COMP20220 Programming II (Conversion)

Assignment 2

1 Introduction

In this assignment, the task is to implement a basic module management system. The task involves implementing classes and various relationships between classes.

The submission deadline is 23:50 hrs, Sunday 16th April. Submit your solution via Moodle.

What to submit:

- Create a new package named `a2_12345678`, where `12345678` is your student number.
- Ensure that all your program files are included in this package. Please ensure that the version of file `Test.java` included in this package is **identical** to that downloaded from Moodle. Create a new class if you wish to carry out your own tests.
- Zip this package and upload the zip file to Moodle using the link provided under section `Assignment 2`.

Late submissions policy — when coursework is submitted late the following penalties apply:

- Coursework submitted at any time up to 1 week after the due date: −10% from mark awarded (e.g. from 60% to 50%).
- Coursework submitted more than 1 week but up to 2 weeks after the due date: −20% from mark awarded (e.g. from 60% to 40%).
- Coursework submitted after 2 weeks from the due date will not be accepted.

Important — this is **not** a team/group assignment — each student must submit her/his own work. Please ask if you have any questions about this. See the course Moodle for information on the UCD plagiarism policy.

2 What you need to do

This assignment is based on material covered in lectures. Specifically, the task involves implementing a number of classes and the relationships (i.e. association, aggregation, composition, and inheritance) between these classes to model a basic module management system.

Briefly, in this system, students take modules and instructors teach modules (association relationships). Students and instructors have names and addresses (composition and aggregation relationships). The classes to be implemented are: `Module`, `Person`, and two subclasses of `Person` — `Student` and `Instructor` (inheritance relationships). Two additional classes — `Name` and `Address` — are available for download from Moodle. These classes are described below.

Once all classes are implemented, execute the `Test` class (also available for download from Moodle). In this class, a number of name, address, student, instructor, and module objects are created; names and addresses are assigned to students and instructors; module instructors are set; students are assigned to modules etc. Refer to the `Test` class for more details and see Section 3 for details on the output that the class `Test` should produce.

Download the following classes from Moodle:

1. A class `Name` that contains:
   - `String` data fields named `firstName` and `lastName` that define the first and last names.
   - A constructor that creates a name with the specified first and last names.
   - Override the `toString()` method to return a string representation of a name. Specifically, the returned string should contain the first and last names separated by a space character.

2. A class `Address` that contains:
   - `String` data fields named `street`, `city`, and `country` that together define the address.
   - A constructor that creates an address with the specified street, city, and country.
   - Override the `toString()` method to return a string representation of an address. Specifically, the returned string should contain the street, city, and country separated by a comma.

3. A class `Test` that contains a main method to execute the code.

Implement the following classes:

1. A class `Person` that contains:
   - A `Name` data field named `name` that defines the name of the person.
   - An `Address` data field named `address` that defines the address of the person.
   - A constructor that creates a person with the specified name and address.
   - A getter method for the data field `name`.
   - Override the `toString()` method to return a string representation of a person. Specifically, the returned string should contain the person's name and address separated by a newline character.

2. A subclass of `Person` named `Student` that contains:
   - A `String` data field named `programme` that defines the programme the student is taking.
   - An array of type `Module` named `modules` that stores the modules the student is taking.
   - A constant named `MAX_NUMBER_MODULES` set to 12 that defines that maximum number of modules a student can take.
   - A constructor that creates a student with the specified name, address, and programme the student is taking.
   - A method named `addModule(Module m)` that adds a new module to the array `modules`. The method should return `true` if the new module can be added; `false` otherwise. A new module can be added provided that less than `MAX_NUMBER_MODULES` modules were added previously.
   - Override the `toString()` method to return a string representation of a student. Specifically, the returned string should contain the student's name, address, and programme separated by a newline character. Also, the code and title of each module taken by the student (if any) should be contained in the returned string. **Note**: if the student is not taking any modules, the returned string should contain the literal "none". See Section 3 for the precise format of this string.

3. A subclass of `Person` named `Instructor` that contains:
   - A `String` data field named `position` that defines the position (e.g. Lecturer, Professor etc.) of the instructor.
   - A `Module` data field named `module` that stores the module the instructor is teaching (for simplicity, it is assumed that each instructor teaches only one module).
   - A constructor that creates an instructor with the specified name, address, and position.
   - A setter method for the data field `module`.

- Override the `toString()` method to return a string representation of an instructor. Specifically, the returned string should contain the instructor's name, address, and position separated by a newline character. Also, the code and title of the module the instructor is teaching should be contained in the returned string. **Note**: if the module the instructor is teaching is not set, the returned string should contain the literal "not set" instead of the module code and title. See Section 3 for the precise format of this string.

4. A class `Module` that contains:
   - `String` data fields named `code` and `title` that define the module code and title.
   - An `Instructor` data field named `instructor` that stores the instructor teaching the module.
   - An array of type `Student` named `students` that stores the students taking the module.
   - A constant named `MAX_NUMBER_STUDENTS` set to 80 that defines that maximum number of students that can take a module.
   - A constructor that creates a module with the specified code and title.
   - Getter methods for the `code`, `title`, and `instructor` data fields.
   - A setter method for the data field `instructor`.
   - A method named `addStudent(Student s)` that adds a new student to the array `students`. The method should return `true` if the new student can be added; `false` otherwise. A new student can be added provided that less than `MAX_NUMBER_STUDENTS` students were added previously.
   - Override the `toString()` method to return a string representation of a module. Specifically, the returned string should contain the code, title, and the instructor's name separated by a newline character. **Note**: if the instructor teaching the module is not set, the returned string should contain the literal "not set" instead of the instructor's name. Also, the name of each student taking the module (if any) should be contained in the returned string. If there are no students taking the module, the returned string should contain the literal "none". See Section 3 for the precise format of this string.

3 Executing the Code

Once all classes are implemented, execute the class `Test`. The output of this class should appear **exactly** as below (if not, marks will be deducted!).

```
Students
========
name: Charles Darwin
address: 20 Lower Rathmines Road, Dublin, Ireland
programme: Computer Science
modules:
        CS-0010 (Data Science)

name: Marie Curie
address: 20 Lower Rathmines Road, Dublin, Ireland
programme: Computer Science
modules:
        CS-0010 (Data Science)

name: Ada Lovelace
address: 111 Clonskeagh Road, Dublin, Ireland
programme: Computer Science
modules:
        CS-0010 (Data Science)
        CS-0020 (Artificial Intelligence)

name: Alan Turing
address: 111 Clonskeagh Road, Dublin, Ireland
programme: Computer Science
modules:
        CS-0010 (Data Science)
        CS-0020 (Artificial Intelligence)

name: Bob Smith
address: 17 Woodlands Road, Dublin, Ireland
programme: Computer Science
modules:
        none

Instructors
===========
name: Albert Einstein
address: 222 Rathgar Road, Dublin, Ireland
position: Professor
module: CS-0010 (Data Science)

name: Grace Hopper
address: 222 Rathgar Road, Dublin, Ireland
position: Professor
module: not set
```

```
Modules
=======
code: CS-0010
title: Data Science
instructor: Albert Einstein
students:
        Charles Darwin
        Marie Curie
        Ada Lovelace
        Alan Turing

code: CS-0020
title: Artificial Intelligence
instructor: not set
students:
        Ada Lovelace
        Alan Turing

code: CS-0030
title: Mathematics 101
instructor: not set
students:
        none
```