

433-254 Software Design

Project A: University Registration System

Due Date: 5pm Friday, September 13

1 Introduction

MelbX, a famous Australian University, has committed to a new registration system for its students, employees and academic staff. The university would like to keep records for each type of university member in a centralised filing system. As the Dean of administration, and staff, have just taken a course in object oriented design, and Java programming, they would like to implement the system using these latest tools and techniques.

Previously, the university had lost one million dollars, in unrealised consulting fees, by proceeding with a software project without working through a modelling phase beforehand. Determined not to make the mistake twice, you, as project manager, programmer, and general software wiz, have been asked to construct this all-important document. The initial model is to be able to handle the following information:

1. Records for all university members are to include their id number, surname, given name, email address, date of birth, and telephone number.
 - Students, general staff and academic staff each have their own unique ID number: studN (students), staffN (general employee), acadN (academic employee), where N is an integer ($N > 0$).
2. in addition to the attributes mentioned above:
 - Students will also have a list of subjects they are enrolled in. A student cannot be enrolled in any more than 10 subjects.
 - Academic employees will have a salary, and a list of subjects they teach. An academic can teach no more than 3 subjects.
 - General staff will have a salary.
3. A count of how many students, general staff and academic staff, as well as the total of all members for the entire university, in the system, is to be kept.
4. A subject has a course code (eg "433-254"), a name, and a list of no more than 3 pre-requisite subjects (possibly none).

2 Requirements

Then the system is to have the following:

1. Capability to open and read from a file, the list of commands to execute.

2. Capability to cope with ill-formed commands in the file by producing an error message to stdout and continuing with the next command.
3. Capability to add new records to the system (both University members and subjects).
4. Capability to delete a specific record, and all records.
5. Capability to list one or more records, by a given set of attributes for a specified group (ordered by idnum).
6. Capability to edit a specific record.
7. Capability to enrol/unenrol a student in a subject (Max 10). It should report and disallow an attempt to enrol a student in more than 10 subjects.
8. Capability to assign/unassign an academic staff member a subject (Max 3). It should report and disallow an attempt to assign an academic to more than 3 subjects.
9. Capability to set pre-requisites for a given subject (Max 3). It should report and disallow an attempt to set more than 3 prerequisites for a subject.
10. Capability to list prerequisites for a given subject (ordered by course code).
11. Capability to report 'X record not found' where X is some reference to the offending parameter in the command (either a student, employee, or subject). This is for any of the above commands. This includes if a list of students or employees yields no records or if an assignment or enrolment is made for a student or subject that does not exist.

You can assume that:

1. id numbers for University members will follow the conventions specified in this document and so will be well formed and not duplicate an already existing id number.
2. correctly typed data will be given for any field (eg salary will be a number and not a name or email address).
3. the input file containing all commands will exist.
4. keep reading the web page and newsgroup as new assumptions may be added.

The commands for your system will come from a file specified on the command line. The details of these commands are given in the file COMMANDS.txt, linked off the 254 web page. You must adhere to this command format.

The final deliverables will be:

1. A design of the system with use-case diagrams and class diagrams that satisfies the above requirements (submitted as a hardcopy).
2. well documented code that implements your design and follows the guidelines for output (to be specified).

3 Output format

All output should go to System.out (stdout). In the file COMMANDS.txt is a sample of the output for various commands. If you are unsure then please ask Chris McCarthy (cdmcc@cs.mu.oz.au).

Some sample command scripts and output will be posted in the next few days (and as needed) to clarify any issues.

4 Submission

By the deadline, you should submit your design document in hardcopy form to the submission box marked 254 proj A in the basement of SEECs, near the stairs. **Please ensure your name, login and student number are clearly written on the document.**

All your source files (.java files) should be submitted using the submit command:

```
submit 254 A
```

Please make sure that the entry point of your program is in a file called **URS.java**.

ie your program will be invoked with the command

```
java URS <file name>
```

You can submit as many java files as you like but please make sure you include URS.java or else your project will fail our automated testing.

You should then verify your submission using the command:

```
verify 254 A
```

It is important to ensure your system can interpret the commands as set out in the command summary linked off the 254 web page. We will be running your program against automated testing and so expect these commands to be executed.

5 Assessment

Exact details of assessment will be posted soon. In the mean time, be aware that this project is as much about design as implementation so you will be assessed on the quality of your design and how well your code implements your design (does it follow your design and is it correct). All the usual things will also be assessed (Style, Readability, Documentation, Correctness).

All the best!

Chris McCarthy

August 22, 2002