file you submit will be named Assignment01\_{StudentNumber}. You should put all java classes for this assignment inside of this one (1) file as discussed in class.

**Test:** You are responsible for testing your program. It is important to not rely solely on the examples presented in this Assignment description.

## Grading:

**MS Teams Submission:** If anything is ambiguous, it is your responsibility to ask questions. It is also your responsibility to complete this assignment in a timely manner. Questions regarding this assignment will likely not be answered if received after 17:00 on the due date of the assignment.

**Quiz:** There will be a quiz based on this assignment given on 22-April. The result of this quiz will be used to determine your grade on this assignment. **Note:** if you do not take the quiz, your score on this assignment **will be 0**.

**Screenshots:** For this assignment, you must provide at minimum the six listed screenshots of your progress. An example of the screen shot for first compile is given below. An example of the screen shot for final run is shown above as the example output. These screenshots must include the entire screen (window, task bar, etc.). **Note:** if you do not submit these images, your score on this assignment **will be 0**.

