ECE 3574 ♦ Applied Software Design
Midterm Examination Solutions (10 questions, 50 points, 75 minutes)
Spring 2016

Name: **SOLUTION** Student ID#:

Use your own words in your answers. Be brief and to the point.

Question 1. What is an abstract class? (What can you do with a concrete class that you cannot do with an abstract class?) In which ways an abstract class can be defined in C++? (Give at least two examples.) [6 points]

[solution] An abstract class 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. Abstract class can be defined by either having at least one pure virtual function or having no public constructors.

Question 2. What member functions cannot be inherited from a base class? Explain why. [4 points]

[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.

Question 3. In the context of container classes, explain the differences between aggregate and composite relations. [4 points]

[solution] An aggregate container is a container that provides only an indexing or reference navigation mechanism to its contents. When an aggregate container is copied, only references to the collected objects are copied. When an aggregate container is deleted, only the references are removed. There is no impact on the underlying objects in the container.

A composite container manages its pointed-to objects. In other words, when a composite is destroyed, it destroys (cleans up) its entire self because the smaller objects are part of its composition.

Question 4. In the Qt programming framework, what are the differences between **value types** and **object types**? Mention at least two differences. For each category, indicate some classes from which you can inherit (include the top-level ones). As well, indicate which meta type subsystem each category is associated with. [6 points]

Instances of value types are usually relatively simple, occupy contiguous memory space, and can be copied or compared quickly. Examples of value types: Anything * , int , char , QString , QDate , and QVariant. Any class that has a public default constructor, copy constructor, and copy assignment operator is a value type. Instances of object types, on the other hand, are typically more complex and maintain some sort of identity. Object types are rarely copied (cloned). If cloning is permitted, the operation is usually expensive and results in a new object (graph) that has a separate identity from the original. Value types are associated to the QMetaType subsystem and they can be introspected via QVariant. Object types should inherit from QObject and they are part of the MetaObject subsystem.

Question 5. What is a design anti-pattern? (Give an example.) [3 points]

[solution] AntiPattern is a term to describe a commonly used programming practice that has proved to be ineffective, inefficient, or otherwise counterproductive.
Example: Copy and paste programming— Copying and modifying existing code without creating more generic solutions. More examples in Section 7.4.2 of the textbook.

Question 6. Consider the function call `finish(&us)` in Example 6.5 of Ezust (page 175). Why is that (a) this call prints `[Student]` and not `[Undergrad]` and (b) this call does not print `Student`'s Scholastic Aptitude Test score (SAT) information? [6 points]

[solution] The method "finish" takes a pointer of Student as an argument. Thus, any subclass passed is pointed to by the base class pointer.

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

(b) toString() is not defined as a virtual function in the class Student. Thus, the call finish(&us) will invoke Student's toString() function, which doesn't print any Support information. In contrast, Undergrad's toString() function prints the Support information.

Question 7. Consider the line of code `namelist << s5 << s1 << s3 << s4 << s2;` in Example 11.6 of Ezust (page 363). What happens to the output of the application if we change the line to `namelist << s4 << s2 << s3 << s5 << s1;` ? [3 points]

[Solution]
s1("Apple"), s2("bear"),s3 ("CaT"), s4("dog"), s5 ("Dog")
qSort uses QuickSort Algorithm.
So, the list before sorting has: dog << bear << CaT << Dog << Apple. When the algorithm is applied, Apple is swapped with Dog. Thus:

"namelist[0] = Apple"
"namelist[1] = bear"
"namelist[2] = CaT"
"namelist[3] = Dog"
"namelist[4] = dog"

Question 8. Consider the line of code `double d(2. / 3.), e (d / 2);` in Example 8.9 of Ezust (page 280). What happens to the output of the application (Example 8.10 of Ezust, page 281) if we change the line to `double d(2. / 3.), e (d / 3);` ? (The question is not asking you to write the exact output of the application but to comment about the new behavior.) [4 points]

[solution] QVERIFY (d == e*2);

```
This line will fail.
```

Question 9. Consider the class definitions and driver code below and answer the following questions. [6 points]

```
class Shape {
public:
    virtual void draw(Position p);
    virtual void draw(PaintEngine* pe, Position p);
};
class Rectangle : public Shape {
public:
    void draw(Position p);
private:
    void draw(int x, int y);
};
int main() {
    Position p(4,3);
    Position q(5,6);
    PaintEngine *pe = .....;
    Rectangle rect;

    rect.draw(p);           1
    rect.draw(pe, p);       2
    rect.draw(3,3);         3

    Shape* sp = &rect;
    sp->draw(q);          4
    sp->draw(pe, q);      5
    sp->draw(3,2);        6
}
```

1. Which method is called for the line labeled **1**?
   a. Shape::draw()
   b. Rectangle::draw()
   c. error - method is hidden
   d. error - method is inaccessible
   e. error - no such method

2. Which method is called for the line labeled **2**?
   a. Shape::draw()
   b. Rectangle::draw()
   c. error - method is hidden
   d. error - method is inaccessible
   e. error - no such method

3. Which method is called for the line labeled **3**?
   a. Shape::draw()
   b. Rectangle::draw()
   c. error - method is hidden
   d. error - method is inaccessible
   e. error - no such method

4. Which method is called for the line labeled **4**?
   a. Shape::draw()
   b. Rectangle::draw()
   c. error - method is hidden
   d. error - method is inaccessible
   e. error - no such method

5. Which method is called for the line labeled **5**?
   a. Shape::draw()
   b. Rectangle::draw()
   c. error - method is hidden
   d. error - method is inaccessible
   e. error - no such method

6. Which method is called for the line labeled **6**?
   a. Shape::draw()
   b. Rectangle::draw()
   c. error - method is hidden
   d. error - method is inaccessible
   e. error - no such method

Question 10. Consider the class definitions and driver code below and answer the following questions. [8 points]

```
#include <QDebug>
class Base {
public:
    Base() { ++sm_bases; }
    ~Base() { --sm_bases; }
    void a(){ qDebug() << "Base::a()"; }
    virtual void b(){ qDebug() << "Base::b()"; }
protected:
    static int sm_bases;
};
class Derived : public Base {
public:
    Derived() { ++sm_deriveds; }
    ~Derived() { --sm_deriveds; }
    void a() { qDebug() << "Derived::a()"; }
    void b() { qDebug() << "Derived::b()" ;}
    static void showCounts() {
        qDebug() << sm_bases << sm_deriveds;
    }
protected:
    static int sm_deriveds;
};
int Base::sm_bases(0);
int Derived::sm_deriveds(0);

void foo() {
    Derived d1;
    Base b1;
    Base* bp = new Derived();
    bp->a();
    bp->b();
    delete bp;
    Derived::showCounts();       1
}
int main() {
    Base b;
    Derived d;
    foo();
    Derived::showCounts();       2
}
```

1. What is the output from the first call to `showCounts()`?

    [✓ **a. 4 3**]
    b. 3 2
    c. 2 2
    d. 1 1
    e. 3 4

2. What is the output of the second call to `showCounts()`?

    a. 1 2
    b. 1 1
    c. 0 0
    [✓ **d. 2 2**]
    e. 2 1