

## Faculty of Technology – Course work Specification 2019/20

<b>Module name:</b>	Object Oriented Development		
<b>Module code:</b>	CTEC2906		
<b>Title of the Assignment:</b>	JavaFX Final Year Module Selection GUI		
<b>This coursework item is:</b> (delete as appropriate)	Summative		
<b>This summative coursework will be marked anonymously</b>		No	
<b>The learning outcomes that are assessed by this coursework are:</b> <ol style="list-style-type: none"> <li>1. Design and develop applications with emphasis on quality, maintainability, correctness and robustness, enhancing their trustworthiness.</li> <li>2. Make effective use of the Java Software Development Kit Application Programming Interfaces</li> </ol>			
<b>This coursework is:</b> (delete as appropriate)	Individual		
<b>This coursework constitutes 100% to the overall module mark.</b>			
<b>Date Set:</b>	Monday 16 <sup>th</sup> March 2020 at 00:01		
<b>Date &amp; Time Due:</b>	Thursday 30 <sup>th</sup> April 2020 at 14:00		
<b>Your marked coursework and feedback will be available to you on:</b> If for any reason this is not forthcoming by the due date your module leader will let you know why and when it can be expected. The Associate Professor Student Experience ( <a href="mailto:cemstudentexperience@dmu.ac.uk">cemstudentexperience@dmu.ac.uk</a> ) should be informed of any issues relating to the return of marked coursework and feedback.  Note that you should normally receive feedback on your coursework by <b>no later than 20 University working days after the formal hand-in date</b> , provided that you have met the submission deadline.			By 2/6/20.
<b>When completed you are required to submit your coursework to:</b> <ol style="list-style-type: none"> <li>1. Blackboard VLE through a submission portal. See attached document for further detail.</li> </ol>			
<b>Late submission of coursework policy:</b> Late submissions will be processed in accordance with current University regulations which state: <i>"the time period during which a student may submit a piece of work late without authorisation and have the work capped at 40% [50% at PG level] if passed is 14 calendar days. Work submitted unauthorised more than 14 calendar days after the original submission date will receive a mark of 0%. These regulations apply to a student's first attempt at coursework. Work submitted late without authorisation which constitutes reassessment of a previously failed piece of coursework will always receive a mark of 0%."</i>			
<b>Academic Offences and Bad Academic Practices:</b> These include plagiarism, cheating, collusion, copying work and reuse of your own work, poor referencing or the passing off of somebody else's ideas as your own. If you are in any doubt about what constitutes an academic offence or bad academic practice you must check with your tutor. Further information and details of how DSU can support you, if needed, is available at: <a href="http://www.dmu.ac.uk/dmu-students/the-student-gateway/academic-support-office/academic-offences.aspx">http://www.dmu.ac.uk/dmu-students/the-student-gateway/academic-support-office/academic-offences.aspx</a> and <a href="http://www.dmu.ac.uk/dmu-students/the-student-gateway/academic-support-office/bad-academic-practice.aspx">http://www.dmu.ac.uk/dmu-students/the-student-gateway/academic-support-office/bad-academic-practice.aspx</a>			
<b>Tasks to be undertaken:</b> See (following) attached document.			
<b>Deliverables to be submitted for assessment:</b> See (following) attached document.			
<b>How the work will be marked:</b> See (following) attached document.			
<b>Module leader/tutor name:</b>	Luke Attwood		
<b>Contact details:</b>	<a href="mailto:lattwood@dmu.ac.uk">lattwood@dmu.ac.uk</a> (GH6.71)		

## JavaFX Final Year Module Selection GUI

### Objectives

The objective of this assessment is for you to demonstrate your ability to design and implement an OO system consisting of a set of Java classes, using advanced libraries within the Java SDK. In particular:

1. To study and correctly make use of a prebuilt student profile data model.
2. To build a suitable user interface using JavaFX 8 libraries.
3. To implement event handling procedures that provide a basis for an interactive and user-friendly system.
4. To adhere to standard principles of the Model-View-Controller (MVC) design pattern and appropriately decompose classes through abstraction and encapsulation.

### Submission

Submit your Eclipse project through the submission portal available on Blackboard. You should create a new folder called “CTEC2906Assignment”, copy your Eclipse project into this folder, ensuring all of the Java source files (and any other associated files) are in it. Then compress the folder into a .zip file, which you can submit through Blackboard.

### Marking and Feedback

Your work will be marked using a predefined Blackboard VLE Rubric grid, which will display your level of achievement and feedback for each of the assessment criteria for this assignment (*shown later in this specification*). The categories within the rubric act as a guide and the 'closest fit' will be selected for each individual piece of work, ensuring the overall result is reflective of the DMU Undergraduate Mark Descriptors.

### Important Note

Please do **NOT** be tempted to search the internet for an existing solution and then hand this in – e.g. if you run out of time. This is cheating – it is better to hand in what you have honestly achieved by yourself than try to get credit for someone else’s work and risk getting caught. Cheating is an **academic offence**.

If you get stuck then please ask the module tutors for assistance and come to the labs – we will be providing help in the labs (but not actually doing it for you of course).

Do **NOT** use anyone else’s material without referencing it – this is **bad academic practice or plagiarism**. These are considered as **academic offences**.

Do **NOT** work jointly on a solution; do **NOT** give your solution to anyone else to “help them” and do **NOT** accept anyone else’s solution as “guidance”. Such practice could lead to an allegation of **collusion** which is an **academic offence**. **All parties** involved in collusion (the givers, the receivers, the collaborators) can be found guilty of an academic offence, irrespective of motive.

## JavaFX Module Selection GUI specification

A student profile captures the details of an individual second year undergraduate computing student and allows them to select their final year module options. There are compulsory modules that must be selected (depending on the course of study), and others that are only associated with certain courses. Modules either run in term 1 or 2, or all year long.

