Credit Card System

Rupin Mittal

UML Pseudocode

Aug 8 2020

There is a program that randomly generates a file with many transactions (date, amount, location). A structure is defined that represents each transaction. An array is filled with all the transactions in the order that they are read from the file. A second array is made which has all the transactions sorted by time. A function is made that can print the list of transactions in a nice, readable format. A function is made that can sort the transactions by money. Functions are also created so the user can add transactions. The program shows how much the user owes the credit card company and the user can also pay the amount they owe. Since the array is used in every function, the array will be an external variable.

CreditCardSystem.c

A. Contains that main() function
B. Calls the function to randomly generate the credit card data and load it into a file (CreditCardRecord.txt)
C. Calls function to load the record into an array
D. Calls function to have the array sorted chronologically
E. Starts user loop with user's menu options and handles the requests.

Record.h (Header file)

Contains the external variable that is the array of structs with the record since all the functions need the array.

extern record[struct];	//extern variable of the record array

RecordGenerator.c

A. Random will be used to create a random number, that will be converted to a date
B. Random will be used to pick one of the many locations in an enumerator (Chipotle, Target, and so on)
C. Random will be used to pick the transaction amount
D. All these will be written into a file called Record.txt.

int generateDate
int generateLocation
int generateTransaction
void generateRecord
	//for date
	//for location
	//for Transaction
	//will take all the outputs and put them into a file called Record.txt

RecordGetter.c

A. Has function to load the records from the file into the external array record[]

void getRecord(file)	//get the record from the file

RecordKeeper.c

A. Sorts the record by time
B. Has function to sort by amount if user wants
C. Contains the menu of options that the user can make
D. Can show maximum and minimum amounts of spending
E. Can print unsorted or sorted list of record
F. User can add transactions to the record
G. Shows how much user owes and keeps track of how much user has paid

void timeSort()
void amountSort()
char[] menuOptions()
int max()
int min()
void addTransaction(struct)
int oweAmount()
	//both sorts
	
	//what the possible options are
	
	//adds a struct transaction
	//should account for amount paid