

School Management And Reporting Tool

Due Tuesday, April 13 at 8 a.m.

CSE 1325 - Spring 2021 - Homework #8 / Sprint 4 - 1 - Rev 0

Assignment Background

The School Management And Reporting Tool (SMART) will assist administrators of elementary and secondary schools with keeping track of students and their parents, teachers, course subjects, sections, and grades, and the many relationships between them.

Your goal is to prove that you can implement the SMART (and possibly smart) specification and thus win the contract to build the full production version along with the associated fame and cash and possible spacecraft tickets for Mars on-site support.

This is sprint 4 of a 5-sprint, 5-week project that you can add to your growing resume, with emphasis on expanding the class diagram and adding your first template to the mix.

Your implementation is permitted to vary from any class diagram provided with the final project as needed without penalty, as long as you correctly implement both the letter of and the intent of the feature list. If you prefer to build a tool that addresses a different cultural approach to educating children, that would simply be delightful. **If you have questions about the acceptability of changes that you are considering, contact the professor first!**

Sprint 4 - More Classes and a Template

The primary goal for sprint 4 is add additional supporting classes in preparation for tying the system together in the final sprint. In addition, you'll have the opportunity to write a simple but very useful template.

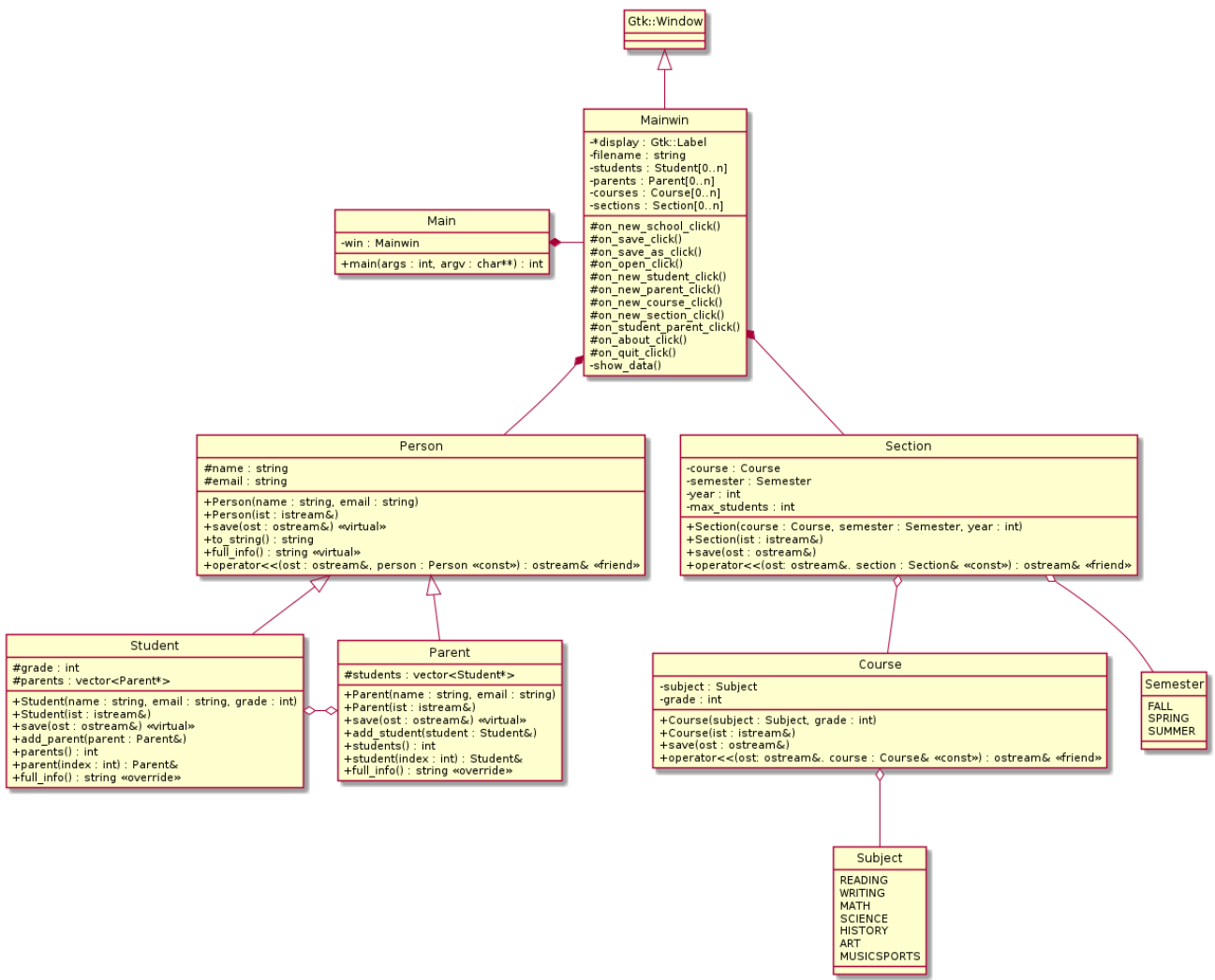
A suggested class diagram for the fourth sprint is on page 2. This diagram is split into multiple diagrams for easier reading, a very common approach as projects grow in size.

