ELECTRONIC, ELECTRICAL AND SYSTEMS ENGINEERING

PER AD ARDUA ALTA

MSc Resit Coursework
Introductory Module for Computing

2017-18

Python Programming: Net Salary Calculator (GUI)
UML: Student Record System

Assignment Format:

This coursework consists of two parts:

1. Python Programming and
2. UML.

You are expected to submit your Python program for the first part and only a report for the second part. **The Python code and the PDF UML report must be zipped together and submitted via canvas unless stated otherwise**.

The two parts will be marked separately and have equal weightage of 50% each.

PART 1: Python Programming

1. Aims and Objectives

This Python assignment is about implementing a graphical user interface to create a *NetSalaryCalculator*. It can calculate the net salary of an employee after the deductions of tax, pension and national insurance has been made. The tax rates that this calculator applies are shown in the table below:

Band	**Taxable income (yearly)**	**Tax rate**
Personal Allowance	Up to £11,850	0%
Basic rate	£11,851 to £46,350	20%
Higher rate	£46,351 to £150,000	40%
Additional rate	over £150,000	45%

The pension contribution as a percentage of gross salary can be chosen by the user. The user can opt out of pension contribution or contribute 100% towards pension.

The National insurance contributions are made according to the table below:

Salary up to £8400 a year	**£8400.01 to £46,350 a year**	**Over £46,350 a year**
0%	12%	2%

It has the following functions:

a) Accept **annual** or **monthly** gross salary from the user.
b) Accept percent of the salary to be contributed towards the pension.
c) Calculate the amount of pension contribution.
d) Evaluate the taxable income.
e) Calculate the amount of tax to be paid.
f) Calculate the amount of National Insurance to be paid.
g) Calculate the net salary after all deductions.

2. Lab Work

Using the **object oriented approach**, implement the *NetSalaryCalculator* as a Python application. The application should be able to create a GUI containing all widgets required to run all the functionality. The *NetSalaryCalculator* should at the least have the following:

1. It should allow the user to enter his/her annual or monthly gross salary.
2. An entry field to enter the gross income.
3. Annual and monthly as radio buttons to choose between the two.

4. An entry field to enter the percent of salary towards pension.
5. Display the taxable income
6. Display the amount of tax deducted.
7. Display the amount of pension contribution.
8. Display the amount of National insurance contribution.
9. Display the net salary after tax, pension and national insurance deductions.

After creating your GUI with all the above widgets, you will then have to bind these to the functions to get your application working.

Think carefully about how many digits your *NetSalaryCalculater* can display. Handle any exceptions that may arise during the calculations

3. Assessment

Marks distribution will be as follows:

a) It can allow the user to enter the gross salary correctly. (5%)
b) It can allow the user to enter the pension to be paid as percentage of salary correctly. (5%)
c) It can calculate the taxable income. (5%)
d) It can calculate and display the amount of tax, pension and the national insurance to be paid. (15%)
e) It can calculate and display the net salary after all deductions. (10%)
f) Annual or monthly view of (a) – (e) available using radio buttons. (10%)
g) Exception handling where required. (10%)
h) Code documentation. (15%)
i) GUI design. (25%)

PART 2: OBJECT ORIENTED DESIGN (UML)

1. Specification

Student Record System

You are required to design object-oriented software for a student record system. This system should record the outcome of personal tutor meetings, examination marks, tutorial results and laboratory attendance. Students should be able to access the system via user name and password to view information of themselves only and to book an appointment with their personal tutor. Academic Staff should be able to view the records of their tutees, send messages to their tutees and record a textual summary of tutorial meetings. Students are required to enter personal details such as current home address, contact telephone number (term time and out of term time), hobby and other interests and brief details of non-University qualifications. It should also be possible to scan doctor's notes and correspondence into the record system. Access by personal tutors requires the entry of a user ID and password. In addition Year Tutors should be able to access the records for a year and the Head of Taught Programmes should be able to access any record or group of records, using a user ID and password. A nominated administrator is the only person able to add student names to create a record and able to delete entries. The administrator has a unique user ID and password. The Head of Taught Programmes, Year Tutors and the Administrator should be able to search the record system by student name, module, number of failed credits and a wide range of similar options. This means that these users should be able to create queries in the form of a database query language. Consider any additional requirements and explain why they may be needed and incorporate them into your design.