Your task is to build an interactive graphical user interface (GUI) that dynamically allows modules to be selected based on the chosen course of study, and then stores this information. The application should be user-friendly and contain appropriate validation to ensure only a legitimate selection of modules is made.

For this prototype, you are only required to use the data of two courses, Computer Science and Software Engineering. However, the system should be designed such that it would be relatively simple to add further courses and modules in the future.

The table overleaf shows all of the available modules, their credit amount, and whether they are an option or compulsory for Computer Science and Software Engineering students. Computer Science students have 45 compulsory credits, whereas Software Engineering students have 60 compulsory credits. Computer Science students can exclusively study IMAT3428.

In total 120 credits must be selected via any legitimate combination of modules, but crucially you may only select 60 credits per term. The yearlong compulsory module CTEC3451 contributes towards 15 credits in each term.

As an example, a Computer Science student would have by default 30 credits selected in term 1 and 15 credits in term 2 due to the mandatory nature of IMAT3423 and CTEC3451. This would mean they would need to select a further 30 credits of term 1 modules, and a further 45 credits in term 2. A Software Engineering student would be similar, but would additionally have the mandatory module CTEC3902 in term 2, therefore requiring an additional 30 credits worth of modules to be chosen in both term 1 and term 2.

In this case study, once students have successfully chosen a legitimate selection of modules as their first preference, they are then required to select 30 credits worth of reserve modules per term from those that remain, i.e. two term 1 modules and two term 2 modules. These reserve modules can then be used in the unlikely scenario of another module not running or being at full capacity for example.

*See module table overleaf...*

Code	Title	Credits	Term	Computer Science	Compulsory	Software Engineering	Compulsory
IMAT3423	Systems Building: Methods	15	1	x	x	x	x
CTEC3451	Development Project	30	Y	x	x	x	x
CTEC3902	Rigorous Systems	15	2	x		x	x
CTEC3605	Multi-service Networks 1	15	1	x		x	
CTEC3606	Multi-service Networks 2	15	2	x		x	
CTEC3110	Secure Web Application Development	15	1	x		x	
CTEC3410	Web Application Penetration Testing	15	2	x		x	
CTEC3904	Functional Software Development	15	2	x		x	
CTEC3905	Front-End Web Development	15	2	x		x	
CTEC3906	Interaction Design	15	1	x		x	
IMAT3104	Database Management and Programming	15	2	x		x	
IMAT3611	Computer Ethics and Privacy	15	1	x		x	
IMAT3406	Fuzzy Logic & Knowledge Based Systems	15	1	x		x	
IMAT3613	Data Mining	15	1	x		x	
IMAT3614	Big Data & Business Models	15	2	x		x	
IMAT3428	Information Technology Services Practice	15	2	x			

*Please turn over...*

## Guidance on building the application

*You are advised to thoroughly read this guidance and to continually refer to it as a means of helping you design and implement the JavaFX Module Selection GUI application.*

Please download **OptionsModuleSelection.zip** and extract the Java Project, then import it into Eclipse. You will note that it contains a template structure for you to work from including logical packages.

### Application Loader

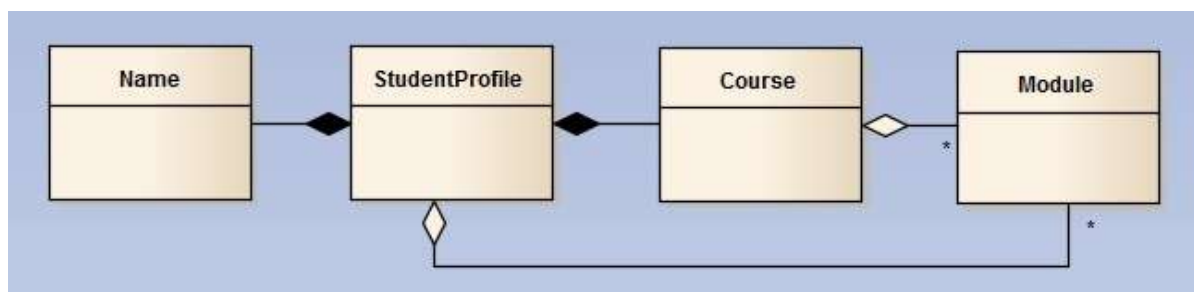
In the main package you have been given an `ApplicationLoader` that simply instantiates the model (i.e. `StudentProfile`), view (i.e. `ModuleSelectionRootPane`) and passes these to the controller (i.e. `ModuleSelectionController`).

This should help you get started. You should clearly showcase the MVC design pattern throughout your implementation. You will be assessed on your ability to sensibly decouple these entities to make a maintainable and reusable solution.

### The Model

You have been provided with the data model for this application - The model contains four classes – `StudentProfile`, `Course`, `Module`, and `Name`, and an enum `Delivery`.

- A **Student Profile** encapsulates a student name, P number, email, submission date, a Course of study, and a set of selected Modules, along with a set of reserved Modules.
- A **Course** has a name and a collection of all Modules available on that course.
- A **Module** has a code, name, credit amount, is either mandatory or not, and has a delivery run plan of either term 1, term 2, or year long.
- A **Name** for a student is made up of a first name and a family name.



The model classes purposely have no javadoc comments, as you are required to study the source code in order to understand how to use them as the data model for your MVC application. You will not be assessed on writing javadoc for this assignment either.

For the vast majority of this assignment, you will not need to add any further code to the model and are advised to make use of existing methods wherever possible.

## The View

In the `view` package you have been given a root pane, from which you can build your user interface. The GUI is made up of four forms, which should be separated onto different tabs. You have been given the first of these forms (placed onto a tab), along with a menu bar. There are also some methods in these classes to help you get started. The overall GUI should include the following:

### Create Profile tab

Displays a combo box, pre-populated with the two aforementioned second year computing courses, and five text fields for inputting a student P number, first and last name, email, and the submission date. There is also a *create profile* button.

