```
//
//  main.cpp
//  AbsoluteCpp_ch7_2
//


#include <iostream>
#include <cmath>
#include <cstdlib>
using namespace std;

//Data consists of two items, an amount of money for the account balance
//and a percent for the interest rate.
class BankAccount
{
public:
    BankAccount(double balance, double rate);
    //Initializes balance and rate according to arguments.

    BankAccount(int dollars, int cents, double rate);
    //Initializes the account balance to $dollars.cents. For a negative
     balance both
    //dollars and cents must be negative. Initializes the interest rate to
     rate percent.

    BankAccount(int dollars, double rate);
    //Initializes the account balance to $dollars.00 and
    //initializes the interest rate to rate percent.

    BankAccount( );
    //Initializes the account balance to $0.00 and the interest rate to
     0.0%.

    void update( );
    //Postcondition: One year of simple interest has been added to the
     account.
    void input( );
    void output( );
    double getBalance( );
    int getDollars( );
    int getCents( );
    double getRate( );//Returns interest rate as a percent.

    void setBalance(double balance);
    void setBalance(int dollars, int cents);
    //checks that arguments are both nonnegative or both nonpositive

    void setRate(double newRate);
    //If newRate is nonnegative, it becomes the new rate. Otherwise abort
     program.

 private:
    //A negative amount is represented as negative dollars and negative
     cents.
```

```
    //For example, negative $4.50 sets accountDollars to -4 and
     accountCents to -50.
    int accountDollars; //of balance
    int accountCents; //of balance
    double rate;//as a percent

    int dollarsPart(double amount);
    int centsPart(double amount);
    int round(double number);

    double fraction(double percent);
    //Converts a percent to a fraction. For example, fraction(50.3) returns
     0.503.
};

int main( )
{
    BankAccount account1(1345.52, 2.3), account2;
    cout << "account1 initialized as follows:\n";
    account1.output( );
    cout << "account2 initialized as follows:\n";
    account2.output( );

    account1 = BankAccount(999, 99, 5.5);
    cout << "account1 reset to the following:\n";
    account1.output( );

    cout << "Enter new data for account 2:\n";
    account2.input( );
    cout << "account2 reset to the following:\n";
    account2.output( );

    account2.update( );
    cout << "In one year account2 will grow to:\n";
    account2.output( );

    return 0;
}

BankAccount::BankAccount(double balance, double rate)
 : accountDollars(dollarsPart(balance)), accountCents(centsPart(balance))
{
    setRate(rate);
}

BankAccount::BankAccount(int dollars, int cents, double rate)
{
    setBalance(dollars, cents);
    setRate(rate);
}

BankAccount::BankAccount(int dollars, double rate)
                              : accountDollars(dollars), accountCents(0)
{
    setRate(rate);
```

```
}

BankAccount::BankAccount( ): accountDollars(0), accountCents(0), rate(0.0)
{/*Body intentionally empty.*/}

void BankAccount::update( )
{
    double balance = accountDollars + accountCents*0.01;
    balance = balance + fraction(rate)*balance;
    accountDollars = dollarsPart(balance);
    accountCents = centsPart(balance);
}

//Uses iostream:
void BankAccount::input( )
{
    double balanceAsDouble;
    cout << "Enter account balance $";
    cin >> balanceAsDouble;
    accountDollars = dollarsPart(balanceAsDouble);
    accountCents = centsPart(balanceAsDouble);
    cout << "Enter interest rate (NO percent sign): ";
    cin >> rate;
    setRate(rate);
}

//Uses iostream and cstdlib:
void BankAccount::output( )
{
    int absDollars = abs(accountDollars);
    int absCents = abs(accountCents);
    cout << "Account balance: $";
    if (accountDollars < 0)
        cout << "-";
    cout << absDollars;
    if (absCents >= 10)
        cout << "." << absCents << endl;
    else
        cout << "." << '0' << absCents << endl;

    cout << "Rate: " << rate << "%\n";
}

double BankAccount::getBalance( )
{
    return (accountDollars + accountCents*0.01);
}

int BankAccount::getDollars( )
{
    return accountDollars;
}

int BankAccount::getCents( )
{
```

```
    return accountCents;
}

double BankAccount::getRate( )
{
    return rate;
}

void BankAccount::setBalance(double balance)
{
    accountDollars = dollarsPart(balance);
    accountCents = centsPart(balance);
}

//Uses cstdlib:
void BankAccount::setBalance(int dollars, int cents)
{
    if ((dollars < 0 && cents > 0) || (dollars > 0 && cents < 0))
    {
        cout << "Inconsistent account data.\n";
        exit(1);
    }
    accountDollars = dollars;
    accountCents = cents;
}

//Uses cstdlib:
void BankAccount::setRate(double newRate)
{
    if (newRate >= 0.0)
        rate = newRate;
    else
    {
        cout << "Cannot have a negative interest rate.\n";
        exit(1);
    }
}

int BankAccount::dollarsPart(double amount)
{
    return static_cast<int>(amount);
}

//Uses cmath:
int BankAccount::centsPart(double amount)
{
    double doubleCents = amount*100;
    int intCents = (round(fabs(doubleCents)))%100;//% can misbehave on
     negatives
    if (amount < 0)
        intCents = -intCents;
    return intCents;
}

//Uses cmath:
```

```cpp
int BankAccount::round(double number)
{
    return static_cast<int>(floor(number + 0.5));
}

double BankAccount::fraction(double percent)
{
    return (percent/100.0);
}
```