

Class "Header" File: Wallet.h

```

1  /**
2   * A wallet class
3   */
4  #pragma once
5
6  #include <iostream>
7  #include <string>
8  using namespace std;
9
10 /**
11  * Wallet interface
12  */
13 class Wallet {
14 public:
15     // default ctor makes an empty, anonymous wallet
16     Wallet( );
17     Wallet( const string& theName );
18     Wallet( const string& theName,
19             int theDollars, int thePennies );
20     // retrieve the wallet identification
21     string id( ) const;
22     // retrieve the monetary value
23     double getValue( ) const;
24     // track expenses
25     void expense( int costDollars, int costPennies );
26     void expense( double costAmount );
27     // track income
28     void income( int earnDollars, int earnPennies );
29     void income( double earnAmount );
30     // read value from the input stream —
31     // returns a reference to the istream argument provided
32     istream& input( istream& is );
33     // print myself to the output stream
34     // returns a reference to the ostream argument provided
35     ostream& output( ostream& os ) const;
36 private:
37     // makes sure monetary amounts are always positive ,
38     // carries >= 100 pennies to dollars
39     // displays a message to cout if amount is negative
40     bool money_logic( int theDollars, int thePennies ) const;
41     // makes sure monetary amounts are positive
42     // displays a message to cout if amount is negative
43     bool money_logic( const double theAmount ) const;
44     // converts dollars and pennies to a decimal dollar amount
45     double dollars_pennies_to_value( int theDollars,
46                                     int thePennies ) const ;
47
48     // data members
49     string name; // owner name
50     double value; // amount of money in wallet, >=0
51 };

```

Class Implementation File: Wallet.cxx

```

1  /**
2   * A Wallet class
3   */
4
5  #include <iostream> // system headers first
6  #include <string>
7  #include "Wallet.h" // developer headers second
8  #include "string_substitute.h" // replace whitespace with _
9  using namespace std;
10
11 /**
12  * Wallet implementation
13  */
14
15 // default ctor makes a anonymous, empty wallet
16 Wallet::Wallet( )
17 {
18     name = "???" ; // anonymous marker
19     value = 0;
20 }

```

Class Implementation File: Wallet.cxx continued...

```

22 // specifies name only
23 Wallet::Wallet( const string& theName )
24 {
25     name = string_substitute( theName, "_\\n\\t", '_' );
26     value = 0;
27 }
28
29 Wallet::Wallet( const string& theName,
30               int theDollars, int thePennies )
31 {
32     name = string_substitute( theName, "_\\n\\t", '_' );
33     value = 0;
34     if( money_logic( theDollars, thePennies ) ) {
35         // valid amount
36         value = 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::id( ) const
45 {
46     return name;
47 }
48
49 // retrieve the monetary value
50 double Wallet::getValue( ) const
51 {
52     return value;
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 = value - costAmount;
62         if( money_logic( newAmount ) ) {
63             // all is aok
64             value = 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         value += 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 // print myself to the output stream
101 // returns a reference to the ostream argument provided
102 ostream& Wallet::output( ostream& os ) const
103 {
104     os << name << " _wallet_contains_" << value;
105     return os;
106 }
107
108 // read value from the input stream —
109 // returns a reference to the istream argument provided
110 istream& Wallet::input( istream& is )
111 {
112     // read into local vars first
113     string n;
114     double v;
115     string wallet, contains;
116     char dsign;
117     if((is>>n>>wallet>>contains>>dsign>>v) && (dsign=='$')) {
118         // make sure money amount is ok
119         if( money_logic( v ) ) {
120             // aok — store data in object
121             name = n;
122             value = v;
123         } else {
124             // error! put the input stream into an error state
125             is.setstate( ios::failbit );
126         }
127     }
128     // return the input stream parameter, if the input
129     // statement failed, is is already in an error state
130     return is;
131 }
132
133 // makes sure monetary amounts are always positive,
134 // this routine handles *all values* for dollars and pennies,
135 // positive, negative, and pennies more than 100
136 // displays a message to cout if amount is negative
137 bool Wallet::money_logic( int theDollars, int thePennies ) const
138 {
139     // make sure is thePennies is >0 for % below
140     int pennies_sign = 1;
141     if( thePennies < 0 ) {
142         pennies_sign = -1;
143         thePennies = -thePennies; // invert
144     }
145     // account for more than 100 pennies
146     if( thePennies >= 100 ) {
147         theDollars += (thePennies/100) * pennies_sign ;
148         thePennies %= 100;
149     }
150     if( theDollars >= 0 && thePennies >= 0 ) {
151         // aok
152         return true;
153     }
154     if( theDollars >= 1 ) {
155         // cannot have enough negative pennies to be in the red
156         return true;
157     }
158     // otherwise, dollars < 0 and pennies < 100
159     // or dollars == 0 and pennies < 0
160     cout << "Invalid_amount_" << theDollars << "." <<
161         thePennies << endl;
162     return false;
163 }
164
165 // makes sure monetary amounts are always positive
166 bool Wallet::money_logic( const double theAmount ) const
167 {
168     if( theAmount < 0 ) {
169         cout << "Invalid_amount_" << theAmount << endl;
170         return false;
171     }
172     return true;
173 }
174
175 // converts dollars and pennies to a decimal dollar amount
176 double Wallet::dollars_pennies_to_value( int theDollars,
177     int thePennies ) const
178 {
179     return theDollars + double(thePennies)/100;
180 }
181

