LONDON METROPOLITAN UNIVERSITY

1st SIT COURSEWORK QUESTION PAPER: Year Long 2021

Module Code:	**CS4001NP**
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:	**Week 24**
When Coursework is given out:	**Week 20**
Submission Instructions:	Submit the following to Informatics College Pokhara RTE department before the due date **A Report in PDF format and a zip file which includes a BlueJ Project File**
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 by on Friday of Week 24. 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

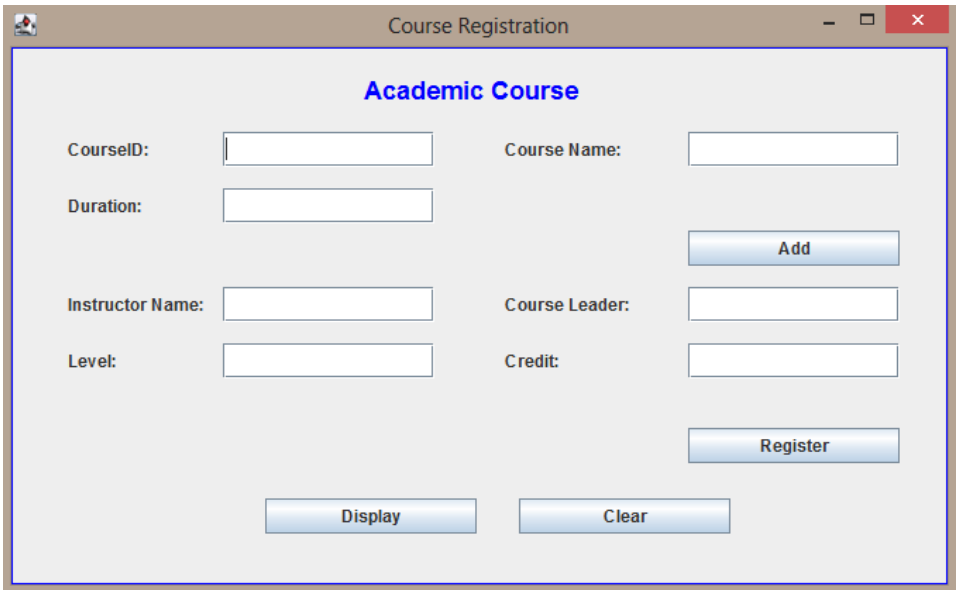
The aim of this assignment is to add a class to the project that you developed for the first part of the coursework to make a graphical user interface (GUI) for a system that stores details of **Course** that includes both **academic** and **non-academic course**. The class will contain a main method and will be tested using the command prompt. You will also need to write a report about your program.

Deliverables

Create a new class within the project called **INGCollege**. When you are ready to submit your solution, upload your **INGCollege.java** file, together with the **Course.java**, **AcademicCourse.java** and **NonAcademicCourse.java** files from the first part of the coursework (not any other files from the project) together with your report .pdf format.

Program (60 marks)

A sample form of GUI is shown below:



1. Your GUI should contain the same components as mentioned, but you are free to use a different layout if you feel that it improves the aesthetics, ease of use etc. The **INGCollege** class should store an array list (not an array) of type **Course** to hold the **AcademicCourse** and **NonAcademicCourse**. There should be text fields for entering:

i. CourseID
ii. Course name
iii. Duration
iv. Course leader
v. Lecturer name
vi. Level
vii. Credit
viii. Start date
ix. Completion date
x. Number of assessments
xi. Instructor name
xii. Exam date
xiii. Prerequisites

2. The GUI should have the following buttons

i. **Add course for Academic Course**

When this button is pressed, the input values of the courseID, course name, duration, level, credit, and number of assessments are used to create a new object of type *AcademicCourse* which is added to an array list of *Course* class.

ii. **Add course for Non-academic Course**

When this button is pressed, the input values of the courseID, course name, duration, and prerequisite are used to create a new object of type *NonAcademicCourse* which is added to an array list of *Course* class.

iii. **Register Academic Course**

The courseID, course leader, lecturer name, starting date, and completion date are entered in the GUI. When the button is pressed, the input value of courseID is compared to the existing CourseID, and if valid courseID has been entered, it is used to register the academic course from the list. The method to register academic course from the AcademicCourse class is called here.

Hint: *An object of Course is cast as AcademicCourse type.*

iv. **Register Non-academic Course**

The courseID, course leader, instructor name, start date, completion date, exam date are entered in the GUI. When the button is pressed, the input value of courseID is compared to the existing CourseID, and if valid courseID has been entered, it is used to register the appropriate non-academic course from the list. The method to register the non-academic course from the NonAcademicCourse class is called here.

Hint: *An object of Course is cast as NonAcademicCourse type.*

v. **Remove Non-academic Course**

The courseID is entered in the GUI. When the button is pressed, the input value of the courseID is compared to the existing courseID in the list. If a valid value has been entered, it is used to remove the specified non-academic course from the array list of Course. The method to remove non-academic course from the NonAcademicCourse class is called here.

Hint: *An object of Course is cast as NonAcademicCourse type.*

vi. **Display**

When this button is pressed, the information relating to the appropriate class is displayed.

vii. **Clear**

When this button is pressed, the values from text fields are cleared.

Additional Information:

Write methods to return the values of each of the text fields using the getText() method. For the duration, credit and number of assessments, get the text from text field, convert it to a whole number and return the whole number.

Additionally, use try & catch blocks to catch any Number Format Exception that might be thrown in converting the string to an integer or double. If the text input is incorrect in any way and output a suitable error message in a message dialog box.

Marks will be awarded as follows:

i. GUI and main method **[12 marks]**
ii. Functionality of Buttons **[26 marks]**
iii. Reading input, checking input and displaying appropriate messages **[12 marks]**
iv. Programming Style (https://www.bluej.org/objects-first/styleguide.html) **[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 appropriate screenshots) of the following testing that you carried out on your program:

Test 1: Test that the program can be compiled and run using the command prompt, including a screenshot like Figure 1 from the command prompt.
[2 marks]

Test 2: Evidences should be shown of:

- **a.** Add course for Academic course
- **b.** Add course for Non-academic course
- **c.** Register academic course
- **d.** Register non-academic course
- **e.** Remove non-academic course

[5 marks]

Test 3: Test that appropriate dialog boxes appear when:

- **a.** Trying to add duplicate courseID
- **b.** Trying to register already registered course
- **c.** Trying to remove the non-academic course which is already removed.

Include a screenshot of the dialog box, together with a corresponding screenshot of the GUI, showing the values that were entered.
[3 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 and 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 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. GUI and main method 2. Functionality of Buttons 3. Reading input, checking input and displaying appropriate messages 4. Program Style	12 Marks 26 Marks 12 Marks 10 Marks
B.	**Report Structure and Format**	**40 Marks**
	1. Class Diagram 2. Pseudocode 3. Method Description 4. Test-1(Compiling & Running using command prompt) 5. Test-2(Adding a courses for Academic course and Non-academic course, registering them and removal of course) 6. Test-3(Testing Appropriate Dialog boxes when unsuitable values entered) 7. Error Detection and Correction 8. Conclusion 9. Overall Report Presentation/Formatting	5 Marks 10 Marks 5 Marks 2 Marks 5 Marks 3 Marks 3 Marks 4 Marks 3 Marks
	Total	**100 Marks**