

Register Entry Inheritance Hierarchy

Purpose:

This exercise is intended to give you experience with building small inheritance hierarchies, and seeing the benefits of programming by differences.

Your test program will give you an opportunity to see that, in fact, virtual behavior means the functions really don't know what function to call (which message invokes which method) until runtime.

What to hand in:

Hand in the header(s), implementation(s) and test program(s), well-formatted and understandable, and the output from a sample run. Ensure that your test program *thoroughly* tests the required class, and your output shows the results of the tests. *Note that the withdrawal, interest, and fee entries can be omitted. You may also omit any header or implementation files drawn directly from the SamplePrograms provided.* The hierarchy, models and sample code are provided as part of the resources for the class.

This register entry hierarchy will comprise one or more header files describing the hierarchy of register classes and one or more implementation files providing the behavior.

Please capture your homework in a single file for submission, in an easily reviewable order.

Problems:

Create a class hierarchy for register lines. Remember that there are four different concrete subtypes to build: check entry, deposit entry, error entry, and reconciliation entry. The four abstract classes in the hierarchy are RegisterLine, Entry, OwnerEntry and AdjustEntry.

Consider carefully what the common interfaces and structure are, and how your class can properly represent that. Be especially careful in applying the virtual keyword appropriately.

Implement a test program to test the class. Please remember that it is not fair to add special test interfaces to the class; your test program should use the class just as your final project will.

You should use a Money class to implement the amount part of the entries, and the Standard C++ string class to implement the name and/or reason parts of the entry.

Notes:

Don't forget proper constructors, and other support functions.

I suggest using a base class pointer in your test program, to hold a pointer to any derived class. That is:

```
RegisterLine *r = new DepositEntry (...);
```

This will provide a reasonable way to test your use of the virtual keyword (Why?).

Use the design a little, code a little, test a little paradigm to implement this—if instead you were to complete the .h files, then the implementation files, and then the test file, it would be quite a long time before you would get anything to run.

If, instead, you code the RegisterLine and ReconciliationEntry classes, implement constructors, destructors, and the Print function, you can build a small, extensible test program right away. You can even start off making classes *concrete*, and make them *abstract* later.

Evaluation criteria:

RegisterLine, Entry, OwnerEntry, AdjustEntry, DepositEntry, InterestEntry, WithdrawalEntry, and ReconciliationEntry class interfaces (.h file(s))	35%
Class implementations (.cpp file(s))	35%
Test program(s) (.cpp file(s)) and output	30%