### Select Modules tab

Should display two list views for unselected term 1 and term 2 modules (for the chosen course), and three further list views for selected year long, term 1 and term 2 modules (including compulsory modules). The accumulated term1 and term 2 credits for the current module selection should be displayed. There should be *add* and *remove* buttons for both term 1 and term 2 modules, and an overall *reset* and *submit* button.

### Reserve Modules tab

Should display the remaining unselected modules for each term. Reserve modules should be chosen (using unselected/selected list views) for term 1, followed by term 2 (ideally using an `Accordion` control to only display one term at a time). There should be *add* and *remove* buttons for both term 1 and term 2 modules, as well as a *confirm* button for each term.

### Overview Selection tab

Should display an overview of the student's details, selected modules and reserved modules based on their submitted profile and module selection from the previous three tabs. The information should be clear and well presented across three separate text areas. There should also be a *save overview* button.

Your GUI also has a menu bar:

### The Menu Bar

This Menu bar has two menus: File and Help. The File menu is made up of menu items: "Load Student Data", "Save Student Data" and "Exit". The Help menu is made up of a single menu item: "About".

**Note:** You are strongly advised to decompose the view by separating logical containers into their own top-level classes, enhancing reusability and abstraction. You will also need to provide suitable methods that allow relevant data to be accessed and modified within the view, adhering to principles of encapsulation and forming a clean public interface.

You can see the **Sample GUI screenshots** section at the end of this document to assist in visualising how the GUI should generally look and behave, although you may use this as a guide and be creative where applicable.

## **The Controller**

The model and view have been decoupled by deploying a MVC architecture with a separate controller, which will contain the event handlers and general behaviour for this application. To help you get started with this you have been provided with a `controller` package containing the `ModuleSelectionController` class. So far the controller initialises a reference to the model and view, along with its current subcontainers.

You will also notice a method within the controller class called `setupAndGetCourses`, which populates an array of type `Course` with the required hardcoded module data. When you load the GUI, the "Create Profile" tab displays a combo box, populated with the two aforementioned computing courses. You will note the `populateCourseComboBox` method undertakes this task.

**Event Handling:** The remaining key purpose of the controller is to define event handlers. To help you get started you have been given the exit handler along with an empty create profile handler, which is attached to the create profile pane in the view.

See below details of what each event handler should broadly do. Note there are multiple add and remove handlers required across both the select and reserve modules tabs (please refer to Sample GUI Screenshots). It is also assumed you will consider appropriate validation to aid the behaviour of the application where relevant:

- **Create Profile Handler** - captures the details of the student and stores them within the data model (including their selected course). It should then dynamically populate the select modules tab with the initial unselected and (compulsory) selected modules based on the chosen course.
- **Reset Handler** - resets the selected modules tab to populate the initial unselected and (compulsory) selected modules based on the current chosen course.
- **Add Handlers** - removes the current selected module from the unselected modules list view and adds it to the selected (or reserved) modules list view (for a given term).
- **Remove Handlers** - removes the current selected module from the selected (or reserved) modules list view and adds it back to the unselected modules list view (for a given term).
- **Submit Handler** - captures the details of the selected modules and adds them into the data model. It should then dynamically populate the reserve modules tab with any remaining unselected term 1 and term 2 modules.
- **Confirm Handlers** - captures the details of the reserved modules for a given term and adds them into the data model. Once both terms have been confirmed an overview of the student's details, selected modules and reserved modules should appear dynamically on the overview selection tab.
- **Save Overview Handler** - save an overview of the student's details, as well as their selected and reserved modules in text format using a `PrintWriter` (or suitable alternative), choosing a custom representation.
- **Save Profile Handler** - save the student profile (data model) to a file in binary format using an `ObjectOutputStream`.

- **Load Student Data Handler** - restore the student profile (data model) from a file using an `ObjectInputStream`. You should also consider how best to bring the GUI (view) back to its previous state (at the point of saving).
- **About Handler** - trigger a popup window containing basic details of the application.
- **Exit Handler** - close the current window and terminate the application.

**Note:** Whilst you can use the hardcoded module data provided in the controller, a more advanced requirement of the application, would be to use a `Scanner` to dynamically load the course and module data into the application from a text file. This would allow for the input data to be more easily evolved in the future.

*Please turn over...*



## Assessment Criteria

The following criteria show how you will be assessed:

- **User Interface (30%):** The user interface appropriately displays and captures all relevant data with a suitable layout.
- **Resizing behaviour (10%):** There has been consideration of resizing the application using appropriate layout policies. The list views displaying modules and the overview of the selection should all maximize available space both horizontally and vertically.
- **Event Handling (20%):** The user interface is responsive to interaction and can be used to achieve all associated tasks.
- **Fitness for purpose (10%):** There is appropriate validation throughout the application to ensure operations behave correctly, and features cannot be misused.
- **MVC Design & View Decomposition (20%):** The Model-View-Controller (MVC) design pattern has been applied to separate concerns and reduce coupling. The view has had logical containers separated into their own top-level classes, with suitable public interfaces. Abstraction has been applied throughout the design to reduce duplicate functionality and enhance maintainability.
- **Saving & Loading (10%):** An overview of the student's details and their selected and reserved modules can be written to a file in text form. The current student profile can be saved to a file in binary form and seamlessly restored at a later time. Courses and associated modules can be dynamically pre-loaded into the application from a text file.

**Important note:** You must develop the GUI using JavaFX (and not any other Java framework such as Swing or AWT). Furthermore, you cannot use a GUI builder of any kind. Failure to meet either of these two requirements will result in an automatic mark of zero for this assignment.

