Programming 2
COSC2136 OUA – Study Period 1, 2015

Assignment 2: *AMS -* *Graphical User Interface (GUI)* (25 marks)

This assignment requires you to implement a *Swing* (or AWT) based graphical user interface for the *"Academic Management System (AMS)"* developed in Assignment 1. The interface should provide a graphical means for the user to 'build up' the program structure. Specifically, the user should be able to:

- create a new program and add/remove courses to/from it;
- view the graphical representation of a program structure (all existing courses).

That is, the user interface is **not** required to support **all** the functionality of the AMS system (as per Assignment 1); however, it should allow the user to perform the following functions:

i) Initialise/Reset the Program. i.e. the user should be able to enter all parameters required to instantiate the University with a new Program. The reset function should remove all Courses, if any, from the Program, but preserve the originally entered program code and title.

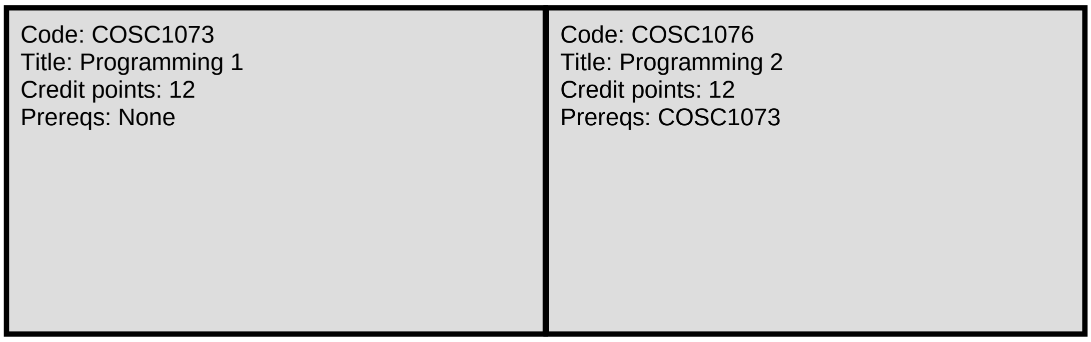
ii) Add a Course to a Program i.e. the user should be able to enter all parameters required to instantiate a new Course, including the course code, title, pre-req courses etc. The system should cover both Core and Elective courses.

iii) Remove a Course from a Program

iv) Display a grid-based view of a Program (a "program map") showing all the existing program Courses.

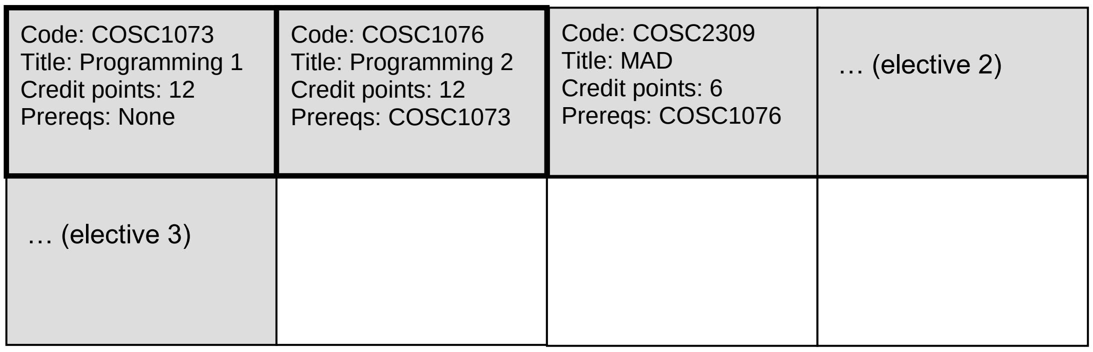
- The program map shall be represented by an equally-sized grid of cells, where each grid cell is sized to take up the maximum amount of available screen area.
- The program map grid can have up to 4 columns and unlimited number of rows, where columns and rows are added/removed on an 'as needed bases' (i.e. dynamically, depending on a total number of courses in the program).
- Cells 'occupied' by a course should have a gray background, and the course details should be shown (i.e. displayed in a dedicated component or drawn in) inside the cell. Also,
    - Cells containing the *Core courses* should have a black border of 5 pixels in thickness.
    - Scrolling functionality should be provided for the individual cells when the course details cannot be fully displayed inside the cell boundaries.
- Unoccupied cells should have a white background.
- There shouldn't be any 'gaps' (unoccupied cells) in-between the occupied cells. i.e. when removing a course from the program map, the existing courses might need to be shifted to cover the resultant 'gap'.
- The user should be able to **sort** the displayed Courses according to the following criteria:
    - NONE (no sorting i.e. original order of insertion) – **default option**
    - TITLE (sorted by the Course title in ascending order)
    - TYPE (sorted by the type of a Course, with all Core Courses displayed first followed by all Elective Courses)

