INFO213 Course Project Milestone Two: The Terminal Operating System Implementation

Due Date and Value

Due date: **9 p.m., Sunday, May 31st 2020**

Grade contribution: **20%**

General Information

The second milestone of the course project will be building on the work completed in the first milestone.

Groupwork?

This milestone must be completed in groups of size N, where 1 <= N <=3. ☺ This is entirely up to you whether you choose to collaborate with someone or not, given that we are in the midst of odd times.

For now everyone in class is allocated to a group of one, but, you can allocate yourself to a group of your choice via UC Learn | INFO213 | Course Assessment | TOS Milestone Two Group Self-Allocation (https://learn.canterbury.ac.nz/mod/groupselect/view.php?id=1192031). There you will be able to create a new group, rename your group, or join someone-else's group, *with the approval of the other group members*.

If you are working in a group, please start by discussing the scope and subdomain of the target implementation. Decide how to coordinate your work (meetings schedule/communication means) and keep track of your decisions throughout your meetings and work.

It is suggested that, to begin with (but after the work has been divided into sub-tasks/schemas), all group members of agree on a simplified/trimmed down set of classes (and specifically their names) to work with. Similarly, it would be helpful to agree on the names of the schemas and subschemas. Following the agreed names for classes and schemas will simplify the process of loading the code contributed by different members of the group.

It is also suggested that at the very start you spend a little time practicing with extracting a schema from one computer/database and loading it into an existing schema on another computer. In addition, you might want to experiment with the advanced options of the extract functionality in JADE IDE which allows extraction of a single class. Using this functionality would help the process of collaboration and code integration.

It is hard to judge which way of completing the project is easier – individually or in a group. Individually you'll have to make all decisions or your own, with all the weight being on your shoulders, but in a group there would be a significant communication overhead, and the need to learn other tools for collaboration, such as version control.

Details for integrating GitHub Version Control are on Learn in the Housekeeping section under 'GitHub Integration with Jade2018'. This process is, however, complex; if you are new to GitHub, or have struggled with the Tutorials, then the use of Version Control over GitHub is not highly recommended for this course.

IMPORTANT: Regardless of if you are working in a Group or Individually, maintain incremental backups of your schemas, preferably on more than one device.

Scale

Needless to say, it is well beyond our capacity in this course to develop anything remotely comparable in its scale to a realistic Terminal Operating System (TOS). And, given that you may be working on your own, our resources are limited. Thus, as with Milestone One, in Milestone Two the whole group can focus the effort on one of the three subdomains of a TOS which were set as the subdomains of interest in Milestone One:

1. General cargo operations/solutions
2. Logs tracking and management
3. Invoicing

As with Milestone One, the starting point is the JADE Master Terminal Overview available along with this specification via the UC Learn | INFO213 | Course Assessment page. In your group you may choose a Milestone One solution of one of your group members. Thus, your implementation may follow an existing set of solutions provided by the group members. However, if you feel the need to adjust the original solution, you are free to do so.

A good project implementation will need to demonstrate competencies in the following areas and specifically, your integrated approach to the prototype implementation:

1. Ability to work in a group, *if you choose to do so;*
2. Ability to implement UML class diagrams in code;
3. Ability to create GUIs to match use case scenarios;
4. Work with data in XML format using the appropriate features of the JADE XML processing API;
5. Create test basic unit test sets with appropriate input data.

Clarification of Requirements/Additional Information

NEEDLESS TO SAY, THE STARTING REQUIREMENTS ARE INCOMPLETE.

You are encouraged to post your questions about the missing requirements on the General INFO213 Forum and your "TOS client" will attempt to resolve the questions and provide additional information as soon as possible. However, please **don't** leave the questions/clarifications to the last two days (the weekend) before the submission deadline!

IMPRORTANT:

AS WITH THE PREVIOUS MILESTONE, MS2 IS QUITE OPEN-ENDED.

The key point to keep in mind is that the main focus of your work is to **PRODUCE A BASIC PROTOTYPE IMPLEMENTATION** for some of the use cases for the chosen TOS subdomain.

You should produce **A SOLUTION THAT INTEGRATES THE VARIOUS COMPONENTS SUCH AS MODEL, INTERFACE, INPUT, TESTING AND DOCUMENTATION.**

PLEASE DON'T LET THIS FOCUS TO BE OBSCURED BY THE ALLURE OF A GRANDIOSE COMPLETE IMPLEMENTATION.