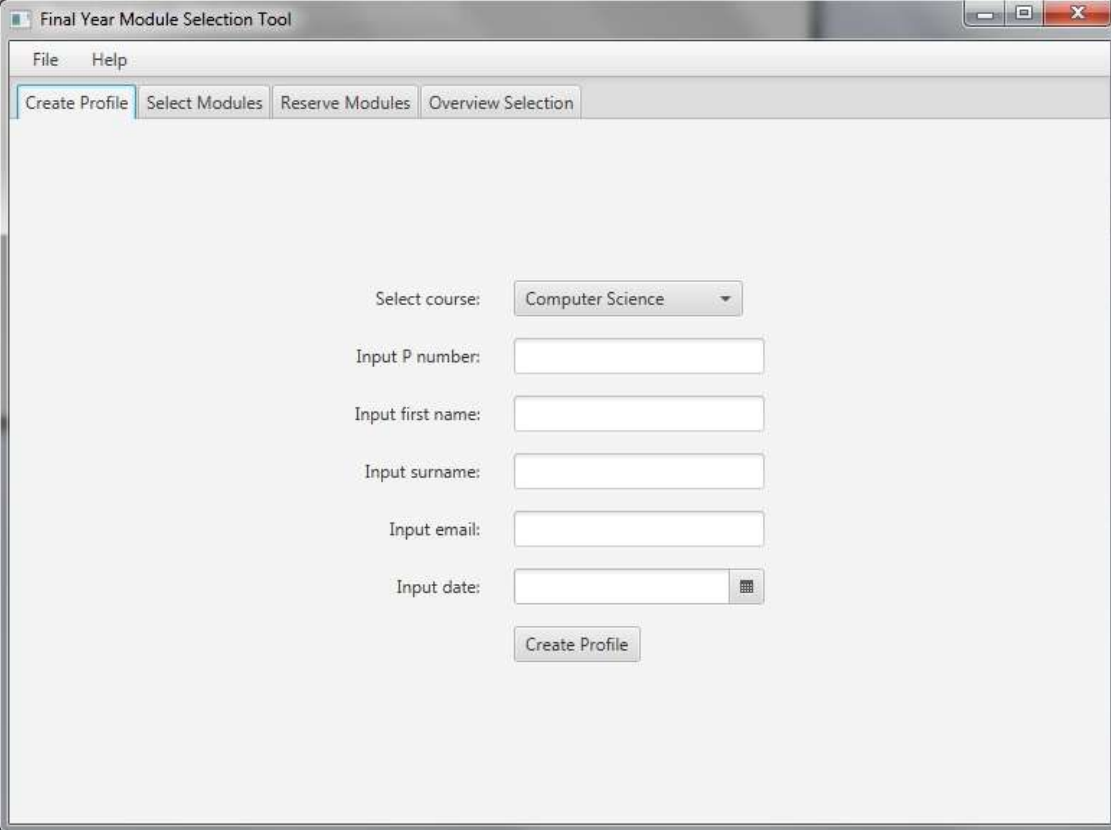
## Sample GUI screenshots

These screenshots are provided to help you visualise the core details that the user interface should capture and display. This prototype design is provided as a guide and you do not need to try and replicate the exact layout shown - you are encouraged to be creative where applicable within the context of the requirements outlined within this specification.

*Please turn over to view screenshots...*

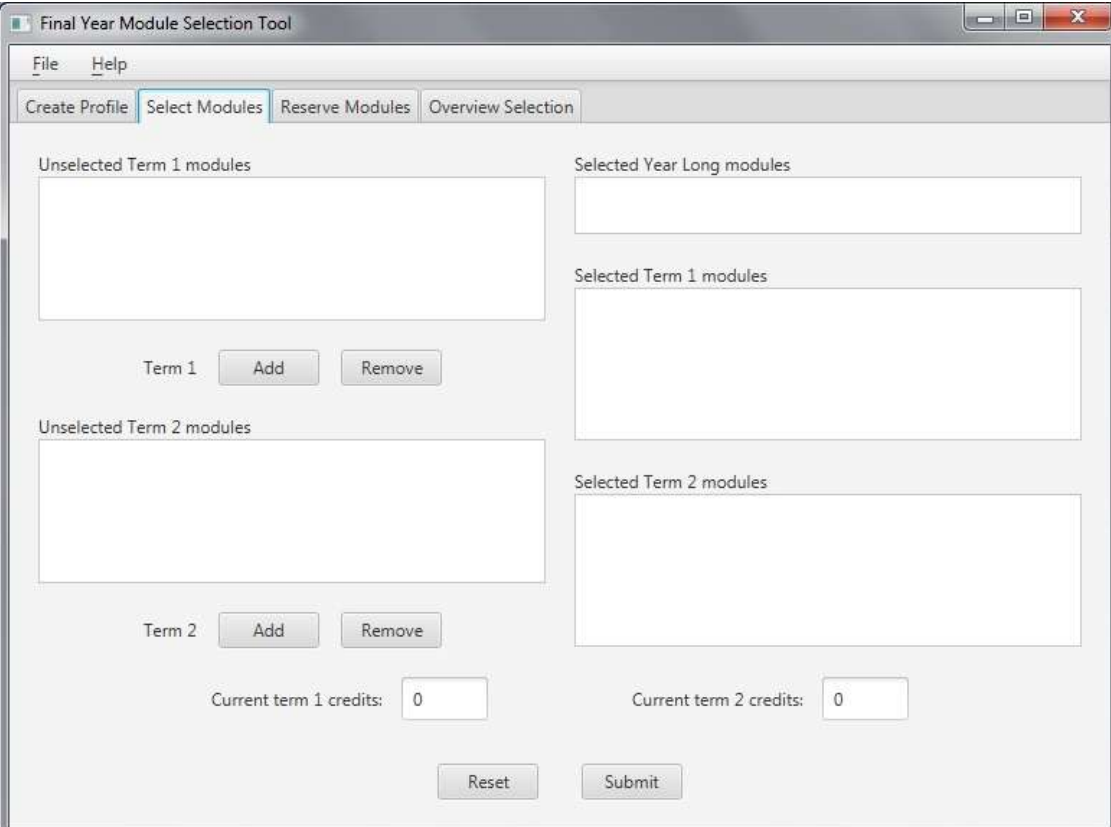
**Default form display**

When the application is first loaded, the four forms (of which you have been given the first) will have default appearances, e.g.



