LONDON METROPOLITAN UNIVERSITY

FIRST SIT FIRST COURSEWORK QUESTION PAPER: Year Long Spring 2020

Module Code:	**CS4001NI**
Module Title:	**Programming**
Module Leader:	**Sushil Paudel** (Informatics College Pokhara)

Coursework Type:	**Individual**
Coursework Weight:	This coursework accounts for **30%** of your total module grades.
Submission Date:	**12th Week**
When Coursework is given out:	**9th Week**
Submission Instructions:	Submit the following to Informatics College Pokhara RTE department before the due date: ● **A report in PDF format and zip file which includes program file** ● **File should be in .java format**
Warning:	London Metropolitan University and Informatics College Pokhara takes Plagiarism seriously. Offenders will be dealt with sternly.

Plagiarism Notice

You are reminded that there exist regulations concerning plagiarism.

Extracts from University Regulations on Cheating, Plagiarism and Collusion

Section 2.3: “The following broad types of offence can be identified and are provided as indicative examples …..

(i) Cheating: including copying coursework.
(ii) Falsifying data in experimental results.
(iii) Personation, where a substitute takes an examination or test on behalf of the candidate. Both candidate and substitute may be guilty of an offence under these Regulations.
(iv) Bribery or attempted bribery of a person thought to have some influence on the candidate’s assessment.
(v) Collusion to present joint work as the work solely of one individual.
(vi) Plagiarism, where the work or ideas of another are presented as the candidate’s own.
(vii) Other conduct calculated to secure an advantage on assessment.
(viii) Assisting in any of the above.

Some notes on what this means for students:

(i) Copying another student's work is an offence, whether from a copy on paper or from a computer file, and in whatever form the intellectual property being copied takes, including text, mathematical notation and computer programs.
(ii) Taking extracts from published sources without attribution is an offence. To quote ideas, sometimes using extracts, is generally to be encouraged. Quoting ideas is achieved by stating an author's argument and attributing it, perhaps by quoting, immediately in the text, his or her name and year of publication, e.g. " e = mc2 (Einstein 1905)". A reference section at the end of your work should then list all such references in alphabetical order of authors' surnames. (There are variations on this referencing system which your tutors may prefer you to use.) If you wish to quote a paragraph or so from published work then indent the quotation on both left and right margins, using an italic font where practicable, and introduce the quotation with an attribution.

Further information in relation to the existing London Metropolitan University regulations concerning plagiarism can be obtained from http://www.londonmet.ac.uk/academic-regulations

Assessment

This assignment will be marked out of 100 and carries 30% of the overall module weighting.
Your .java files and report for this part must be uploaded and submitted on Week 12th. The assignment must be carried out individually so you must not obtain help from anyone other than the module teaching staff. You must not copy code from any source apart from the module core text and the module materials. Collusion, plagiarism (unreferenced copying) and other forms of cheating constitute Academic Misconduct, which can lead to failure of the module and suspension.
The viva will be conducted for this assignment.

Note: *If student would be unable to defend his/her coursework, s/he might be penalized with 50% of total coursework marks*

Aim

The aim of this assignment is to implement a real world problem scenario using Object-oriented concept of Java that includes creating a class to represent a **Course**, together with its two subclasses to represent an **Academic course** and a **Non-academic Course** respectively. You will also need to write a report that should contain about your program.

Deliverables

Create a new project in **BlueJ** and create three new classes (**Course**, **Academic Course** and **NonAcademicCourse**) within the project. AcademicCourse and NonAcademicCourse are subclasses of the class Course. When you are ready to submit your solution, upload your codes ***Course.java,*** **AcademicCourse.java** and **NonacademicCourse.java** files (not any other files from the project) together with your report in pdf format.

Program (60 marks)

The program should include the following classes (with no additional attributes or methods).

1) The **Course** class has four attributes, which correspond to the courseID, course name, course leader and duration. The courseID and coursename are each represented as a string of text and duration as a number.
   The courseID, coursename and duration are initialized in the constructor by being assigned the value of the constructor's parameters. The courseLeader is initialized with empty string (“”) in the constructor.

   Each attribute has a corresponding accessor method.

   Define a method to set the courseLeader to a new name by accepting the new name of the course leader as a parameter.

   A display method should output (suitably annotated) the courseID, coursename, and duration and, if the courseLeader is not an empty string, the name of the courseLeader also should be displayed.

   [10 marks]

2) The **AcademicCourse** class is a subclass of the **Course** class and has seven attributes:

   Lecturer name - a string of characters
   Level - a string of characters
   Credit - a string of characters
   Starting Date - a string of characters
   Completion Date - a string of characters
   Number of Assessments- a whole number
   isRegistered - either true or false (boolean)

   The **constructor** accepts six parameters, which are the courseID, coursename, duration, level, credit, and number of assessments. A call is made to the superclass constructor with three parameters, the courseID, coursename, and duration. The level, credit and the number of assessments are initialized by being assigned the value of constructor’s parameters. The Additionally, in the

constructor, assign lecturer name, starting date and completion date as an empty ("") string, isRegistered status is initialized to false.

Each attribute has a corresponding **accessor method.**

A method is required to **set the lecturer name** as changes to lecturers inevitably occur. The method accepts a new lecturer name as a parameter and assigns the new value to the lecturer name.

A method is required to **set the number of assessments** as changes to the number of assessments also occur. The method accepts a new number of assessments as a parameter and assigns the new value to the number of assessments attribute.

