Assignment Tutorial Letter 2024

Advanced Programming

COS3711

Year module

Computer Science Department

Assignment 2 Questions

BARCODE

Assignment 2

1. Introduction

Please use CMake (and not qmake) when setting up your assignment projects.
Qt Designer should not be used to design user interfaces, and you are expected to manually set up GUIs to ensure that you properly handle memory using Qt's parent-child functionality.

Good programming practices should be followed.

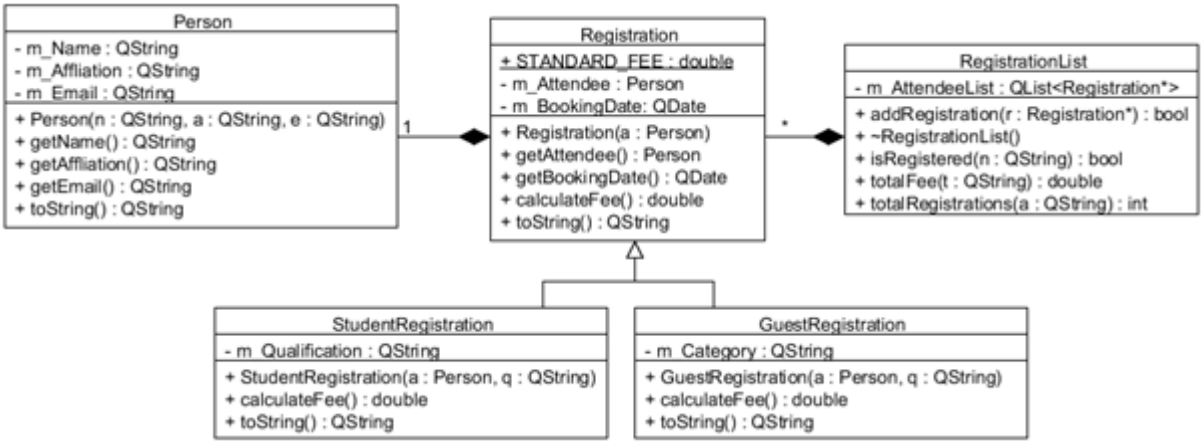
Follow standard naming conventions: class names start with a capital letter, variable and function names start with a lowercase letter, using camelCase for names made up of multiple words.
Ensure consistent code layout and use of blank lines. Use
forward class declarations in header files.
Use initialiser lists in constructors.

Have proper GUI management: setting cursor focus, sequential tabbing, clearing input widgets (like text input fields being cleared and spin boxes being returned to some default value), and enabling and disabling buttons as appropriate.
Provide appropriate user feedback.
Your code should build and run without any warnings.

2. Questions

Question 1

Study the following UML class diagram:



The UML class diagram is an **initial** design of classes for a conference registration application.

There are three types of registration allowed for the conference: standard registration, student registration and invited guests registration. The standard registration fee is the standard fee (stored in STANDARD_FEE). For those registered as students, the conference fee is half the price of the standard registration fee. For those registered as guests, the conference fee is 10% of the standard registration fee.

An instance of `RegistrationList` contains a list of `Registration*`. No two registrations are allowed to have attendees with the same e-mail addresses, unless the names are different. You should be able to query a `RegistrationList` whether a person (by the name) is already registered for the conference. A `RegistrationList` can also return the number of attendees that are registered for the conference from an institution. Similarly, you can request the total registration fees for a type (“Registration”, “StudentRegistration”, “GuestRegistration” or “All”) of registration. You are expected to use Qt’s metaobject system to access the type of a `Registration` in `RegistrationList`.

Write a GUI application that allows users to register for a conference. The GUI should display the current list of registrations, including the respective registration fees, in the `RegistrationList` at all times, using a table widget. The interface should also allow users to request and view the results of the various operations supported by the `RegistrationList` class.

You are allowed to add new features to the classes given in the UML diagram, although you are not allowed to delete or change the given functions. You can decide on the design of the GUI, and it can be done manually or using Qt Designer. You are expected to follow good programming principles.

Question 2

Extend the solution in Question 1, to add another class named `RegistrationListWriter` that produces an XML-formatted `RegistrationList` using DOM as follows:

```
<registrationlist>
    <registration type="Registration">
        <attendee>
         <name>..</name>
         <affliation>..</affliation>
         <email>..</email>
         </attendee>
         <bookingdate>..</bookingdate>
         <registrationfee>..</registrationfee>
    </registration>
      ..
      ..
</registrationlist>
```

Note: `..` means data has to be filled in based on the `Registration` objects in the `RegistrationList`.

Modify the GUI in Question 1, to give the user an option to save the registration list. Allow the user to choose the file, using the standard file dialog, where the XML formatted registration list should be saved.

Question 3

Add another class to Question 2, named `RegistrationListReader`, that can parse the XML formatted asset list using SAX, and create and return a list of `Registration` objects.

To test the `RegistrationListReader` class, modify the GUI in Question 2, to allow the list of `Registration`s to be read in so that they can be appended to the `RegistrationList` maintained by the application. Note that the user can request to read in the XML formatted file when there are `Registration`s, or none, in the `RegistrationList`. Use the standard file dialog to assist the user in choosing the XML formatted file.

Question 4

Implement a factory class (`RegistrationFactory`) using the Factory method design pattern, which creates and returns a `Registration` based on an appropriate string argument and other arguments (for initializing concrete `Registration` objects) provided. Make the `RegistrationFactory` an implementation of the Singleton pattern.

Modify the solution in Question 3, so that the parts of the solution that were creating concrete `Registration` objects (based on the user input and `RegistrationListReader`) are replaced by appropriate code to utilise the `RegistrationFactory`.