Class "Header" File: `Wallet.h`

```
/**
 * A wallet class
 */
#pragma once

#include <iostream>
#include <string>
using namespace std;

/**
 * Wallet interface
 */
class Wallet {
public:
    // default ctor makes an empty, anonymous wallet
    Wallet( );
    Wallet( const string& theId);
    Wallet( const string& theId,
            int theDollars, int thePennies );
    // retrieve the wallet identification
    string identification( ) const;
    // retrieve the monetary value
    double getCash( ) const;
    // track expenses
    void expense( int costDollars, int costPennies );
    void expense( double costAmount );
    // track income
    void income( int earnDollars, int earnPennies );
    void income( double earnAmount );
    // read value from the input stream --
    // returns a reference to the istream argument provided
    istream& input( istream& is );
    // print myself to the output stream
    // returns a reference to the ostream argument provided
    ostream& output( ostream& os ) const;
private:
    // makes sure monetary amounts are always positive,
    // carries >= 100 pennies to dollars
    // displays a message to cout if amount is negative
    bool money_logic( int theDollars, int thePennies ) const;
    // makes sure monetary amounts are positive
    // displays a message to cout if amount is negative
    bool money_logic( const double theAmount ) const;
    // converts dollars and pennies to a decimal dollar amount
    double dollars_pennies_to_value( int theDollars,
            int thePennies ) const ;

    // data members
    string id; // owner name
    double cash; // amount of money in wallet, >=0
};
```

Class Implementation File: `Wallet.cxx`

```
/**
 * A Wallet class
 */

#include <iostream>     // system headers first
#include <string>
#include "Wallet.h"    // developer headers second
#include "string_substitute.h"   // replace whitespace with _
using namespace std;

/**
 * Wallet implmplementation
 */

// default ctor makes a anonymous, empty wallet
Wallet::Wallet( )
{
    id = "???"; // anonymous marker
    cash = 0;
}
```

Class Implementation File: `Wallet.cxx` continued...

```
22 // specifies name only
23 Wallet::Wallet( const string& theId )
24 {
25     id = string_substitute( theId, "␣\n\t", '_' );
26     cash = 0;
27 }
28
29 Wallet::Wallet( const string& theId,
30         int theDollars, int thePennies )
31 {
32     id = string_substitute( theId, "␣\n\t", '_' );
33     cash = 0;
34     if( money_logic( theDollars, thePennies ) ) {
35         // valid amount
36         cash = dollars_pennies_to_value( theDollars,
37                     thePennies);
38     }
39     // otherwise, money logic has displayed an
40     // error message
41 }
42
43 // retrieve the wallet identification
44 string Wallet::identification( ) const
45 {
46     return id;
47 }
48
49 // retrieve the monetary value
50 double Wallet::getCash( ) const
51 {
52     return cash;
53 }
54
55 // track expense by a real number
56 void Wallet::expense( double costAmount )
57 {
58     // temporary vars
59     if( money_logic( costAmount ) ) {
60         // input is valid, can we afford this?
61         double newAmount = cash - costAmount;
62         if( money_logic( newAmount ) ) {
63             // all is aok
64             cash = newAmount;
65         }
66     }
67     // other wise money_logic has displayed an error message
68     // to cout
69 }
70
71 // track expenses
72 void Wallet::expense( int costDollars, int costPennies )
73 {
74     // temporary vars
75     double amount;
76     // convert to a double
77
78     amount = dollars_pennies_to_value( costDollars, costPennies );
79     // chain to the amount implementation
80     expense( amount );
81 }
82
83 // track income by amount
84 void Wallet::income( double earnAmount )
85 {
86     if( money_logic( earnAmount ) ) {
87         // positive amount -- aok
88         cash += earnAmount;
89     }
90 }
91
92 // track income via dollars and pennies
93 void Wallet::income( int earnDollars, int earnPennies )
94 {
95     double amount;
96     amount = dollars_pennies_to_value( earnDollars, earnPennies );
97     // chain to the amount implementation
98     income( amount );
99 }
```

