CSE 102 Programming Assignment 2

Due:

Friday, 05-April-2024 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
  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. You may create any additional classes or methods that you think necessary.

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

1. Department
   a. Attributes
      i. Code: String
         1. Must be 3 or 4 characters
      ii. Name: String
      iii. Chair: Teacher
   b. Methods
      i. Constructor that takes the code and name as a parameter. The chair will be set later as shown below.
      ii. getCode(): String and setCode(code: String)
      iii. getName(): String and setName(name: String)
      iv. getChair(): Teacher and setChair(chair: Teacher)
         1. Throws DepartmentMismatchException if Teacher is not in this department
2. Course
   a. Attributes
      i. Department: Department (replaces Department Code)
      ii. Teacher: Teacher
      iii. Other attributes – no change from previous Assignment

- b. Methods
  - i. Constructor takes department, number, title, description, AKTS, and teacher
    - 1. Throws 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. Throws DepartmentMismatchException if Teacher and Course are not in the same department
  - iv. courseCode(): String
    - 1. returns the department Code and number with no spaces between (i.e. “CSE102”)
  - v. toString(): String – “{department Code}{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 throw InvalidRankException if the rank falls outside of valid range (which is now 1 to 5)
     - iv. getTitle(): String – returns the following based on rank value
       1. Teaching Assistant
       2. Lecturer
       3. Assistant Professor
       4. Associate Professor
       5. Professor

5. Student – a child of Person
    a. Attributes
        i. AKTS 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. 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. Throws 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:

Points	GPA Points	Grade Letter	Result
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

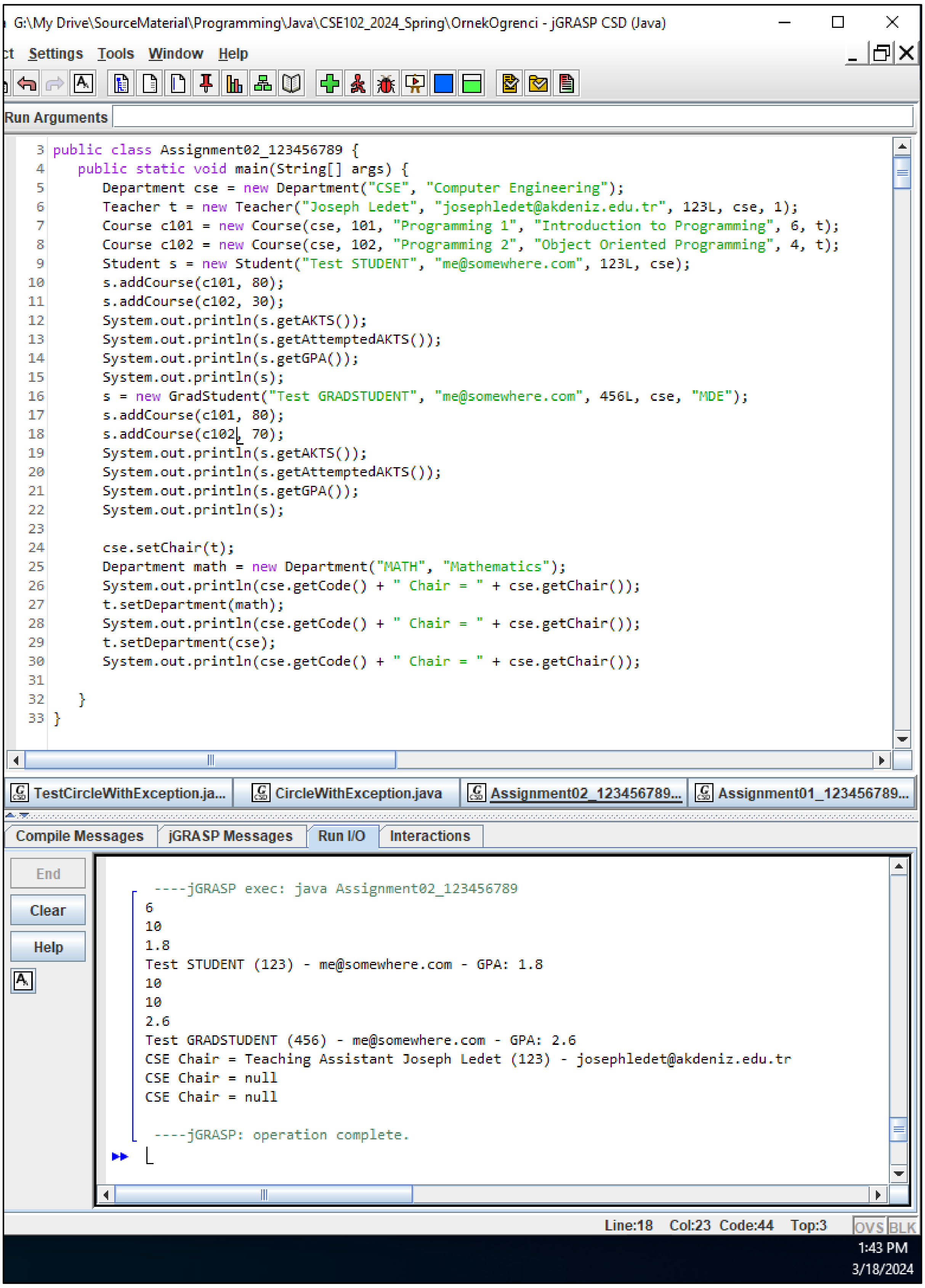
            2. If the course has not been taken, throws CourseNotFoundException
        vii. getGPA(): double
            1. Calculates the GPA of the student based on the AKTS of each course taken and the grades earned
        viii. toString(): String – toString() of parent + " – GPA: " + {GPA}
6. GradStudent – a child of Student
    a. Attributes
        i. No change from previous Assignment
    b. Methods