Figures 1-4 illustrate examples of a program map display. Note how the course cell size changes, to take up the maximum amount of screen area, based on the number of courses shown on the map.



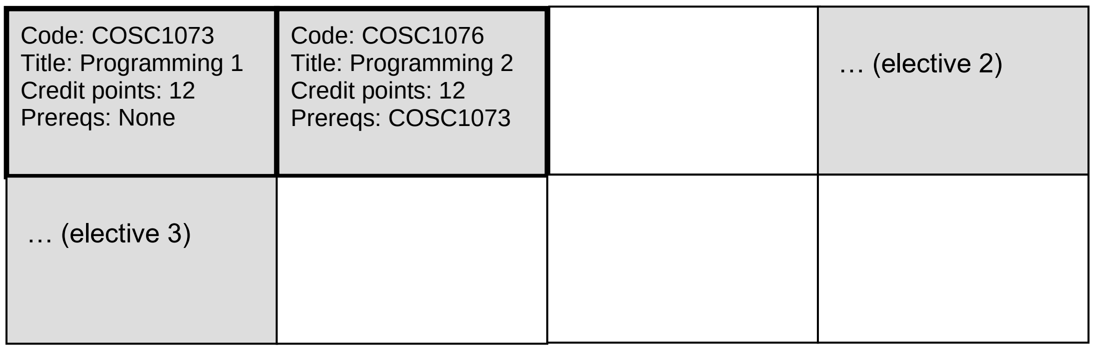
Code: COSC1073 Title: Programming 1 Credit points: 12 Prereqs: None	Code: COSC1076 Title: Programming 2 Credit points: 12 Prereqs: COSC1073

Fig. 1. An example map for a program with two (core) courses



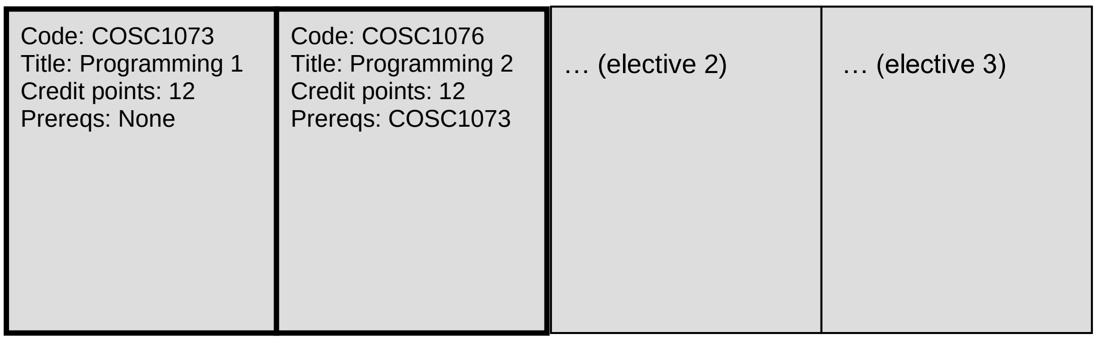
Code: COSC1073 Title: Programming 1 Credit points: 12 Prereqs: None	Code: COSC1076 Title: Programming 2 Credit points: 12 Prereqs: COSC1073	Code: COSC2309 Title: MAD Credit points: 6 Prereqs: COSC1076	… (elective 2)
… (elective 3)			

Fig. 2. An example map for a program with two core and three elective courses



Code: COSC1073 Title: Programming 1 Credit points: 12 Prereqs: None	Code: COSC1076 Title: Programming 2 Credit points: 12 Prereqs: COSC1073		… (elective 2)
… (elective 3)			

