

Name:

Sec:

07/11/2006

Lab 5.1 Defining and Using Classes

Recall that the two structured data types we have studied so far are arrays, which have elements all of the same type, and structures, which have elements of differing types. Another structured data type is a class, which is specifically designed to combine data and functions in a single unit. A class is a collection of a fixed number of components. The components of a class are called the members of the class. If a member of a class is a variable, you define it just like any other variable, but you cannot initialize it at definition. If a member of a class is a function, you typically use the function prototype to define that member. Member functions can directly access any data member of the class without passing that data member as an argument.

No memory is allocated in a class definition. Instead, memory is allocated when the class is instantiated (an object is created). Additionally, the semicolon (;) is part of the syntax at the end of the class definition. The members of a class are classified into three categories: private, public, and protected. By default, all members of a class are private. Private members cannot be accessed outside of the class. A public member is accessible outside the class. Protected members will be discussed in Chapter 13.

Objectives

In this lab, you define a class and declare objects.

After completing this lab, you will be able to:

- Write a driver program that contains a class definition.
- Instantiate an object of a class.
- Practice using coding projects and working with a program defined across multiple files.

Instructions

1. We will first create a simple structure called Money. We will be using multiple files in this lab to build a program. So, you should start off by creating a project to hold all of your files. Name the project Lab5.
2. Create a new file called Money.h and add it to your Lab5 project. Define a struct, called Money, in this file. The Money structure should have only 2 member items. One member is a char data type and should be called **type**. The other is of type double and should be called **amount**. Since this is a structure, both of these members are publically accessible by default. The type member will hold the type of the monetary value being represented, for example Mexican Pesos, or Russian Rubbles, or U.S. Dollars, using a

single character to represent the monetary type. The amount will be, as suggested by the name, the amount of Pesos, Rubles, Dollars, etc respectively being saved in the structure.

3. Create a new source file and name it TestExchange.cpp. Add a single main() function to this file. Include your Money.h header file by doing a:

```
#include "Money.h".
```

You need to use the parentheses (instead of the <>) for a header file that you define. Test your Money class by creating a few variables of type Money. For example, do the following, and make sure your TextExchange.cpp and Money.h files compile correctly:

```
Money dollars;  
Money pesos;  
Money francs;  
  
dollars.type='d';  
dollars.amount=55.23;  
pesos.type='p';  
pesos.amount=1000.00;  
francs.type='f',  
francs.amount=15.35;
```

4. Now we will implement a full blown class, that has both data members as well as function members. Create a new file called Exchange.h and add it to your project. Inside of this file we will create a class called Exchange. The purpose of this class is to calculate an exchange of Money from one type of currency into U.S. dollars, for example from Pesos to U.S. dollars.
5. Add 2 private data members to your Exchange class. Both of these members will be of type Money (defined in step 3), so you will need to #include "Money.h" at the top of your "Exchange.h" header file. Make sure the members are private by using the private: keyword before the definition of the member items.
6. We will add 3 public function members to the Exchange class next. One function should be called getData() and it should return a bool as its result. Another function should be called displayExchange(), and it returns no result so it is a void function. The third function is called convert() , and it is also a void function that takes no parameters as input.
7. At this point you have defined a class called Exchange in the Exchange.h header file. We now need to implement the 3 member functions that you declared as being members of the Exchange class. We will code the implementations of these member functions in a file called Exchange.cpp. Create this file and add it to your project. At the top of the file you will

need to #include "Exchange.h" in order to pull in the definitions of the Exchange class from the header files.

8. First implement the getData() member function. Remember, this is a member function of the Exchange class. Therefore, when you create the function you need to declare it as being inside of the Exchange class, like this:

```
bool Exchange::getData()
{
    // implementation of getData here
}
```

This says that the getData() function is a member function of the Exchange class, since we put the Exchange:: in front of the name of the function. This also says that getData() takes no parameters as input, and returns a bool as its result, as we specified in step 6 when declaring this member function. The member function named getData() should prompt the user for the type of currency: (p) for Mexican pesos, (d) for U.S. dollars, (f) for Swiss Francs, (e) for Euro dollars, and (q) for quit, and then input the character into the data member entered.type. Remember that our Exchange class has 2 data members, US and entered. We are inputting a single character and saving it in the entered.type data member. If the user enters a 'q', then false is returned by the getData() function. If a valid character is entered, the program should prompt the user for an amount, and inputs that amount into the data member entered.amount. The getData() function should return true then once a valid currency type and amount have been input into the entered data member.

9. Now implement the convert() member function. Again remember that convert() is a member of your Exchange:: class, so you need to specify that when you implement the function. The member function named convert() assigns the exchange amount of the value entered by the user (in the getData() function) to the US Money member data type. For example, if the user enters 20 pesos as the entered.amount and 'p' for pesos as the entered.type, the job of the convert() function is to convert the 20 pesos into US dollars, and place the amount in US.amount (and put a 'd' for dollars in the US.type). The conversion rates are 1 U.S. dollar = 0.9553 Euro dollars = 1.4054 Swiss francs = 9.815 Mexican pesos. For example 20 Mexican pesos is about 2.04 US dollars.
10. Finally implement the display() member function of the Exchange:: class. This function should simply display a message on the console such as:

20 Mexican Pesos = \$2.04 U.S. Dollars.

You will use the entered and US member data items to create your display.

11. As a last step, use this main loop to test your Exchange and Money classes/structures:

```
int main()
{
    Exchange e;
    while (e.getData())
    {
        e.convert();
        e.display();
    }
}
```

This will keep asking the user for a monetary amount and currency type, then converting it and displaying the conversion. Note that you need to return false from your `getData()` member function if the user enters 'q' so that the program knows that the user is done entering monetary amounts.

Lab 5 Finished

You have now completed Lab 5. If your program compiles and runs correctly and you have successfully uploaded your source file to the eCollege online submission site, then you are done.