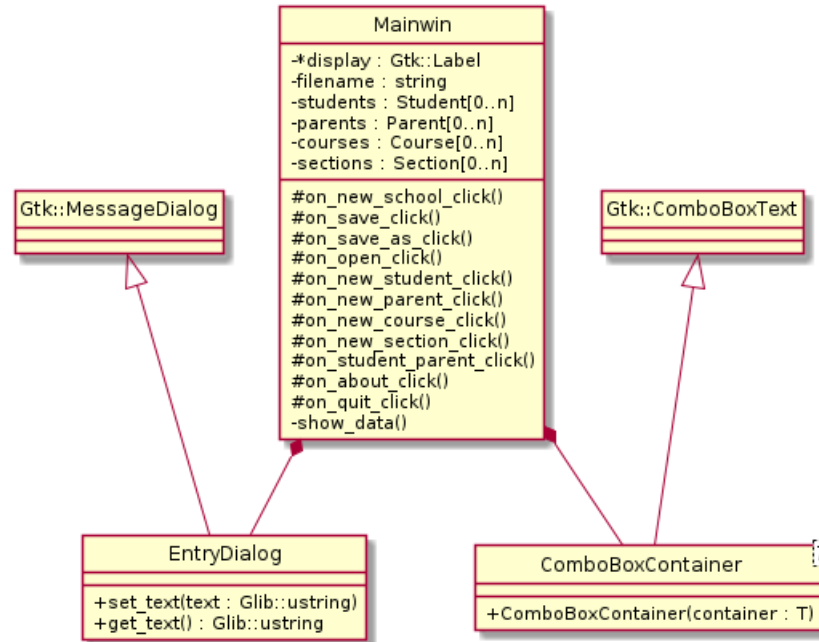
For this sprint, you **MAY** deviate from the class diagram as long as you fulfill the intent of the requirements, including all features allocated to this sprint in Scrum's Product Backlog.

For this sprint, you **MAY** adopt code from the sprint 3 suggested solution with the following caveats:

- Code from any class may be adapted under the terms of the Gnu General Public License 3 (GPL3), which requires attribution among other restrictions. The About dialog added in Sprint 3 is a good way to fulfill these obligations. **Remember to observe these license restrictions if you publish your project code after May 30 as part of your resume.**

Class Overview





EntryDialog, Main, Person, Student, Parent

These classes should require no changes for this sprint.

Semester and Subject

These are enum classes, which should be stored in separate .h and .cpp files. In addition to the enum class declaration, include

- **operator<<** function - Stream out the lower-case word matching the enum element (e.g., for `Subject::READING`, output "reading"). This is used both for human output and for File > Save.
- **load_subject(ist)** and **load_semester(ist)** functions, respectively - These read a single word from the input stream, and return the corresponding Subject or Semester element (respectively). This is used for File > Open.