   i. Constructor that takes the name, email, ID, department, rank, and thesis
   ii. setRank() now throws InvalidRankException if an invalid value is passed.
   iii. Use the following table to override courseGPAPoints(), courseGradeLetter(), and courseResult()

Points	**GPA Points**	**Grade Letter**	**Result**
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	0.0	FF	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 code}
   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 thrown 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}

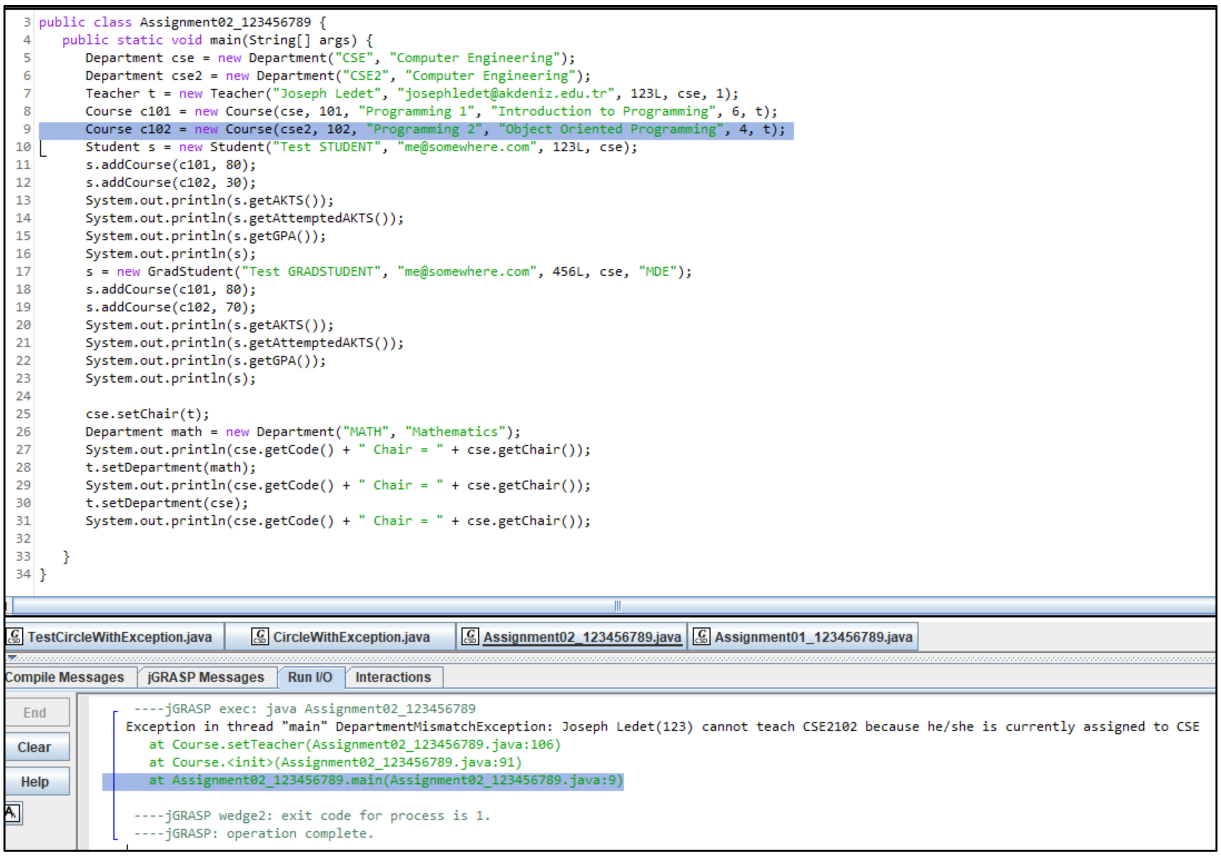
Design: Your program does not require a main method. You are only responsible for creating the six (6) classes and five (5) Exceptions described above.