The screenshot shows the 'Final Year Module Selection Tool' window with the 'Create Profile' tab selected. The window has a menu bar with 'File' and 'Help'. Below the menu bar are four tabs: 'Create Profile', 'Select Modules', 'Reserve Modules', and 'Overview Selection'. The main area contains the following form elements:

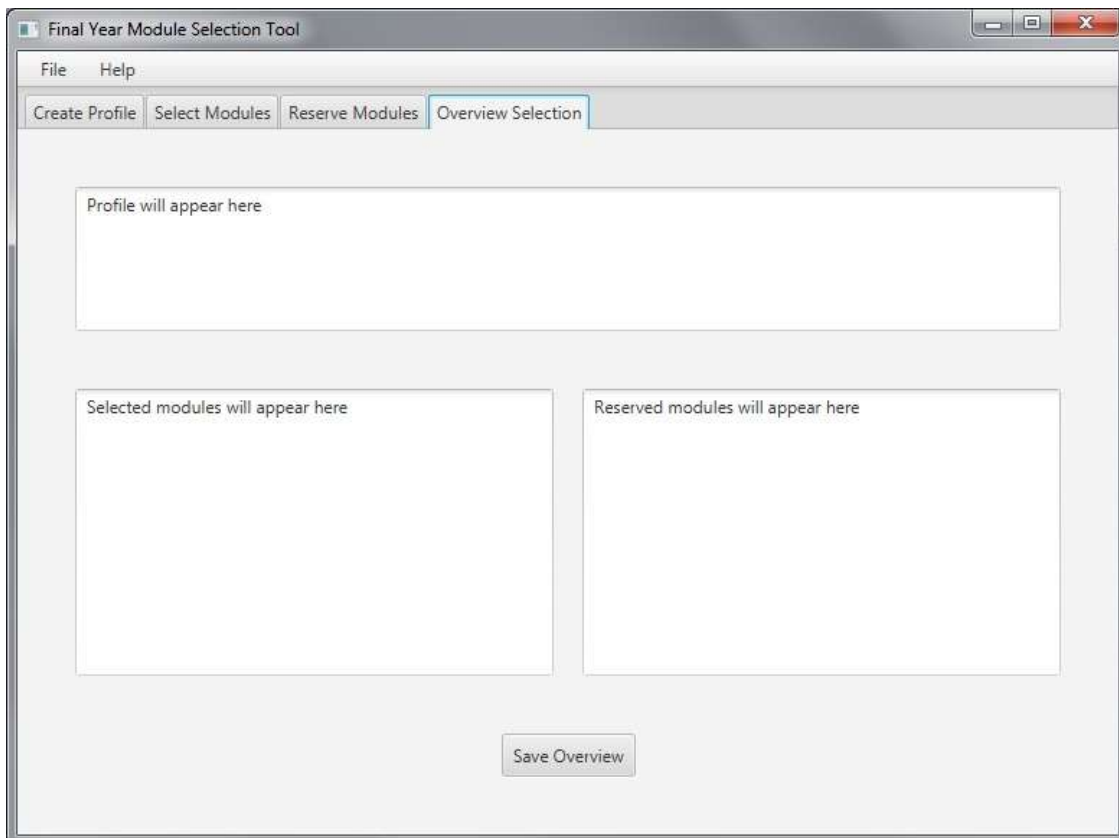
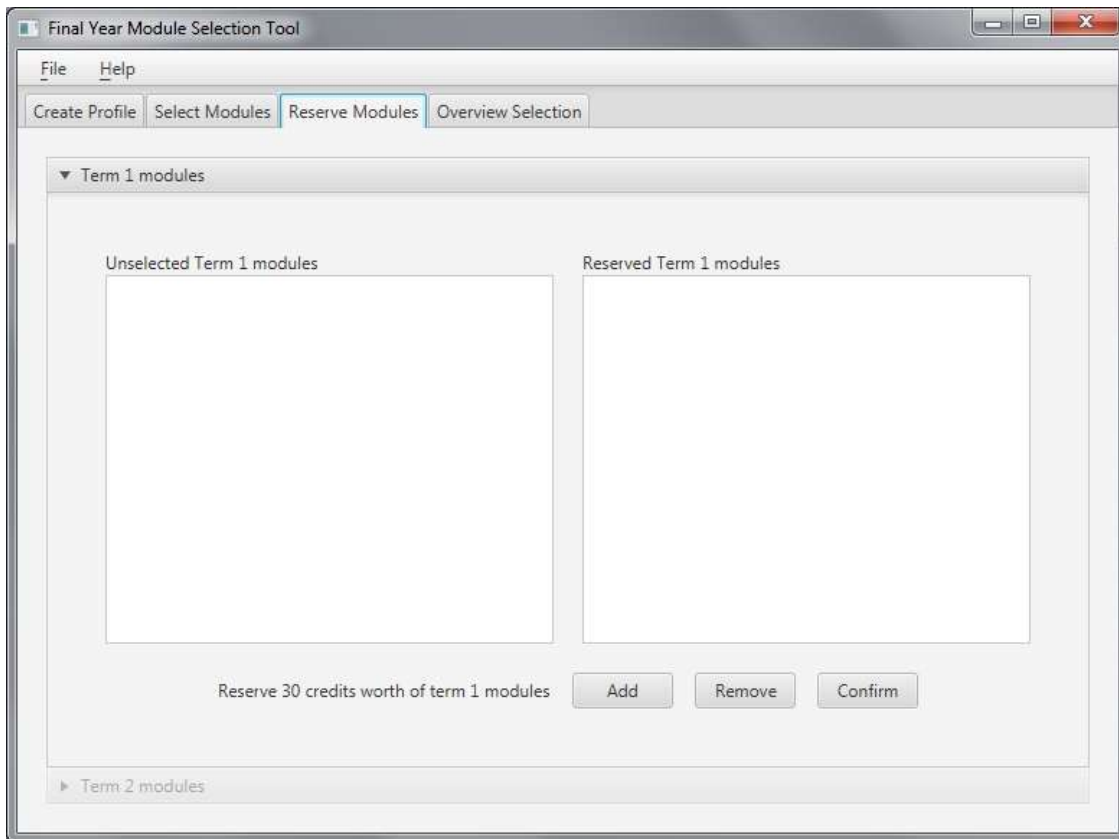
- 'Select course:' with a dropdown menu showing 'Computer Science'.
- 'Input P number:' with a text input field.
- 'Input first name:' with a text input field.
- 'Input surname:' with a text input field.
- 'Input email:' with a text input field.
- 'Input date:' with a text input field and a calendar icon.
- A 'Create Profile' button at the bottom.