Class Implementation File: `Wallet.cxx` continued...

```
100
101 // print myself to the output stream
102 // returns a reference to the ostream argument provided
103 ostream& Wallet::output( ostream& os ) const
104 {
105     os << id << " wallet contains $" << cash;
106     return os;
107 }
108
109 // read value from the input stream --
110 // returns a reference to the istream argument provided
111 istream& Wallet::input( istream& is )
112 {
113     // read into local vars first
114     string n;
115     double v;
116     string wallet, contains;
117     char dsign;
118     if((is>>n>>wallet>>contains>>dsign>>v) && (dsign=='$')) {
119         // make sure money amount is ok
120         if( money_logic( v )) {
121             // aok --- store data in object.  n must be a
122             id = n; // non-empty string, (read in with >>)
123             cash = v;
124         } else {
125             // error! put the input stream into an error state
126             is.setstate( ios::failbit );
127         }
128     }
129     // return the input stream parameter, if the input
130     // statement failed, is is already in an error state
131     return is;
132 }
133
134 // makes sure monetary amounts are always positive,
135 // this routine handles *all values* for dollars and pennies,
136 //   positive, negative, and pennies more than 100
137 // displays a message to cout if amount is negative
138 bool Wallet::money_logic( int theDollars, int thePennies ) const
139 {
140     // make sure is thePennies is >0 for % below
141     int pennies_sign = 1;
142     if( thePennies < 0 ) {
143         pennies_sign = -1;
144         thePennies = -thePennies; // invert
145     }
146     // account for more than 100 pennies
147     if( thePennies >= 100 ) {
148         theDollars += (thePennies/100) * pennies_sign ;
149         thePennies %= 100;
150     }
151     if( theDollars >= 0 && thePennies >= 0 ) {
152         // aok
153         return true;
154     }
155     if( theDollars >= 1 ) {
156         // cannot have enough negative pennies to be in the red
157         return true;
158     }
159     // otherwise, dollars < 0  and pennies < 100
160     // or         dollars == 0 and pennies < 0
161     cout << "Invalid amount $" << theDollars << "." <<
162             thePennies << endl;
163     return false;
164 }
165
166 // makes sure monetary amounts are always positive
167 bool Wallet::money_logic( const double theAmount ) const
168 {
169     if( theAmount < 0 ) {
170         cout << "Invalid amount $" << theAmount << endl;
171         return false;
172     }
173     return true;
174 }
175
176 // converts dollars and pennies to a decimal dollar amount
177 double Wallet::dollars_pennies_to_value( int theDollars,
178             int thePennies ) const
179 {
180     return theDollars + double(thePennies)/100;
181 }
```

Driver "main": `Wallet_main.cxx`

```
/***
 * An application using the wallet class application
 */

#include <iostream>
#include <fstream>
#include <string>

#include "Wallet.h"

using namespace std;

// send wallet data to out1 and out2, followed by newlines
void dualOutput( const Wallet& w, ostream& out1, ostream& out2 )
{
    w.output( out1 ) << endl;
    w.output( out2 ) << endl;
}

int main()
{
    ofstream outfile( "Wallet.log" );
    // an empty anonymous wallet
    Wallet anonymous;
    // two wallets for two different people
    Wallet Bob( "Bob" ); // no cash
    Wallet Alice( "Alice", 50, 32 ); // $50.32

    // print out and log
    dualOutput( anonymous, cout, outfile  );
    dualOutput( Bob, cout, outfile  );
    dualOutput( Alice, cout, outfile );

    // more readable output
    cout << endl; outfile << endl;

    cout << "Bob␣tries␣to␣spend␣10␣dollars..." << endl;
    // can't be done, Bob is broke
    Bob.expense( 10, 0 );  // generates error message, Bob is broke
    Alice.expense( 10.0 ); // this is OK, Alice has cash
    outfile << "Alice␣spends␣10␣dollars,␣her␣new␣value␣is:␣" <<
            Alice.getCash() << endl;
    dualOutput( Bob, cout, outfile  );
    dualOutput( Alice, cout, outfile );

    // more readable output
    cout << endl; outfile << endl;

    // Bob earns money with doubles
    while( true ) {
        cout << "Enter␣a␣double␣value␣for␣Bob's␣income," << endl;
        cout << "Enter␣a␣zero␣amount␣to␣stop:␣" << flush;
        double amount;
        if( !( cin >> amount ) || !amount ) {
            break;
        }
        outfile << "Bob␣income␣" << amount << endl;
        Bob.income( amount );
        dualOutput( Bob, cout, outfile );
    }

    // more readable output
    cout << endl; outfile << endl;

    // Alice earns money with integer dollars and pennies
    while( true ) {
        cout << "Enter␣integer␣dollars␣and␣cents␣for␣Alice's␣expense," << endl;
        cout << "Enter␣a␣zero␣amount␣(for␣both)␣to␣stop:␣" << flush;
        int dollars, pennies;
        if( !( cin >> dollars >> pennies ) || ( !dollars && !pennies ) ) {
            break;
        }
        outfile << "Alice␣expense␣" << dollars << "␣" << pennies << endl;
        Alice.expense( dollars, pennies);
        dualOutput( Alice, cout, outfile );
    }

    cout << endl << "End␣of␣program." << endl;
    return 0;
}
```

Example Driver Console Output (the > ... lines represent user input)

```
> 1.23
> 0
> 4 56
> 0
??? wallet contains $0
Bob wallet contains $0
Alice wallet contains $50.32

Bob tries to spend 10 dollars...
Invalid amount $-10
Bob wallet contains $0
Alice wallet contains $40.32

Enter a double value for Bob's income,
Enter a zero amount to stop: Bob wallet contains $1.23
Enter a double value for Bob's income,
Enter a zero amount to stop:
Enter integer dollars and cents for Alice's expense,
Enter a zero amount (for both) to stop: Alice wallet contains $35.76
Enter integer dollars and cents for Alice's expense,
Enter a zero amount (for both) to stop:
End of program.
```

`Wallet.log` Data File

```
??? wallet contains $0
Bob wallet contains $0
Alice wallet contains $50.32

Alice spends 10 dollars, her new value is: 40.32
Bob wallet contains $0
Alice wallet contains $40.32

Bob income 1.23
Bob wallet contains $1.23

Alice expense 4 56
Alice wallet contains $35.76
```