G:\My Drive\SourceMaterial\Programming\Java\CSE102_2024_Spring\OrnekOgrenci - jGRASP CSD (Java)
Settings Tools Window Help
Run Arguments
TestCircleWithException.ja...
CircleWithException.java
Assignment02_123456789...
Assignment01_123456789...
Compile Messages
jGRASP Messages
Run I/O
Interactions
End
Clear
Help
Line:18 Col:23 Code:44 Top:3
OVS BLK
1:43 PM
3/18/2024

```java
public class Assignment02_123456789 {
   public static void main(String[] args) {
      Department cse = new Department("CSE", "Computer Engineering");
      Teacher t = new Teacher("Joseph Ledet", "josephledet@akdeniz.edu.tr", 123L, cse, 1);
      Course c101 = new Course(cse, 101, "Programming 1", "Introduction to Programming", 6, t);
      Course c102 = new Course(cse, 102, "Programming 2", "Object Oriented Programming", 4, t);
      Student s = new Student("Test STUDENT", "me@somewhere.com", 123L, cse);
      s.addCourse(c101, 80);
      s.addCourse(c102, 30);
      System.out.println(s.getAKTS());
      System.out.println(s.getAttemptedAKTS());
      System.out.println(s.getGPA());
      System.out.println(s);
      s = new GradStudent("Test GRADSTUDENT", "me@somewhere.com", 456L, cse, "MDE");
      s.addCourse(c101, 80);
      s.addCourse(c102, 70);
      System.out.println(s.getAKTS());
      System.out.println(s.getAttemptedAKTS());
      System.out.println(s.getGPA());
      System.out.println(s);

      cse.setChair(t);
      Department math = new Department("MATH", "Mathematics");
      System.out.println(cse.getCode() + " Chair = " + cse.getChair());
      t.setDepartment(math);
      System.out.println(cse.getCode() + " Chair = " + cse.getChair());
      t.setDepartment(cse);
      System.out.println(cse.getCode() + " Chair = " + cse.getChair());

   }
}
```

```
 ----jGRASP exec: java Assignment02_123456789
6
10
1.8
Test STUDENT (123) - me@somewhere.com - GPA: 1.8
10
10
2.6
Test GRADSTUDENT (456) - me@somewhere.com - GPA: 2.6
CSE Chair = Teaching Assistant Joseph Ledet (123) - josephledet@akdeniz.edu.tr
CSE Chair = null
CSE Chair = null

 ----jGRASP: operation complete.
```

Example Execution (above)



```
 3 public class Assignment02_123456789 {
 4    public static void main(String[] args) {
 5       Department cse = new Department("CSE", "Computer Engineering");
 6       Department cse2 = new Department("CSE2", "Computer Engineering");
 7       Teacher t = new Teacher("Joseph Ledet", "josephledet@akdeniz.edu.tr", 123L, cse, 1);
 8       Course c101 = new Course(cse, 101, "Programming 1", "Introduction to Programming", 6, t);
 9       Course c102 = new Course(cse2, 102, "Programming 2", "Object Oriented Programming", 4, t);
10       Student s = new Student("Test STUDENT", "me@somewhere.com", 123L, cse);
11       s.addCourse(c101, 80);
12       s.addCourse(c102, 30);
13       System.out.println(s.getAKTS());
14       System.out.println(s.getAttemptedAKTS());
15       System.out.println(s.getGPA());
16       System.out.println(s);
17       s = new GradStudent("Test GRADSTUDENT", "me@somewhere.com", 456L, cse, "MDE");
18       s.addCourse(c101, 80);
19       s.addCourse(c102, 70);
20       System.out.println(s.getAKTS());
21       System.out.println(s.getAttemptedAKTS());
22       System.out.println(s.getGPA());
23       System.out.println(s);
24
25       cse.setChair(t);
26       Department math = new Department("MATH", "Mathematics");
27       System.out.println(cse.getCode() + " Chair = " + cse.getChair());
28       t.setDepartment(math);
29       System.out.println(cse.getCode() + " Chair = " + cse.getChair());
30       t.setDepartment(cse);
31       System.out.println(cse.getCode() + " Chair = " + cse.getChair());
32
33    }
34 }
```

```
 ----jGRASP exec: java Assignment02_123456789
Exception in thread "main" DepartmentMismatchException: Joseph Ledet(123) cannot teach CSE2102 because he/she is currently assigned to CSE
   at Course.setTeacher(Assignment02_123456789.java:106)
   at Course.<init>(Assignment02_123456789.java:91)
   at Assignment02_123456789.main(Assignment02_123456789.java:9)

 ----jGRASP wedge2: exit code for process is 1.
 ----jGRASP: operation complete.
```

Example Exception (above)

Code: The file you submit will be named Assignment02_{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.
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 shown in Assignment 1. 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**.