

Lab 4.1 Defining and Using Classes

Recall that two structured data types are arrays, which have elements 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. 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.

A class and its members can be described graphically using Unified Modeling Language (UML) notation. A UML diagram contains three boxes stacked vertically. The top box contains the name of the class; the middle box contains the data members and their data types; and the bottom box contains the member method names, parameter list, and return types. The + (plus) sign indicates that it is a public member; the - (minus) sign indicates that it is a private member. The # (pound) symbol indicates that it is a protected member.

Objectives

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

After completing this lab, you will be able to:

- ▶ Use basic UML notation to design and specify a class.
- ▶ 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.

Describing Classes Using the Unified Modeling Language (UML) Notation and Declaring Objects

- 1a. Create a UML diagram for a struct named Money that will contain data members. Your struct (or class) should contain a char data member named type and a double data member named amount as public members. (See Ch #12, pg. 599 for a quick discussion and example of UML diagrams).

- 1b. Implement the struct in your diagram in C++ and save your code in a file named **Money.h**.

- 2a. Create a UML diagram for a class named **Exchange** that will contain data members and member functions to allow a user to input, validate, and convert pesos, Euro dollars, and Swiss francs to U. S. dollars. Your class should meet the criteria in the following list:★
 - ▶ The class has 2 data members, **US** and **entered**, which will be of the type **Money** that you defined in Exercise 7, and should be private members.
 - ▶ The Boolean function **getData** and the void functions **convert** and **displayExchange** have no formal parameters and should be public members.
 - ▶ The void function **outputType** has no formal parameters and is a private member.

- 2b. Implement the design and member functions of your diagram in C++ and save your code in 2 files. One file named **Exchange.h** should contain the definition of your Exchange class. The other file, named **Exchange.cpp**, should contain the implementation of the function members of the Exchange class. You will need to *#include "Money.h"* in your **Exchange.h** file. Likewise you will need a *#include "Exchange.h"* at the top of your **Exchange.cpp** file.
- 2c. Implement the member functions so that they meet the following criteria:
- ▶ 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`. If a 'q' is entered, the value `false` is returned. If a valid character is entered, the program prompts the user for an amount, and inputs that amount into the data member `entered.amount`. Use a loop in the function until the user enters q to quit or enters a valid type.
 - ▶ The member function named `displayExchange` displays the type of currency and the beginning amount and the amount in U. S. dollars. The amounts are displayed in decimal places. The member function `outputType` is called from `displayExchange`.
 - ▶ The member function named `outputType` displays a description for the character type of currency; for example, 'd' would display "U. S. dollars".
 - ▶ The member function named `convert` assigns the exchange amount of the value entered in U. S. dollars to `US.amount` and assigns the character type of `entered.type` to `US.type`. The conversions are: 1 U. S. dollar = 0.9553 Euro dollars = 1.4054 Swiss francs = 9.815 Mexican pesos
- 2d. Since you are now writing a program that contains multiple files (3 so far, **Money.h**, **Exchange.h** and **Exchange.cpp**) you MUST begin using projects in your IDE (Dev or Visual C++ compilers). If you haven't already, create a new project in your development environment (ConvertMoney would be a good name for the project). Add the 3 files you have created so far to your project.
- 3a. Now we will create what is known as a driver program that will use and test your Exchange class. Design a driver program that *#include "Exchange.h"* the Exchange class that you created in step 2. Name the program **TestExchange.cpp** (and add it to your project). This file will contain your main function for the program you are writing. Write a loop that creates an Exchange object and uses the `getData()` member. The loop should continue until the returned value from `getData()` is `false`. After calling `getData()`, within the loop you should then call the member functions `convert()` and `displayExchange()`.

3b. Enter, compile, link, and execute **TestExchange.cpp**.

The following is a copy of the screen results that might appear after running your program, depending on the data entered. The input entered by the user is shown in bold.

This program prompts the user to enter a character to designate the type of currency to be exchanged and the amount of money to be exchanged.

Enter a character for currency type and an amount.
(P)esos, Swiss (f)francs, (E)uro dollars or (q)uit: **x 123.543**
You entered an invalid type, please re-enter.
p 123.543
You entered 123.54 in Pesos, which is \$12.59 in U.S. dollars.

Enter a character for currency type and an amount.
(P)esos, Swiss (f)francs, (E)uro dollars or (q)uit: **f 500**
You entered 500.00 in Swiss francs, which is \$355.77 in U.S. dollars.

Enter a character for currency type and an amount.
(P)esos, Swiss (f)francs, (E)uro dollars or (q)uit: **e 987**
You entered 987.00 in Euro dollars, which is \$1033.18 in U.S. dollars.

Enter a character for currency type and an amount.
(P)esos, Swiss (f)francs, (E)uro dollars or (q)uit: **q**

Lab 4 Finished

You have now completed Lab 4. Turn in the previous pages to the instructor and make sure you have uploaded your **TestExchange.cpp**, **Exchange.h**, **Exchange.cpp** and **Money.h** in a new Lab4 folder in your online student account.

Attached at the end of this lab is a description of your third programming assignment, which will be due next Friday Oct 21. After completing this lab, now is a good time to begin thinking about how you will implement the second programming assignment.