Subject: Re: Difficulty with Class declaration
Posted by brokndodge on Thu, 01 Apr 2010 04:19:09 GMT
View Forum Message <> Reply to Message

So if I understand you correctly, UDMS can't access Popups, but Popups CAN access UDMS. Did I get that right?

So in order to get access to Popups, it needs to inherit from a class that UDMS inherits from. Possibly Upp? Or is my thinking twisted. Your analogy would indicate that BallWithSquare can access any class above it, but that Ball CANNOT access Square directly. So I have a menu function in main.cpp that needs access to Popups. I would actually have to define

```
class MenuOnMenuBar : Popups, UDMS {
    void menu ();
    Popups pop;
};
```

in order for menu to call any function in Popups. But then how does UDMS access MenuOnMenuBar? I'm missing something.

The normal hierarchy I am accustomed to is that main sets up the basic structure of my app. Then every other function comes under main. With this description of classes, my lowest level function would be in my top level class and everything would derive from it. That's completely backwards.

Maybe I shouldn't be using class at all. Can I have just one class UDMS and define everything in that one class? That means just one header file for the entire project.

My original blueprint calls for

```
class UDMS : TopWindow{}; //main class, creates topwindow, menubar and
                          //TabBar, calls Popups::Login then passes the rest
                          //of the work off to other modules.
                          // I had some difficulty with my initial attempt
                          // to call Sales from UDMS so I put together a
                          // simple popup in an effort to learn how all
                          // of this will work together.

class Popups : UDMS {}; // every class will share
                        // many common popups
class Sales : UDMS, Popups{};
class Inventory : Sales, Popups{};
class Finance : Sales, Inventory, Popups {};

class Service : UDMS, Popups{};
class Mechanics : Service, Popups{};
class Parts : Service, Popups{};
class Cashier : Mechanics, Service, Popups{};
```

```
class Reception : Sales, Service, Popups {};
class Accounting : Sales, Service, Popups {};
class Admin : Sales, Service, Reception, Accounting, Popups {};
```

With the reasoning described in your analogy I have the entire sketch backwards. I understand that since UDMS needs to call each of these modules, UDMS would have to derive from each of them. So defining each module in a class is just not the way to handle this project: or is there a way to achieve my goal thru pointers?

I'm going to go back to the tutorials and try to find a better way to do this.

Thanks each and every one of you for the information you gave me.