The screenshot shows the 'Final Year Module Selection Tool' window with the 'Select Modules' tab selected. The window has a menu bar with 'File' and 'Help'. Below the menu bar are four tabs: 'Create Profile', 'Select Modules', 'Reserve Modules', and 'Overview Selection'. The main area is divided into four sections for module selection:

- 'Unselected Term 1 modules' with a large text area.
- 'Selected Year Long modules' with a large text area.
- 'Selected Term 1 modules' with a large text area.
- 'Selected Term 2 modules' with a large text area.

Below the 'Unselected Term 1 modules' section, there is a 'Term 1' label and two buttons: 'Add' and 'Remove'. Below the 'Unselected Term 2 modules' section, there is a 'Term 2' label and two buttons: 'Add' and 'Remove'. At the bottom, there are two input fields for credits: 'Current term 1 credits: 0' and 'Current term 2 credits: 0'. At the very bottom are two buttons: 'Reset' and 'Submit'.



## Using the GUI

A profile is created by selecting a course and inputting student details. The details should be stored in the student profile data model. The select modules form should then be loaded.

The screenshot shows the 'Final Year Module Selection Tool' window with the 'Create Profile' tab selected. The window has a menu bar with 'File' and 'Help'. Below the menu bar are four tabs: 'Create Profile', 'Select Modules', 'Reserve Modules', and 'Overview Selection'. The main area contains the following fields:

- 'Select course:' with a dropdown menu showing 'Software Engineering' (selected), 'Computer Science', and 'Software Engineering' (highlighted).
- 'Input P number:' with an empty text field.
- 'Input first name:' with a text field containing 'Joe'.
- 'Input surname:' with a text field containing 'Bloggs'.
- 'Input email:' with a text field containing 'jb@dmu.ac.uk'.
- 'Input date:' with a text field containing '16/03/2020' and a calendar icon.
- A 'Create Profile' button at the bottom.

Depending on whether Software Engineering or Computer Science is chosen, associated modules will dynamically populate the select modules form, with the respective initial term-based credit amounts shown based on the mandatory modules for a given course. You should utilise `ListView<Module>` controls to capture and display modules on this form.

The screenshot shows the 'Final Year Module Selection Tool' window with the 'Select Modules' tab selected. The window has a menu bar with 'File' and 'Help'. Below the menu bar are four tabs: 'Create Profile', 'Select Modules', 'Reserve Modules', and 'Overview Selection'. The main area is divided into two columns:

- Left Column:**
  - 'Unselected Term 1 modules' list: CTEC3110 : Secure Web Application Development, CTEC3426 : Telematics, CTEC3605 : Multi-service Networks 1, CTEC3906 : Interaction Design.
  - 'Term 1' section with 'Add' and 'Remove' buttons.
  - 'Unselected Term 2 modules' list: CTEC3410 : Web Application Penetration Testing, CTEC3606 : Multi-service Networks 2, CTEC3904 : Functional Software Development, CTEC3905 : Front-End Web Development.
  - 'Term 2' section with 'Add' and 'Remove' buttons.
  - 'Current term 1 credits:' text field with value '30'.
  - 'Current term 2 credits:' text field with value '30'.
- Right Column:**
  - 'Selected Year Long modules' list: CTEC3451 : Development Project.
  - 'Selected Term 1 modules' list: IMAT3423 : Systems Building: Methods.
  - 'Selected Term 2 modules' list: CTEC3902 : Rigorous Systems.

