ECE 3574 Applied Software Engineering

Midterm Examination Solutions (16 questions, 50 points)

Fall 2010

Name: Student ID#: Questions 1-15: 3 points each. Use your own words in your answers.

Question 1. Consider the function call graduate( ) in Example 6.5 of Ezust (page 141). Why is that (a) this call prints [ Student] and not [GradStudent] and (b) this call does not print GradStudent's support (Support) information?

[solution]

(a) getC1assName( ) is not defined as a virtual function in the class Student . Thus, the call graduate ( ) will invoke Student's getClassName(), which returns the string "Student". If getC1assName( ) was defined as virtual in class Student, then graduate( &gs) will invoke GradStudent's getC1assName( ) , which returns the string "GradStudent", as result of dynamic binding.

(b) tostring( ) is not defined asa virtual function in the class Student. Thus, the call graduate( ) will invoke Student's tostring( ) function, which doesn't print any Support information. In contrast, GradStudent's tostring( ) function prints the Support information.

Question 2. What is an abstract class? What is it used for? What can you do with a concrete class that you cannot do with an abstract class?

[solution] An abstract class contains at least one pure virtual function, and cannot be instantiated like a concrete class. An abstract class is used to group features that are common to many classes in a design, but the class by itself doesn't have a real world physical counterpart.

Question 3. For each of these items, determine whether it would normally be found in a header (.h) file or an implementation (.cpp) file and explain why.

(a) function definitions

[solution] implementation - they should be only compiled once, and not included by other modules that use it.

(b) function declarations

[solution] usually the header file, to refer to functions defined in other separate implementation files.

(c) static object declarations

[solution] header files, like most declarations.

(d) static object definitions

[solution] implementation file, just like functions. We do not wish to have redundant storage allocation for statics.

(e) class definitions

[solution] these go in the header file.

(t) class declarations

[solution] header file - usually they are forward declarations.

(g) inline function definitions

[solution] header files - unlike regular functions which can be linked together, inline functions need to be fully defined before they can be used.

(h) inline function declarations

[solution] Header file - Rarely used for member functions except when the declaration is separated from the definition - but kept in the same header file.

(i) default argument specifiers

[solution] header file - default arguments to a function changes the way the function is called. This is the realm of the header file.

Question 4. 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.

Question 5. What happens to a QObject when it is reparented?

[solution] It is removed from its former parent's child list, and added to the new parent's child list.

Question 6. Why is the copy constructor of QObject not public?

[solution] To prevent it from being copied. QObjects are supposed to have identity, and so there should be no "exact" copy. In addition, copying a QObject would mean copying all its children, a potentially expensive operation.

Question 7. How can you access the children of a QObject?

[solution] The children of a QObject can be obtained through the findChildren() member function.

Question 8. What is the difference between value types and object types? Give examples.

[solution] Value types are simple types that can be copied or compared quickly—

e.g., int, char, QString, QDate, pointer types, etc. In contrast, object types are complex, maintain an identity, and can rarely be copied—e.g., QObject.

Question 9. When connecting a signal to a slot, are there any restrictions on the parameters of the signal and the slot?

[solution] A signal of one object can be connected to the slots of one or more other objects, provided the parameter lists are assignment compatible from the signal to the slot. That is, the slot must have at least as many parameters as the signal (the slot can ignore extra parameters), and those parameters must be value types.

Question 10. How can a container of pointers to heap objects become a "managed container"?

[solution] There are several ways to do this. One way is for the container to call the Qt algorithm qDe1eteA11(container) . This calls delete on each (pointer) element in the container, which invokes the destructors of the heap-allocated pointed-to objects and destroys them.

Question 11. What is the purpose of the QLayout class? What is an example of a concrete QLayout class?

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

Question 12. Which containers provide a mapping 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 look-ups than QMap.

Question 13. What is the Flyweight pattern? Give 3 examples of Qt classes that implement the Flyweight pattern.

[solution] The Flyweight pattern is a generic solution to the problem of avoiding creating multiple copies of an object, when a single shared instance of the object will do. The pattern uses a handler class that manages a single shared instance of the object and keeps track of references to the shared instance through a reference counter. The counter is incremented when new references to the shared instance

are added, and decremented when references are deleted. When the counter reaches zero, the handler destroys the shared object.

Qt example implementations of this pattern include QList, QString, and QSet.

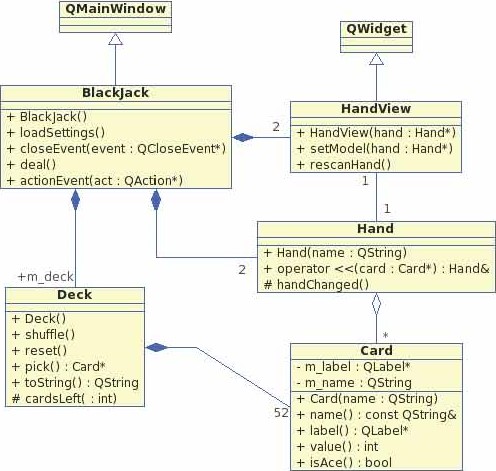
Question 14. What is the difference between a spacer and a stretch?

[solution] A spacer is a fixed amount of space that does not stretch, while a stretch starts with a certain amount of space and expands to gobble up available space. In the case of multiple stretches in the same layout, the ratio can be used to determine which stretch expands faster.

Question 15. What is the Model View pattern? How is it useful?

[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 16. Consider the following UML diaglam:



Explain this diagram by identi&ing the different types of class relationships (i.e., which class is related to which class and how) and the multiplicities in the relationships.

This design follows the model/view paradigm Identi& the model class(es) and the view class(es). [5 points]

[solution]

Class relationships:

 Class BlackJack is derived from class QMainWindow

Class HandView is derived from class QWidget

Class Blackjack is composed of Wo Hand View objects

Class Blackjack is composed of two Hand Objects

 Class Blacl$ack is composed of one Deck object

Class Hand View and class Hand has a 1:1 bidirectional association relationship between them

Class Hand is an aggregate of zero or more instances of Card objects

Class Deck is composed of fifty two Card objects

Model/view classes:

 Class HandView is the view class Class Hand is the model class