Module Code:	**CET212**
Module Title:	**Advanced Software Development**
Module Leader:	**Chris Knowles**
Assessment Number:	**2 of 2**
Assessment Title:	**Design and Implementation of Software Prototype**

The following learning outcomes will be assessed:

	Knowledge	
1.	**Use of advanced OO software design, development and testing techniques and practices.**	
2.	**Understanding of issues associated with the modern software development life cycle.**	✓
	Skills	
3.	**Application of advanced OO techniques and practices to the design, development and testing of professional-standard software.**	✓
4.	**Application of standard industrial methods and practices to the development of a multi-threaded distributed software artefact.**	✓

Deadline for submission of deliverables:

Submission Date and Time	**5:00PM, FRIDAY 22nd MAY 2020**
Submission Location	**Submission via Canvas Assignment Page**

Important Information

You are required to submit your work within the bounds of the University Infringement of Assessment Regulations (see the Programme Handbook). Plagiarism, paraphrasing and downloading large amounts of information from external sources, will not be tolerated and will be dealt with severely. Although you should make full use of any source material, which would normally be an occasional sentence and/or paragraph (referenced) followed by your own critical analysis/evaluation. You will receive no marks for work that is not your own. Your work may be subject to checks for originality which can include use of an electronic plagiarism detection service. Where you are asked to submit an individual piece of work, the work must be entirely your own. The safety of your assessments is your responsibility. You must not permit another student access to your work. Where referencing is required, unless otherwise stated, the Harvard referencing system must be used (see the Programme Handbook).

This contributes 70% to your final module mark and assesses learning outcomes 2, 3 & 4

Scenario

You are working as a consultant to a local authority IT department. You have been tasked with developing a C# system to manage air particulate sensor records (in micrograms per cubic metre, µg/m^3) for various locations across the authority. Each set of readings for a single location will be submitted as an XML data file according to a standardised structure (see later), containing location and reading dates together with the noon particulate reading (µg/m^3), temperature (in degree Celsius) and relative humidity (as a %) for each date at the location. Your system will be required to read and process the XML files for all locations of interest in order to determine total particulates for a given location, total particulates across all locations on each date and to find out which location has the largest individual particulates reading. A significant number of locations will be submitted and it is important that the authority is able to process the data quickly so that they can identify particulate hot-spots when necessary. Therefore, a multi-threaded system that can process multiple files concurrently will be required.

Ultimately, the intention is for the system to be made available as a client-server application but your task has been simplified. You are required to develop a prototype application that will run on a standalone PC but you should make use of separation of concerns so that the job of transferring the application to the local authority's client-server environment in the future will be possible with minimum adjustments to your code.

An example XML file showing the structure of the data is provided below:

```
<?xml version="1.0" encoding="utf-8" ?>
<Particulates>
  <Location name="Quayside">
    <Reading date="18/02/2020">
      <value>13</value>
      <temperature>8.0</temperature>
      <humidity>51.5</humidity>
    </Reading>
    <Reading date="19/02/2020">
      <value>21</value>
      <temperature>11.4</temperature>
      <humidity>55.0</humidity>
    </Reading>
  </Location>
</Particulates>
```

This XML file represents the data for a location called *Quayside*, with a reading taken *18th February 2020* of *13 µg/m^3* with *temperature of 8.0^{o}C* and *Humidity of 51.5%* plus another reading *19th February 2020* of *21 µg/m^3* with *temperature of 11.4^{o}C* and *Humidity of 55.0%*.

The data for each location will be stored in a separate file and each location may have any number of date readings recorded, which will be on different dates in the same file.

You have been tasked with designing this application as a multi-threaded windows application, making use of the Producer/Consumer pattern and LINQ query language. You have been given the following list of functional requirements for the system:

- Select and queue the processing of multiple XML files.
- Merge the data from multiple XML files and carry out the necessary processing in order to generate the following output reports (via display on a Windows Form):

- A list of locations, with the ability to select and view details of all readings for a selected location (including the date, particulates value, temperature and humidity readings)
- A list of locations, sorted alphabetically, with the total particulates across all date readings for that location
- A list of dates, with the total recorded particulates values (across all locations), sorted by date
- The largest individual particulates value recorded by the system (the location, date and highest recorded particulates value across all amalgamated files)

Design & Reporting Task

Write a report, in either Word or PDF format, which contains the following sections:-

UML Class Diagram (15 Marks)

Propose an object oriented model to solve the specified problem and document this as a UML class diagram. You should include

- data types of instance variables, parameters and method return values (when non-void)
- visibility modifiers
- parameterised constructors (where appropriate)
- relationships such as implementation and dependency.

Hand drawn UML diagrams are not acceptable. You may use a UML drawing tool e.g. *Software Ideas Modeller* or *draw.io* for this task but if you do so then the UML notation used must be consistent with that covered in the module and you should export the diagram to an image file which can be embedded within the report document.

You have been provided with an incomplete UML class diagram in the file *CET212-Assessment2-UML-Incomplete.docx* and you are to use this when producing your own design. You will needed to reproduce the class and relationships on the UML class diagram and then add in classes and relationships for the *data related classes*, the *XML file reader class* and the *main GUI form class*.

Design Rationale (10 marks)

A 200 word explanation of how the separation of concerns is evident in your final design.

Implementation and Testing Task

C# Application (55 marks)

Implement the system described in the scenario above as a C# forms application using Visual Studio (version VS2019). You should apply ALL the coding conventions used in the module including naming, layout and commenting: Marks will be explicitly awarded on this and poor presentation may adversely affect marks for other aspects if it obscures the correctness of otherwise satisfactory code.