Milestone Two (MS2) Task

You are required to implement a minimal working prototype of one of three TOS subdomains listed earlier. The prototype should include the following:

1. The core classes, as chosen from the options of General Cargo, Logs, or Invoicing, with the key relationships between those classes.
2. A very simple MDI-based interface for your implementation to reflect the following use cases:
   - Import core objects of the domain from XML-base file format. The implementation of the XML handling code must include appropriate XML error/exception handling and error/exception logging to a file for future review/debugging/corrections.
   - Display the core objects in tabular format and allow for modification of the data in a suitable manner. This may include creation of new instances of the core objects, or allocation of objects into containment hierarchies as dictated by the relationships between those core objects.
     - The implementation of this functionality must include appropriate error/exception handling and error/exception logging to a file for future review/debugging/correction.
       - **PLEASE NOTE, IT IS A GOOD IDEA TO HAVE AN EXCEPTION HANDLER THAT PROVIDES A REASONABLY DETAILED MESSAGE TO THE SYSTEM USER WHEN AN ERROR OCCURS!**

3. A simple set of unit tests which would serve as a starting point for building up the testing functionality. This does not need to be an exhaustive test set, but it should at least verify the following:
   - A correctly formatted minimal dataset can be imported into the prototype implementation.
   - A “broken” minimal dataset will cause an exception that will be caught and logged to a file on disk.
4. The implementation documentation/description where you briefly illuminate the following aspects of your solution:
   - What functionality is implemented and how they can be verified via the GUI provided with the prototype.
   - What exception handling/logging functionality is implemented for the core objects and the XML data import code.
   - List of unit tests are provided

Sample Datasets

Along with this specification there is a bunch of sample data to help you get started, but some of those datasets are related to another domain.

Please note, that these datasets are somewhat substandard, and are simply intended as a starting point. It is recommended that you tweak them to your needs, instead of adjusting your implementation to the data, which would be the wrong direction in this case.

Alternatively, you may want generate your own datasets with the help of the one of the following random data generator websites:

- Generate Data: http://www.generatedata.com/
- Mockaroo: https://mockaroo.com/

Please explore the handy functionality offered by these websites instead of crafting the test datasets by hand. Creating broken datasets will necessitate editing (breaking) to suit the purpose – testing erroneous input.

Submission Requirements

The MS2 submissions must include the following components:

1. **Assignment Coversheet/Contribution declaration**:
   - Please use the ACIS Individual or Group Assignment Coversheet provided under the Course Project Milestone Two section on the UC Learn | INFO213 | Course Assessment page. This is a collective document signed by all group members stating the individual contributions to the project. In the simplest/ideal case this document will simply need to state that all group members contributed to the project in equal measure.
2. **Documentation**: this is the starting point for grading the MS2 TOS implementation. The document must be in MS Word or PDF format, in the range of three to five pages.
   - The document must include a revised version of the UML class diagram, as it is likely that during the implementation process you will find the parts of the proposed initial solution that will need to be refined/changed/clarified.
   - The document must describe what you have done, how it all fits together, how you tested what you implemented, and what you have done in terms of error handling.
3. **Code**: this is the implementation code (schemas) that can be loaded into JADE IDE and run as an application.
   - Please remember to define a GUI application for one of your schemas.
4. **Data**: this is the test data to be loaded into the application to demonstrate the functionality.

PLEASE SUBMIT ALL FOUR REQUIRED ITEMS AS ONE DIGITAL COMPRESSED ARCHIVE FILE IN ZIP FORMAT VIA UC Learn | INFO213 | Course Assessment | Course Project Milestone Two.

Marking Schedule

The MS2 TOS Implementation submission will be graded out of 100 marks with the following breakdown:

- **DOCUMENTATION:** **20 MARKS**
  - Simplicity
  - Consistency
  - Presentation quality
  - Matches Code
  - List of Unit Tests

- **CODE:** **30 MARKS**
  - Appropriate functionality, as described in the documentation.

- **EXCEPTION/ERROR HANDLING:** **30 MARKS**
  - Clear, simple instructions in the documentation, which will help verify how to trigger an exception, what message is produced, and what is recorded in the log file
  - Clear message production in the program
  - Record of the error sent to a Log File

- **UNIT TESTS:** **20 MARKS**
  - A small set of unit tests which verify the functionality of your code
    - the set of tests may include items which are skipped/failing if this is not related to the basic functionality related to import and display of information.
    - The key focus of this section is demonstrating your thinking about testing the code.