Course

A course is similar to CSE1325, but in elementary and secondary would look like "English (grade 4)". The data is defined in the constructor, accessed for output via `operator<<`, and saved and restored with the usual `Course(ist)` and `save(ost)` members.

Section

This is similar to Course, but with 4 attributes rather than 2 attributes.

Mainwin

Mainwin adds courses and sections attributes, most likely vectors, and `on_new_course_click()` and `on_new_section_click()` methods in which the data is obtained from the user.

Provide a mechanism to display sections or courses rather than students and parents in the main window display area. The usual convention would be a new menu named View, with View > Courses, View > Sections, View Students, and View Parents. Other acceptable options include Toolbar buttons or tabs.

Also update `Mainwin::on_save_click` and `Mainwin::on_open_click` to handle the new vectors, of course.

ComboBoxContainer

As a first step toward unified dialogs, and also to practice writing a template, derive a new widget named `ComboBoxContainer` from `Gtk::ComboBoxText`. This will be very similar to our `EntryDialog` class, which we derived from `Gtk::MessageDialog`.

For `ComboBoxContainer`, *make it a template* (covered in Lecture 21). Assume the template type is a vector of pointers to any of our classes here - `Student`, `Parent`, `Course`, or `Section` - for which operator<< is defined. Only one member need be defined, the constructor, which will accept as a parameter a vector of pointers such as `Mainwin::students` or `Mainwin::parents`.

In the constructor, iterate over the vector, using an output string stream to convert each object into a string. Append each resulting string to the `ComboBoxText` widget base class using its `append` method. So, basically, `ComboBoxContainer` is a self-populating combo box - pass a vector of pointers in the constructor, and the vector elements are listed in the drop down automatically!

This template will make creating a drop-down list of students, parents, courses, or sections trivial. As a drop-down list of students, for example, you would only need this:

```
ComboBoxContainer<std::vector<Student*>> cbt_students(students);  
dialog->get_vbox()->pack_start(cbt_students);
```

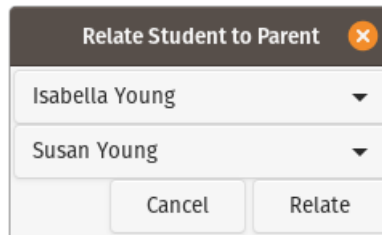
That's it!

To put our new widget to work, improve `Mainwin::on_student_parent_click()` by replacing the sequence of text menu dialogs for selecting a student and parent to relate with a single custom dialog, with a student and a parent drop-down list and "Relate" and "Cancel" buttons. See the example dialog below.

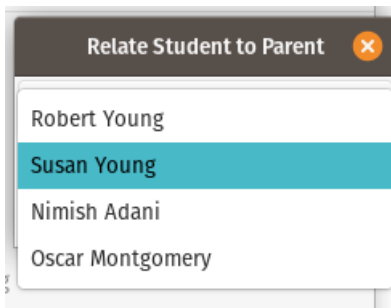
Makefile

You will need to accommodate building the 2 new classes and 2 new enum classes this week. Since the template is only in a header file, it is not represented in the Makefile at all.

Here are some screenshots from the suggested solution. **Your application should look different than these**, since your logo and some icons should be custom. Your application may in fact look much *better* than these. Just ensure you meet all of the required features for this sprint!



A dialog box titled "Relate Student to Parent" with a close button (X) in the top right corner. It contains two dropdown menus. The first dropdown menu shows "Isabella Young" and the second shows "Susan Young". At the bottom, there are two buttons: "Cancel" and "Relate".



A dialog box titled "Relate Student to Parent" with a close button (X) in the top right corner. It contains a list of names: "Robert Young", "Susan Young", "Nimish Adani", and "Oscar Montgomery". "Susan Young" is highlighted with a blue background.

If you have questions about any part of this project, contact me via email or Teams. Ask questions. I **love** questions!