Assignment 5

Deadline:	Hand in by ***midnight Wed, 28th January 2015*** and demonstrate at the following Lab session (29th January 2015, 9 – 11 AM). The assignment is to be submitted via Stream. ***NOTE: the lab session has been moved to Thursday because the preceding Monday 26th January 2015 is a public holiday.***
Evaluation:	8 marks – which is 8% of your final grade.
Late Submission*:*	*1 mark deduction per day late*
Marking and Penalty	The assignment will be marked face-to-face with a tutor *Failure to attend the face-to-face marking session with your tutor (without prior alternative arrangement) will result in your marks being reduced by 50% (out of the possible maximum of 8 marks).*
Work	This assignment is to be done **individually** – your submission may be checked for plagiarism against other assignments and against Internet repositories. If you adapt material from the Internet acknowledge your source.
Purpose:	To learn to develop software using multiple windows forms in an object-oriented architectural style that is testable as per the TDD approach, and to apply appropriate data structure and file input/output mechanisms.

Problem to solve:

Design a student university-enrolment application

Functional Requirements:

Create an application to act as a University student management system. You should create classes to model a university, students and papers. The system should store the following information about each student – name, birthdate, ID number, an address, and a collection of papers in which the student is enrolled. The system should also store the following information about each paper – name, paper code, course coordinator and a list of students enrolled in the paper. All data members should be private and any properties should be read only.

The application should also have the functionality to add a new student to the university, add a new paper to the university, and enrol a student in a paper. A separate form should be used to perform each of these functions.

As output, the application should be able to display a list of all the students enrolled in the university, to display a list of all the papers taught, to allow the user to select a student and display all the papers that student is enrolled in, and to allow the user to select a paper and display all of the students enrolled in it.

Your application should also have the functionality to read and write to a file all the data that is in memory. A user should be able to upload all the saved data into the application and append the data if necessary to what is already in the application memory. If data is being appended, then your application must verify that duplicate entries are not loaded into the application.

Marking Criteria:

Functionality (3.5%)

- Proper display of the list of all students enrolled in the university and the list of all the papers taught in the university (1.5%)
- Use of appropriate form control(s) for the selection of an enrolled student and, subsequently, a proper display of all the papers that the student is enrolled in (1%), and
- Use of appropriate form control(s) for the selection of a paper taught in the university and, subsequently, a proper display of all students enrolled in the paper selected (1%).

Programming Approach (3.5%)

- Appropriate decoupling of the presentation layer from the business logics such that the business logics. In particular, the application should be designed using the following architectural approach (1.5%):
    - the use of a Class Library Project to develop classes that encapsulate the necessary classes, including the necessary business logics, and
    - a Windows Forms Project that creates the UI and uses the class library project containing all the business logic by referencing it.

- The use of appropriate collection classes (1%)
- Testable application as per the TDD approach. You need not write unit tests for event-handlers but you must write at least one unit test for each of your user-defined methods/properties within your classes (1%).

Good Programming Practice (1%)

- Variables and function names must be named appropriately.
- Looping and conditional statements must follow conventional and best-practice principles.
- The application must be broken down into subroutines/functions that perform one task each.
- All code duplication must be eliminated**.**
- Proper handling of possible errors in the application for all input and operations in the application
- The inclusion of refactoring in the application as necessary.

Hand-in: Submit your **program** via **stream** in a **zip** file (**ONLY** if the submission site is not available email to s.suriadi@massey.ac.nz). Make sure you zip your **entire** solution folder so that the .sln as well as the relevant project files are included.

If you have any questions or concerns about this assignment, please ask the lecturer well in advance.