

Project 2

For this project you will be rewriting project 1 using classes, vectors, and independently compiled modules.

Specifications:

1. This program should be written using 5 separate files: Category.h, Category.cpp, Transaction.h, Transaction.cpp, and Proj2.cpp. You will also need to create a makefile.
2. Your application program should declare a vector of Category objects. Your application program should use Category's overloaded input operator (>>) to read the information for each category from a file called "Category.txt". The category objects should be pushed onto the back of the Category vector. After all of the category descriptions have been read, the vector should be sorting using the sort function from the algorithm library. The information in "Category.txt" will be formatted as follows:
 - a. column 1-3: category number.
 - b. columns 5-13: current category balance.
 - c. columns 15-80: category name.
3. Each category has a vector of Transaction objects. Your application program should use Transactions's overloaded input operator (>>) to read the information for each transaction from a file called "Transaction.txt". As each object is read, it should be added to the appropriate category (using the AddTransaction method) based on the number attributes of the category and transaction classes. The information will be formatted as follows:
 - a. column 1-3: budget category number.
 - b. columns 5 -14: assignment date – yyyy/mm/dd
 - c. columns 16-24: transaction amount.
 - d. columns 26-40: transaction description.
4. After all of the transactions have been read, your application program should print a report similar to the report generated for project 1. To print the report, your application program should use Category's overloaded output (<<) operator for each category in the vector of Category objects.
 - a. Category's overloaded output operator should sort the transactions associated with the category using the sort algorithm. The operator should then print the heading line for the Category, call Transaction's output operator to print each of the transaction lines, and then print the final line for the category. Each transaction line of the report will include the date of the transaction, the amount, the description (name) of the transaction, and the current category balance.

Date Due: 6 October 2008, 11:59 pm.

To turn in: A tarred and zipped directory called *yourlastname*P2 containing your .h, .cpp, and makefiles. The tar file should be called *yourlastname*P2.tgz. The steps for creating the tar file are:

1. Create a directory in your file system called <last_name>P2
eg. wattsP2
2. Copy all of the files you are submitting to your <last_name>P2 directory.
ie. *.cpp, *.h, makefile

3. Make the parent directory of your <last_name>P2 the current directory.
4. Enter
umask 033
5. Enter the follow tar command:
tar cfvz <last_name>P2.tgz <last_name>P2/*
6. Enter
cp <last_name>P2.tgz ~tiawatts/cs215drop/.
7. Enter
umask 077

Category.txt		Transaction.txt	
301	1000.00 Rent	301	2007/08/21 -1000.00 Rent
302	500.00 Apartment Supplies	302	2007/08/22 -123.45 Target Purchase
102	-50.00 Books	102	2007/08/23 -82.00 Absolute C++
501	300.00 Clothing	101	2007/08/27 -2.79 Stapler
101	200.00 School Supplies	202	2007/08/28 -40.00 Gas
202	200.00 Gas	302	2007/08/24 -85.71 Blender
100	3000.00 Tuition	102	2007/08/25 -55.00 Intro to Linux
		501	2007/08/27 -300.00 Jeans
		102	2007/08/24 500.00 Text Stipend
		100	2007/09/01 -1234.56 Fall Tuition
		501	2007/08/27 -30.00 Sweatshirt

Tuition

Category: 100	Balance:	3000.00
2007/09/01 -1234.56 Fall Tuition		1765.44
Category: 100	Balance:	1765.44

School Supplies

Category: 101	Balance:	200.00
2007/08/27 -2.79 Stapler		197.21
Category: 101	Balance:	197.21

Books

Category: 102	Balance:	-50.00
2007/08/23 -82.00 Absolute C++		-132.00
2007/08/24 500.00 Text Stipend		368.00
2007/08/25 -55.00 Intro to Linux		313.00
Category: 102	Balance:	313.00

Gas

Category: 202	Balance:	200.00
2007/08/28 -40.00 Gas		160.00
Category: 202	Balance:	160.00

Rent

Category: 301	Balance:	1000.00
2007/08/21 -1000.00 Rent		0.00
Category: 301	Balance:	0.00

Apartment Supplies

Category: 302	Balance:	500.00
2007/08/22 -123.45 Target Purchase		376.55
2007/08/24 -85.71 Blender		290.84
Category: 302	Balance:	290.84

Clothing

Category: 501	Balance:	300.00
2007/08/27 -300.00 Jeans		0.00
2007/08/27 -30.00 Sweatshirt		-30.00
Category: 501	Balance:	-30.00

```

// File Category.h
#ifndef CATEGORY_H
#define CATEGORY_H

#include <iomanip>
#include <fstream>
#include <string>
#include <vector>
#include <algorithm>
#include "Transaction.h"

using namespace std;

class Category
{
public:
    Category ();
    Category (const Category & other);
    ~Category ();
    Category & operator = (const Category & other);
    int GetNumber () const;
    void AddTransaction (const Transaction & transaction);
    bool operator < (const Category & other) const;
    void SortTransactions ();
    friend istream & operator >> (istream & ins, Category & cat);
    friend ostream & operator << (ostream & outs, const Category & cat);
private:
    int number;
    float balance;
    string name;
    vector <Transaction> trans;
};
#endif

// File Transaction.h
#ifndef TRANSACTION_H
#define TRANSACTION_H

#include <iomanip>
#include <fstream>
#include <string>

using namespace std;

struct Transaction
{
public:
    Transaction ();
    Transaction (const Transaction & other);
    ~Transaction ();
    Transaction & operator = (const Transaction & other);
    int GetNumber () const;
    float GetAmount () const;
    bool operator < (const Transaction & other) const;
    friend istream & operator >> (istream & ins, Transaction & trans);
    friend ostream & operator << (ostream & outs, const Transaction & trans);
private:
    int number;
    string date;
    float amount;
    string name;
};
#endif

```

```

// File Proj2.cpp
#include <iostream>
#include <fstream>
#include <vector>
#include <algorithm>
#include "Category.h"
#include "Transaction.h"

int main (int argc, char * argv[])
{
    // Create vector to store Categories
    vector <Category> budget;

    // Read category file ans store categories in budget vector

    // Sort budget vector

    // Read transaction file
    // Find matching category in budget vector
    // Use AddTransaction to add transaction to matching category

    // Sort transactions in each category in budget vector using SortTransactions

    // Print report for each category in budget vector
    // Category will print line for each transaction in its trans vector

    return 0;
}

```

Makefile:

```

P2test: Proj2.o Transaction.o Category.o
    g++ -g -o P2test Proj2.o Transaction.o Category.o

Proj2.o: Proj2.cpp Category.h
    g++ -g -c Proj2.cpp

Transaction.o: Transaction.cpp Transaction.h
    g++ -g -c Transaction.cpp

Category.o: Category.cpp Category.h Transaction.h
    g++ -g -c Category.cpp

clean:
    rm *.o P2test

```