Your application should be multi-threaded, making use of the Producer/Consumer pattern and should be make use of Windows Forms, designed with an interface suitable for users with limited computing experience. Your program should demonstrate the use of multiple producers and consumers, with an appropriately sized queue.

You should include any example XML files (placed in the bin/debug folder of your project) used in the testing of your application.

Appropriate exception/error handling should be included to ensure that the application is robust.

You have been provided an example C# application in the archive file *CET212-Election-Example.zip* and you are strongly recommended to utilise the code from this example when producing your own implementation. You are permitted to include the C# classes from this example solution in your own implementation as you see fit. Then you should modify and/or add any relevant C# classes so that you provide the *data related functionality*, the *XML file reader functionality* and the *main GUI form functionality* required for the assignment scenario.

Unit Testing (15 marks)

Create appropriate unit tests, using the Microsoft Testing Framework, to automate the testing of one aspect of the functionality of your application. You are given the freedom to choose which aspect of your system to demonstrate unit testing but this must include at least the automated unit testing of TWO distinct methods. These automated unit tests must be properly integrated into your overall Visual Studio Solution (through the Microsoft Testing Framework) and be fully separated from the production code (not embedded within it). This will most likely require additional “fake” and/or “mock” classes to be provided within the test project so that isolation testing can be achieved. All unit test code must be subjected to the same level of quality (code structure, naming standards and commenting) as would be expected within your production code.

Recorded Screencast (5 marks)

You are required to provide a recorded screencast, demonstrating the running of your system (including all functionality) from a user perspective and evidence that your automated unit testing is working. This screencast may be created using any suitable screen recording software e.g. Camstudio and saved in a format that can be viewed on a Windows machine e.g. .avi, *.mp4 or .wmv.

Submission

Your submission should be provided as a single zipfile containing the following:

- Your UML class diagram as a Word or PDF document.
- Your Design Rationale as a Word or PDF document.
- The complete folder structure for the Visual C# solution (version VS2019), containing the projects for both your application and unit tests.
- Your screencast, which should be included in the zipfile along with your Visual Studio solution and clearly named so that it can be easily identified.

You are expected to spend around 28 hours on this assignment.

Submission Date: FRIDAY 22nd May 2020, 5:00PM via Canvas.

Marking Criteria

UML Design and Separation of Concerns 0	Some classes correctly identified. 1 to 3	Most classes correctly identified and reasonably structured. 4 to 6	All classes correctly identified and structured. Some evidence of separation of concerns identified. 7 to 9	Diagram virtually correct in structure and function (though lacking obvious interfaces, methods or attributes). Good separation of concerns. 10 to 12	Diagram correct in structure and function. Full separation of concerns demonstrated. 13 to 15
Design Rationale 0	Simple description of design but lacking detail on separation of concerns. 1 to 2	Some description of how concerns were separated but lacking justification. 3 to 4	Basic description of how concerns were separated with some attempt at justification. 5 to 6	Good description of how concerns were separated with reasonable justification. 7 to 8	Excellent description of how concerns were separated backed up with detailed justification. 9 to 10
C# Application 0	Some attempt but basic compilation errors evident. 1 to 3	Basic user experience but lacking functionality. 4 to 6	Reasonable user experience but lacking functionality. 7 to 9	Good user experience with functionality that provides a basic simulation of all the required features 10 to 12	Excellent user experience with functionality that provides a realistic simulation of all the required features 13 to 15
0	Some attempt at using interfaces but basic errors evident. 1 to 2	Basic use of one interface but minor errors or omissions present. 3 to 4	Basic use of multiple interfaces but minor errors or omissions present. 5 to 6	Good use of interfaces (though lacking in some areas). 7 to 8	Excellent use of interfaces to provide full separation of concerns 9 to 10
0	Some attempt at multi-threading but not using Producer/Consumer pattern 1 to 2	Some attempt at using Producer/Consumer pattern but basic errors evident. 3 to 4	Basic use of Producer/Consumer pattern but no multi-threading. 5 to 6	Good use of Producer/Consumer pattern with multi-threading (though lacking in some areas). 7 to 8	Excellent use of Producer/Consumer pattern to provide efficient multi-threaded functionality 9 to 10
0	Some attempt at processing XML data but basic errors present. 1 to 2	Successfully processes XML data but does not use LINQ 3 to 4	Basic attempt at using LINQ to process XML data but some errors present. 5 to 6	Good use of LINQ to process XML data (though lacking in some areas). 7 to 8	Excellent use of LINQ to process XML data. 9 to 10
0	Virtually no attempt to follow layout, coding conventions or commenting. Code very weak and does not fulfil program requirements. 1 to 2	An attempt at layout, some adherence to coding conventions and comments though generally weak. Code weak and only partly meets requirements 3 to 4	Generally good layout, adherence to coding conventions and comments though weak in parts. Code generally good fulfilling most of the program requirements. 5 to 6	Generally good layout, adherence to coding conventions and comments though weak in parts. Code generally good fulfilling all of the program requirements. 7 to 8	Excellent layout, adherence to coding conventions and comments for all classes and methods. Excellent code throughout. 9 to 10

Unit Testing 0	Some attempt but basic compilation errors evident. 1 to 3	Basic attempt at automated unit testing of a single method (eg. not involving isolation testing). 4 to 6	Good attempt at automated unit testing of a single method (eg. does involve isolation testing). 7 to 9	Good set of automated unit tests for two methods, properly isolated but lacking in some areas. 10 to 12	Excellent set of automated unit tests which thoroughly and properly tests two methods. 13 to 15
Screencast 0	Basic screencast but lacking a full demonstration of the system and/or evidence of unit testing. 1 to 2		Excellent screencast providing a full demonstration of the system and evidence of unit testing. 3 to 5		