Fig. 3. Removing Elective Course COSC2309 - INCORRECT program map display



Code: COSC1073 Title: Programming 1 Credit points: 12 Prereqs: None	Code: COSC1076 Title: Programming 2 Credit points: 12 Prereqs: COSC1073	… (elective 2)	… (elective 3)

Fig. 4. Removing Elective Course COSC2309 - CORRECT program map display

1) Data Requirements

The following data input requirements should be enforced by the system:

- Program:
  code - must be exactly 6 alphanumeric characters;
  title - min length of 2 characters (no character type restrictions).
- Course:
  code - must be exactly 8 alphanumeric characters;
  title - min length of 2 characters (first character must be an upper-case letter).

2) GUI Requirements

At a minimum the user interface should provide:

- **Two sets of action controls**: i) series of buttons; and ii) menu items; with common functions (init and reset the program, and add course and remove course) being present on both control sets.
- **A status bar showing the following information**: i) Program name; ii) Total number of core courses; and iii) Total number of elective courses.
- **Dialog boxes to confirm all important operations**. Specifically: exiting the application (you should create a custom WindowListener to achieve this); resetting the Program; and removing a Course.
- **Dialog boxes to notify the user about any error conditions.** e.g. when invalid data was entered, or when the program structure rule violation occurred.
- **Two different means for the *removal* of holdings**:
  - ***Direct manipulation*** - the user should be able to left click on any occupied cell on the grid and remove the corresponding holding. Note: the user must confirm the removal first.
  - ***Indirect manipulation*** – upon invoking the remove holding button or menu item option, the user should be presented with the list of all currently available holdings. The user should then be able to select any number of holdings from this list and remove them.
- **The main application window can be resized and your application should handle all the program map resizing using appropriate containers and LayoutManagers. Do not use fixed x,y sizing and positioning.**

3) GUI Suggestions

Your graphical user interface should pay attention to factors that increase its usability. Some of the issues you should consider include:

- Choosing GUI elements wisely, ensuring they are appropriate for the task required and that they **minimise user input errors where possible**. For example, to enter pre-req courses for a new course, you should be using a combo box or a list instead of basic text fields.
- Using Layout Managers appropriately to size and position components.

4) <u>Design / Implementation Requirements</u>

- **Your system design must be based on the *<u>Model-View-Controller (MVC)</u>* design pattern**. That is, you should organise all your code, using classes and packages, according to the Model-View-Controller approach.
- **All interaction with the Model component of MVC must be performed via the original AMSModel interface.** This is similar to what you did in Assignment 1 where our TestHarness interfaced with the model (your Ass 1 core system) via this interface.
- A model implementation of the AMS system (Assignment 1) will be made available; you can use this implementation as the Model component in your assignment, but should try using your own Ass 1 solution if possible. Note that your own implementation, if you decide to use it, *must still expose the system functionality via the AMSModel interface* as was done in Assignment 1.

<u>Regulations and Submission Instructions</u>

- Code must be well-structured in terms of the cohesion and coupling. *NOTE: "Rapid Application Development" user interface toolkits or libraries rarely produce well-structured code so their usage is **not** recommended.*
- You are free to refer to textbooks, notes, work in study groups etc. to discover approaches to problems; however, the assignment must be your own individual work.
- Where you do make use of other references, please cite them in your work. Note that you will only be assessed on your own work so the use of third party code is discouraged.

The source code for this assignment (i.e. **Eclipse** project) must be submitted through **Weblearn** by the listed due date (see Blackboard->Course Information->Study Calendar).

You can develop your system using any IDE but will need to create an Eclipse project using your source code files for submission purposes. Please **make sure that you submit the whole system** i.e. the model (Ass1) and GUI (Ass2). Also, **include a short "readme" file** that specifies the location of the driver class (i.e. the class containing the main method), and also includes relevant running/usage instructions, if needed.

Please note that Weblearn only accepts a single file for submission so you should compress your whole project (i.e. the project folder in Eclipse workspace) into a .zip archive and submit the archive.

Late submissions are handled as per usual RMIT regulations - 10% deduction (2.5 marks in Assignment 1) per day. You are only allowed to have 5 late days maximum.