```

Driver "main": Wallet_main.cxx

```

1  /**
2   * An application using the wallet class application
3   */
4
5  #include <iostream>
6  #include <fstream>
7  #include <string>
8
9  #include "Wallet.h"
10
11 using namespace std;
12
13 // send wallet data to out1 and out2, followed by newlines
14 void dualOutput( const Wallet& w, ostream& out1, ostream& out2 )
15 {
16     w.output( out1 ) << endl;
17     w.output( out2 ) << endl;
18 }
19
20 int main()
21 {
22     ofstream outfile( "Wallet.log" );
23     // an empty anonymous wallet
24     Wallet anonymous;
25     // two wallets for two different people
26     Wallet Bob( "Bob" ); // no cash
27     Wallet Alice( "Alice", 50, 32 ); // $50.32
28
29     // print out and log
30     dualOutput( anonymous, cout, outfile );
31     dualOutput( Bob, cout, outfile );
32     dualOutput( Alice, cout, outfile );
33
34     // more readable output
35     cout << endl; outfile << endl;
36
37     cout << "Bob tries to spend 10 dollars..." << endl;
38     // can't be done, Bob is broke
39     Bob.expense( 10, 0 ); // generates error message, Bob is broke
40     Alice.expense( 10.0 ); // this is OK, Alice has cash
41     outfile << "Alice spends 10 dollars, her new value is:" <<
42         Alice.getValue() << endl;
43     dualOutput( Bob, cout, outfile );
44     dualOutput( Alice, cout, outfile );
45
46     // more readable output
47     cout << endl; outfile << endl;
48
49     // Bob earns money with doubles
50     while( true ) {
51         cout << "Enter a double value for Bob's income," << endl;
52         cout << "Enter a zero amount to stop:" << flush;
53         double amount;
54         if( !( cin >> amount ) || !amount ) {
55             break;
56         }
57         outfile << "Bob income" << amount << endl;
58         Bob.income( amount );
59         dualOutput( Bob, cout, outfile );
60     }
61
62     // more readable output
63     cout << endl; outfile << endl;
64
65     // Alice earns money with integer dollars and pennies
66     while( true ) {
67         cout << "Enter integer dollars and cents for Alice's expense," << endl;
68         cout << "Enter a zero amount (for both) to stop:" << flush;
69         int dollars, pennies;
70         if( !( cin >> dollars >> pennies ) || ( !dollars && !pennies ) ) {
71             break;
72         }
73         outfile << "Alice expense" << dollars << " " << pennies << endl;
74         Alice.expense( dollars, pennies);
75         dualOutput( Alice, cout, outfile );
76     }
77
78     cout << endl << "End of program." << endl;
79     return 0;
80 }

```

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
```