ECE 3574 • Applied Software Engineering

Midterm Examination (19 questions, 50 points)

Fall 2008

Name: Student ID#:

Questions 1-15: 2 points each.

Question 1. For overloading arithmetic symbols (+ - \* /) on Fraction objects (see Example 2.14; page 65, Ezust), which is better, member functions or non-member global operators?

[solution] Non-member global operators are preferred because then the first or second operand can be converted just easily to a Fraction. As a member function, only the right hand side is convertible, which means the operator won't work in a symmetric way.

Question 2. Which member functions cannot be inherited from the base class? Explain why.

[solution] The constructors, destructors, copy and assignment operators are all not inherited. They get generated for classes depending on certain rules, and the generated versions call base class versions.

Ouestion 3. What is a design pattern? What do most design patterns have in common?

[solution] Most design patterns describe how to separate code by responsibility. They have a name, a description, and describe a problem (anti-pattern) that it attempts to solve.

Ouestion 4. There are three kinds of design patterns: structural, creational, and behavioral. Which kind is the Visitor pattern? Why?

[solution] Behavioral - it describes how code is organized, and assigns the responsibility of visiting objects to a specific class.

Ouestion 5. Why does the FileVisitor need to be recursive?

[solution] Because hierarchical directory structures are recursive data structures.

Question 6. What does it mean when QObject A is the parent of QObject B?

[solution] B is managed by A, and B will be destroyed when A is.

Ouestion 7. How can QObject be both composite and component?

[solution] When it has both parents and children.

Question 8. Qt l s container classes are used to collect value types. What kinds of things are not appropriate to store by value in a value collection?

[solution] QObjects, polymorphic collections in general, and any class that cannot be copied because it has no copy constructor.

Ouestion 9. Which containers provide a mappings from key to value? List and describe at least two, and tell how they are different from each other.

[solution] QMap and QHash. QMap maintains its values sorted by keys, where QHash does not. Also, QHash provides much faster value lookups than QMap.

Ouestion 10. What is a QLayout? What is its purpose? What is an example of a concrete QLayout class?

[solution] A QLayout is a class whose sole purpose it is to arrange other QWidgets or sub-layouts. It has no visual representation itself. There are Boxes, Grids and Stacks.

Ouestion 11. QWidget is derived from QObject and QPaintDevices. What does this multiple inheritance allow QWidget -derived classes to do?

[solution] They are QObjects, so they can have parents, children, properties, signals, and slots. They are QPaintDevices, so they have a height, width, depth, and can be painted on the screen.

Ouestion 12. What is a QAction? How are actions triggered?

[solution] A QAction is an object representing something that the user can do to the program. It can be triggered through menu items, button clicking, keyboard presses, etc.

Ouestion 13. It is possible to create QMenus without using QActions. What are the advantages of using a QAction?

[solution] By creating a unique QAction for each action, you can add it to different menus and toolbars, and from one place, disable/enable it, or change other properties (such as its label or tooltip) and it will reflect in all views (menu-items, toolbar buttons, etc) that refer to it.

Ouestion 14. What does it mean for a container to "manage" its heap objects? How can a container of pointers to heap objects become a managed container"?

[solution] A container is said to "manage" its heap objects if it is responsible for the lifetimes of those heap objects (i.e., it destroys the objects when it is destroyed). A container of pointers to heap objects can become a managed container by calling qDeleteAll(container), which will call delete on each contained pointer, thereby invoking the destructors of the pointed-to objects.

Ouestion 15. Give 3 examples of Qt classes that implement the Flyweight pattern.

[solution] QList, QString, QSet.

Ouestion 16. What is the Model View pattern? How is it useful? [4 points]

[solution] The Model View pattern is a generic solution to the problem of visually rendering a collection of data items. Instead of combing the code for storing, indexing, and accessing the data with that for visually rendering them, the pattern uses different classes for doing this. The Model class provides an interface for accessing the data and the View class visually renders that data. The pattern thus allows decoupling of the model and the view - i.e., it allows one to use different View classes (for obtaining different views) for a given model, and different Model classes (for storing and accessing data in different ways) for a given view.

Question 17. See an example use of the QStringListModel given in

Example 11.24 of Ezust (page 272-273). If you didn't use the Model

View pattern and Qt classes that support this pattern (i.e.,

QStringListModel, QListView) to write this application, how would you then write this application? Explain the general steps that you would take, the Qt classes that you will use, the order in which you will use the APIs, and any possible signal/slot connections that will be needed. [5 points]

There are several "low-level" ways to do this. An example approach would be:

Use a container class (e.g., QHash, QMap) to store the QStrings ("Mandaag .. ' ). For rendering this data, use a QApplication, define a central QWidget for it, and set up a parent QLayout (e.g., QGridLayout) for the widget. To the parent QLayout, add a QVBoxLayout (for rendering the QStrings "Mandaag...") and the "insert" QPushButton widget (for rendering the output QStrings "item #....").

Since the QStrings ("Mandaag...") must be rendered and editable, use

an input widget such as a QLineEdit. Retrieve each of the QString from the container (using an iterator pattern with keys(), values(), etc.), insert it into a QLineEdit (using insert() or setText()), and add the QLineEdit as a widget to the QVBoxLayout (note that QLineEdit is a QWidget). This will display each of the QStrings in the displayed window.

Connect the textChanged() signal of QLineEdit to a slot, which will insert the signal's returned QString parameter into the container class. (textChanged() signal is emitted whenever the text changes and the user-entered text is returned as the signal's parameter.)

Connect the "insert" QPushButton's clicked() signal to the slot addltem() as before. But this slot will now create the QString "item #. . insert it into a QLineEdit, and add the QLineEdit as a widget to the QVBoxLayout. It will also add the QString to the container class.

Ouestion 18. What is the difference between aggregation and composition relationships? [3 points]

[solution] In an aggregate relationship, the lifetimes of the objects on either end are not related to each other. In contrast, in a composition relationship, the "whole" object is responsible for the lifetime of the part" object.

Question 19. Draw a UML diagram with three or more classes, different from all the examples given in Ezust, and make sure that all of these different kinds of relationships are represented:

aggregation composition  one-to-one and one-to-many unidirectional

The classes should represent real-world concepts, and the relationships should attempt to represent reality. Justify why there is a relationship of each kind in your diagram. [8 points]