2. Assessment

Assessment is based on a written report. Clearly the report must fully address all relevant issues. The assignment is open-ended and it is possible that you will not complete the design but it is important that all stages of the design process are considered to a reasonable level of detail.

3. Presentation of the report

You are required to submit a written report. The report should be a single pdf file. Use reasonable margins, e.g. 2 cm left right, top and bottom, and a reasonable font size (e.g. 10 pt Arial or 12 pt Times Roman). The figures, tables and pages should be numbered. Captions should be beneath figures and above tables.

As you will not be working in a group you may wish to brain-storm user requirements with friends or family but remember the work submitted must be your own and you should not collaborate with anyone after an initial brain storm session.

The report should contain:

Front cover: includes the statement of originality that the work is your own work

Contents List: with page numbers

Introduction: This is the discussion and interpretation of the specification, which should identify any issues that require clarification.

Use-Case Model: This is a formal statement of user requirements with Use-Case views, survey descriptions, simple interaction, state-chart, class diagrams and CRC cards. Remember the diagrams should be in outline form at this stage.

Analysis Model: This is a technical specification. The class descriptions should be detailed with attributes and methods and relationships to define architecture. Detail should be added to the interaction and state-chart diagrams to show events, actions and qualifiers. Non-function requirements should be identified. The analysis model should identify service and analysis packages as appropriate.

Design Model: This should include detailed design classes in which all methods and attributes and their visibility is identified. Arguments and return data types should be identified and defined. Key implementation mechanisms should be defined. Interaction diagrams, state-charts should also be detailed. The mechanisms for the implementation of non-functional requirements should be defined.

You are not required to provide Implementation and Test Models.

4. Resources available

This assignment involves the development of designs on paper. It should be possible to draw good quality design diagrams using Word or other word processing package, adopting the correct notation. Hand drawings will not be accepted. There is no need to use any Object-Oriented Design Tools. This is because learning curve associated with object-oriented design tools is normally steep. Many design consultants use graphical design programs and not object-oriented design tools to document designs. Two tools that you might find useful are Visio (might need licence) and ArgoUML (http://argouml.tigris.org/).

5. Submission

The complete report (along with the Python Code for PART 1 zipped together) should be submitted by the deadline provided.

Marks distribution will be as follows:

Presentation: 5%

Structure, clarity and presentation of report. Contents page, page numbering, presence of correct front cover, labelling of diagrams.

Introduction and conclusion: 5%

Interpretation of specification. Identification of appropriate elaboration.

Use- Case Model: 40%

Correct use of notation. Complete set of diagrams. Appropriateness of use-case structure, survey descriptions, CRC cards, interaction diagram(s), class diagram, class identification and statechart diagrams. Remember these are outline diagrams.

Analysis Model: 25%

Appropriateness of elaboration. Identification of methods and attributes. Definition of class architecture. Identification of non-functional requirements. Consideration and appropriateness of package diagrams.

Design Model: 20%

Appropriateness of elaboration. The visibility of attributes and methods. The identification of arguments, return data types, class creation and destruction, process forks and joins as appropriate. Definition non-functional requirement implementation

Discretionary Marks: 5%

Evidence of special insight.

A word of caution ⚠

"Plagiarism will not be tolerated. It is the act of a Student claiming as their own, intentionally or by omission, work which was not done by that Student. Plagiarism also includes a Student deliberately claiming to have done work submitted by the Student for assessment which was never undertaken by that Student, including self-plagiarism and the other breaches. Sanctions of a plagiarism include the Student failing the Programme of study".