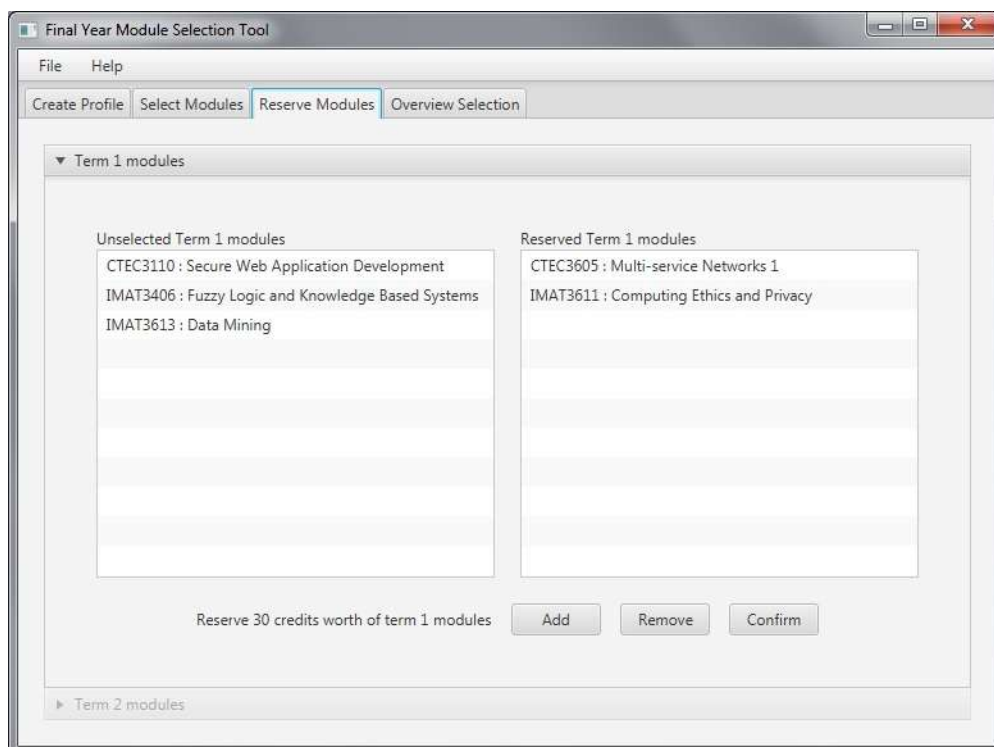
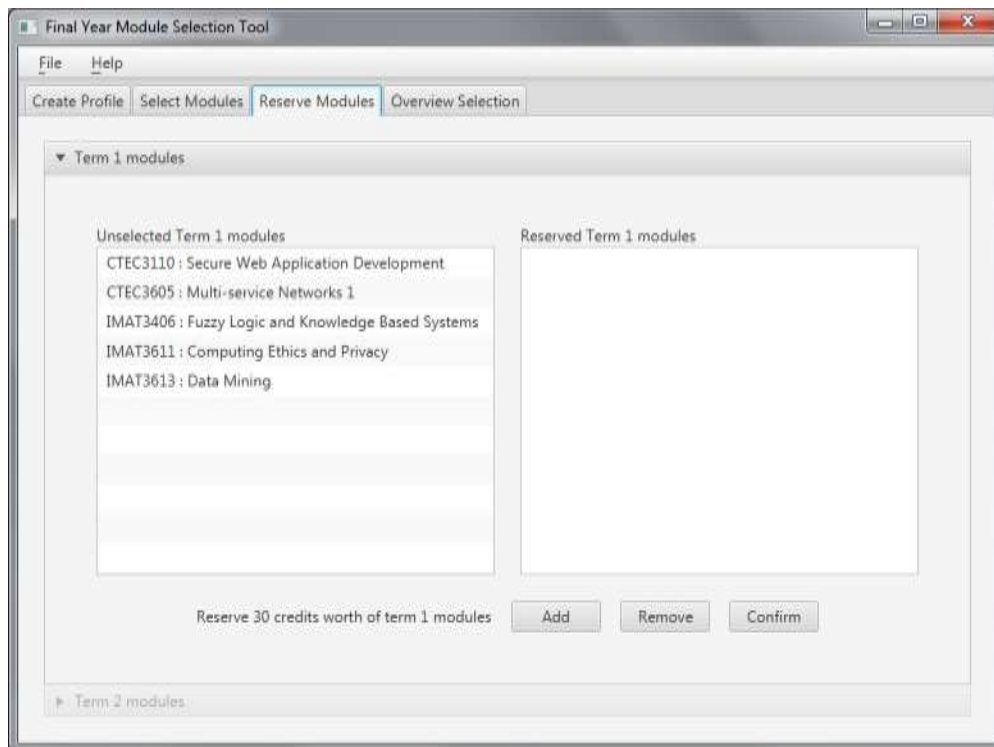
At the bottom of the window are 'Reset' and 'Submit' buttons.

The screenshot shows the 'Final Year Module Selection Tool' window. The 'Select Modules' tab is active. On the left, there are two lists of 'Unselected' modules for Term 1 and Term 2. Term 1 has 4 modules: CTEC3110, CTEC3426, CTEC3605, and CTEC3906. Term 2 has 4 modules: CTEC3410, CTEC3606, CTEC3902, and CTEC3904. On the right, there are two lists of 'Selected' modules. The 'Selected Year Long modules' list contains CTEC3451. The 'Selected Term 1 modules' list contains IMAT3423. The 'Selected Term 2 modules' list is empty. At the bottom, the 'Current term 1 credits' is 30 and 'Current term 2 credits' is 15. There are 'Add' and 'Remove' buttons for each term, and 'Reset' and 'Submit' buttons at the bottom.

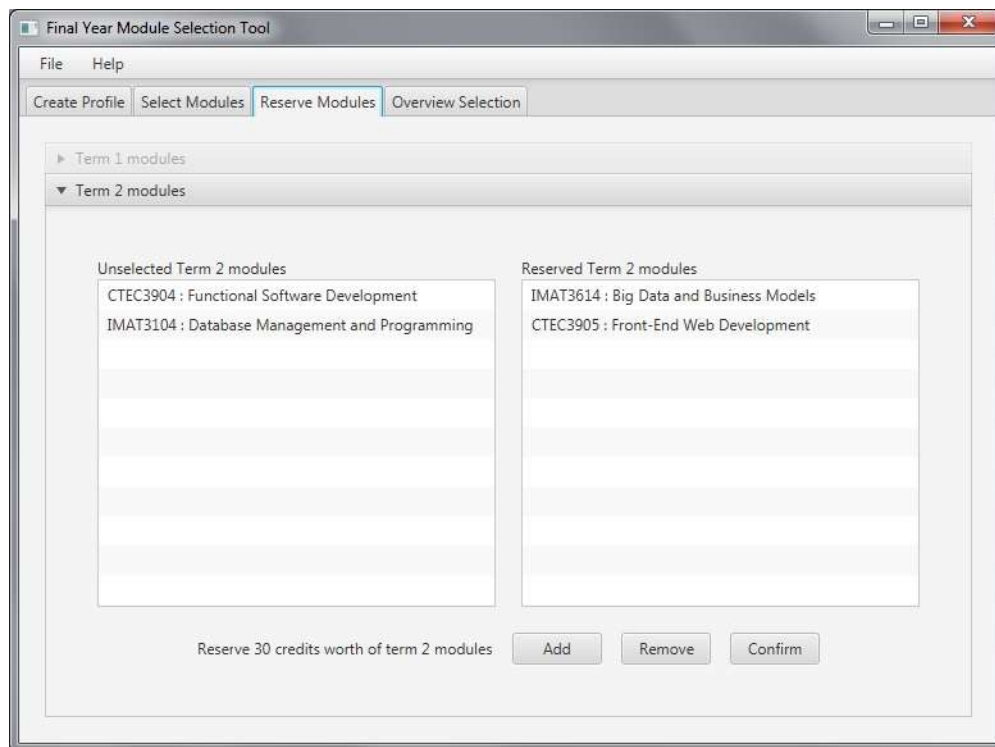
The add/remove buttons allow modules to be chosen for a given term. Once 120 credits have been selected (and no more than 60 per term) you should submit the modules to the student profile model, and direct the user to select reserve modules. The reset button should bring this form back to its default state based on the course selected (as shown in previous screenshots).