There is a method to **register** the particular academic course. The method accepts four parameters that includes a course leader name, lecturer name, starting date, and completion date when the course is tentatively planned to complete. If the academic course is already registered, an appropriate message including the lecturer name, starting date and completion date is displayed.
If the academic course has not been registered yet, the method to set the course leader (course leader name in parent class) is called from the parent class with the course leader name as a parameter. The lecturer name, starting date, and completion date are required to update by the parameter values input to the method, the registered status of the course is changed to true.

A method to **display** the details of the course is required. It must have the same signature as the display method in the course class. It will call the method in Course class to display the courseID, course name, duration, and course leader. If the academic course is already registered then lecturer name, level, credit, starting date, completion date and number of assessments must also be displayed. Each output must be suitably annotated.

[20 marks]

3) The **NonAcademicCourse** class is also a subclass of the **Course** class and it has seven attributes:

Instructor name	- a string of character
Start date	- a string of character
Completion date	- a string of character
Exam date	- a string of character

prerequisite	- a string of character
isRegistered	- either true or false (boolean)
isRemoved	- either true or false (boolean)

The **constructor** accepts four parameters which are the courseID, course name, duration and prerequisite. A call is made to the superclass constructor with three parameters, the courseID, course name and duration. The attribute prerequisite is given the corresponding parameter value. Additionally, the constructor initializes starting date, completion date and exam date to empty ("") string, and isRegistered and isRemoved status to false.

Each attribute has a corresponding **accessor method.**

A method is required to **set the instructor name** as changes to name of the instructor inevitably occur. The method accepts a new instructor name as a parameter and, if the non-academic course has not registered yet, the new value is assigned to the instructor name attribute. If the non-academic course is already registered, then a suitable message is output to the user indicating that it is therefore not possible to change the instructor name.

There is a method **register** for registering the non-academic course. The method has five parameters, the course leader name, instructor name, starting date, completion date, and exam date. If the non-academic course has not registered yet, the method to set the instructor name is called with the instructor name as a parameter, the method to set the course leader (course leader name in parent class) is called with the course leader name as a parameter, the starting date, completion date and exam date are updated by the corresponding parameter's values and isRegistered status is set to true, otherwise a suitable message is displayed to the user indicating that the course is already registered.

There should be a **remove** method for non-academic courses. If the non-academic course is already removed (by checking isRemoved status is true), a suitable message is output. If the course is not removed yet, the method to set the course leader name (method to set course leader name in parent class) is called with "" as a parameter, the instructor name, the starting date, completion date and exam date are set to "", the status of isRegistered is set to false and the isRemoved status is set to true.

A method to **display** the details of the non-academic course is required. It must have the same signature as the display method in the Course class. It will call

the method in Course class to display the courseID, course name and duration. If the non-academic course is already registered then instructor name, starting date, completion date and exam date must also be displayed. Each output must be suitably annotated. **[20 marks]**

Additional marks will be awarded for good programming styles, particularly naming, layout and comments.
See http://www.bluej.org/objects-first/styleguide.html for details.

[10 marks]

Report (40 marks)

Your report should describe the process of development of your classes with:

a. A class diagram **[5 marks]**

b. Pseudocode for each method in each class **[10 marks]**

c. A short description of what each method does **[5 marks]**

d. You should give evidence (through inspection tables and appropriate screenshots) of the following testing that you carried out on your program:
Test 1: Inspect AcademicCourse class, register an academic course, and re-inspect the AcademicCourse Class **[2 marks]**
Test 2: Inspect NonAcademicCourse class, register a non-academic course and re-inspect the NonAcademicCourse Class **[2 marks]**
Test 3: Inspect NonAcademicCourse class again, remove the non-academic course and re-inspect the NonAcademicCourse class **[2 marks]**
Test 4: Display the detail of AcademicCourse and NonAcademicCourse classes. **[4 marks]**

e. The report should contain a section on error detection and error correction where you give examples and evidence of three errors encountered in your implementation. The errors (syntax, semantic or logical errors) should be distinctive and not of the same type. **[3 marks]**

f. The report should contain a conclusion, where you need to include the following things:
   - Evaluation of your work,
   - Reflection on what you learnt from the assignment,

- What difficulties you encountered and
- How you overcame the difficulties.

[4 marks]

The report should include a title page (including your name and ID number), a table of contents (with page numbers), an introduction part that contains a brief about your work and a listing of the code (in an appendix). Marks will also be awarded for the quality of writing and the presentation of the report.

[3 marks]

Viva

Note: ***If a student would be unable to defend his/her coursework, s/he might be penalized with 50% of total coursework marks.***

Marking Scheme

Marking criteria		Marks
A.	**Coding Part**	**60 Marks**
	1. Creating Course Class 2. Creating Academic Class 3. Creating Nonacademic Class 4. Program Style	10 Marks 20 Marks 20 Marks 10 Marks
B.	**Report Structure and Format**	**40 Marks**
	1. Class Diagram 2. Pseudocode 3. Method Description 4. Test-1 5. Test-2 6. Test-3 7. Test-4 8. Error Detection and Correction 9. Conclusion 10. Overall Report Presentation/Formatting	5 Marks 10 Marks 5 Marks 2 Marks 2 Marks 2 Marks 4 Marks 3 Marks 4 Marks 3 Marks
	Total	**100 Marks**