The screenshot shows the 'Final Year Module Selection Tool' window after more modules have been selected. The 'Select Modules' tab is still active. The 'Unselected Term 1 modules' list now contains 4 modules: CTEC3110, CTEC3605, IMAT3406, and IMAT3611. The 'Unselected Term 2 modules' list contains 4 modules: CTEC3904, CTEC3905, IMAT3104, and IMAT3614. The 'Selected Year Long modules' list still contains CTEC3451. The 'Selected Term 1 modules' list now contains 3 modules: IMAT3423, CTEC3426, and CTEC3906. The 'Selected Term 2 modules' list now contains 3 modules: CTEC3902, CTEC3410, and CTEC3606. At the bottom, the 'Current term 1 credits' is now 60 and 'Current term 2 credits' is also 60. The 'Add' and 'Remove' buttons for each term are still present, along with 'Reset' and 'Submit' buttons at the bottom.

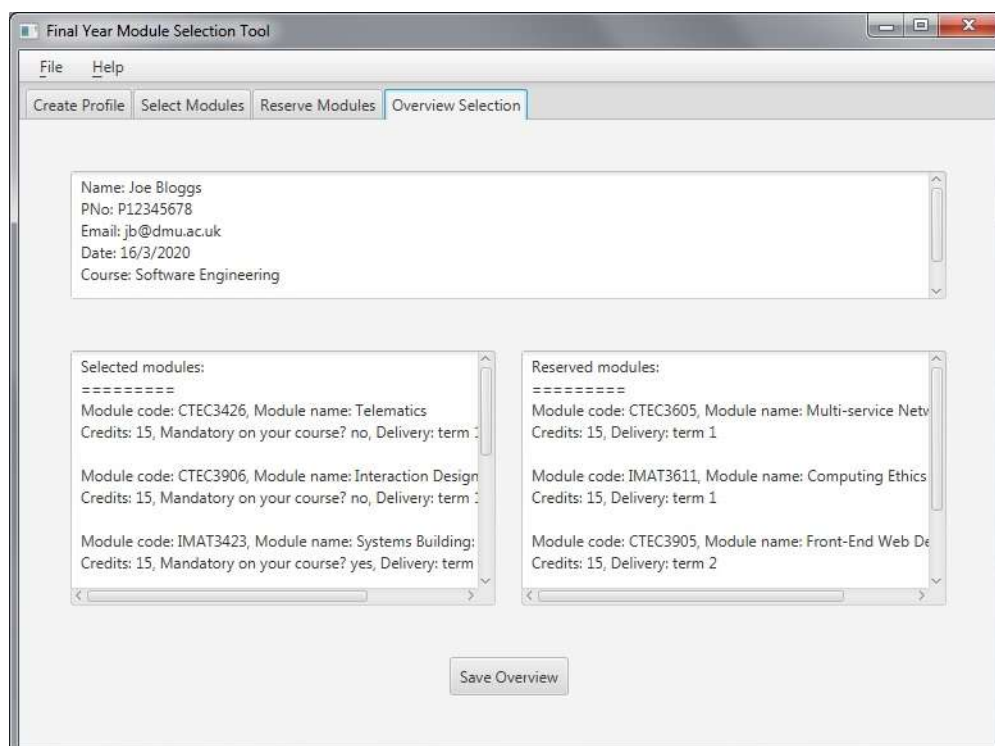
The reserve modules form should display one term at a time (e.g. using an `Accordion`). The add/remove buttons allow modules to be reserved for the given term.



Once 30 credits per term have been reserved the confirm button should store these modules in the student profile data model. Upon reserving term 1 modules, the confirm button should then display the term 2 reserve module selection.

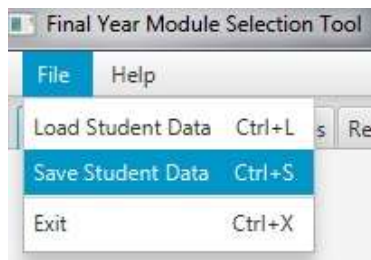


Once the term 2 reserve modules have been chosen, the confirm button should load the overview selection form. An overview of the user's details, along with their selected and reserved modules should be displayed dynamically using three text areas to maximise space. There is also the ability to save this textual information to a file.



### The menu bar

The File menu will allow the data model to be saved to a file in binary form, and restored at a later time, which would load the user interface back to a sensible previous state.



### Resizing the GUI

You may impose a sensible minimum size for the stage, but it should be possible to resize it through expansion both horizontally and vertically - there should be no maximum size. Layout policies should be used to maximize the available space in a sensible fashion.

In particular the list views that display modules on the second and third forms should use any spare (horizontal and/or vertical) available space, allowing more information to be seen at a given time. The same applies to the overview shown on the fourth form with respect to the text areas. On the second form the selected term 1 and term 2 modules should prioritise space over the year long area, for which there are fewer modules. On the fourth form, the selected and reserved text areas should prioritise space, allowing more modules to be viewed.

The screenshots below provide examples of how the application may resize to maximize any available space in accordance with the aforementioned requirements.

