

COMP167 - Major Programming Assignment 1 – Spring 2025

1 Introduction

This assignment requires you to create a Java application that will be used to plan and track expenses. The user will be able to set up a budget and then track the financial transactions.

2 Classes

You are required to implement the following classes at a minimum. **You must use the class, property and method names that are given.** You may add other classes (and methods) if you need them.

2.1 Client Class

This class is used to model one of the individuals whose finances are being budgeted.

Client	
-firstName : String -lastName : String -contact : Contact -budget : Budget -tracker : Tracker	Client first name Client last name Client contact information Client budget Client monthly transactions
+Client() +Client(fname:String, lname:String) +//Mutator and Accessor methods +loadClientFile(fileName : String) : void +saveClientFile(fileName : String) : void +getBudgetReport() : String +getTransactionReport() : String +toString()	No-arg constructor– use Java default values. Load the data from the file. Save the client data using the given filename. Returns a formatted budget report. Returns a formatted transaction report Return a string representation that conforms to input file format. Use System.lineSeparator() instead of “\n” for the newline.

2.2 Contact Class

The contact information for the client.

Contact	
-homeAddress : String -phone : String -email : String	Entire home address 10 digit phone number Email address
+Contact() +Contact(hmAddress : String, phone:String, email:String) +//Mutator and Accessor methods +toString()	Use Java defaults

2.3 BudgetCategory Class

A single budget category in the client's budget. It can be an expenditure category or an income source.

BudgetCategory	
-catName : String -catDescription : String -amount : double -categoryType : int	Name of budget category (e.g. "Housing") Optional detailed description of the category. Amount allocated to the category per month. 0 = expense, 1 = income
+BudgetCategory() +BudgetCategory(cn:String, cd:String, amt:double, type:int) +//Mutator and Accessor methods +toString()	

2.4 Budget Class

Budget information of the client.

Budget	
-categories : ArrayList<BudgetCategory> -totalAllocated : double -totalIncome : double	Collection of all budget categories. Total monthly amount of all budget categories. Total client income.
+Budget() +getAllocationBalance():double +//Mutator and Accessor methods +toString()	total income - totalAllocated See section below on how to handle the ArrayList

2.5 Transaction Class

A monthly expenditure.

Transaction	
-category : String -date : String -amount : double -description : String	Will match a budgetCategory name. Date of the transaction Amount of the transaction Description of the transaction
+Transaction() +Transaction(dt:String, amt:double, desc:String) +//Mutator and Accessor methods +toString()	

2.6 Tracker Class

Keeps track of all monthly transactions for all of the budget categories.

Tracker	
-transactions : ArrayList<Transaction>	Collection of all Transaction
-balance : double	Total monthly budget amount minus total of transactions.
+Tracker() +//Mutator and Accessor methods +toString()	

3 Handling ArrayLists

Each ArrayList should have five associated methods to perform: getNum, add, remove, get and set. So if you have an ArrayList named *widgets* that stores items of type *Widget*, then the associated UML behaviors would be:

```
+getWidgetsSize() : int //Return the number of items in the ArrayList widgets.  
+getWidget(index:int) : Widget //get the Widget at location index in ArrayList widgets  
+setWidget(index:int, item:Widget):Widget //store item at location index in the ArrayList widgets.  
+addWidget(item:Widget):void //Append the Widget to the ArrayList.  
+removeWidget( index:int ) : Widget //remove and return the Widget at  
location index
```

4 Input File

The name of the input file will be supplied using command-line arguments. If no command-line argument is supplied, then your program should prompt the user for the input file using the *JFileChooser* class. See Figure 1 for the format of the input file.

5 Output File

The format for the output file should be identical to that of the input file. In other words, after writing your output file, you should be able to read it back in as an input file. The *toString()* methods of your classes are designed to make file output simple.

```
firstName  
lastName  
address  
phone number  
email address  
amount type catName  
description  
amount type catName  
description  
...  
o  
catName  
date amount  
description  
catName  
date amount  
description  
...
```

Figure 1 - Input File Format

6 Budget Report

Your budget report should list the client's name followed by the client contact information. It should then list the budget expenditure information in tabular format followed by the total expenditure amount. Next, the report should list the income sources in tabular format followed by the source totals. Finally, the balance of expenditures and income.

7 Graphical User Interface

If you would like to add a GUI to your application, look for the GUI addendum. It will be posted later in the same folder as this assignment. You should not attempt this portion of the assignment if you have not completed the other classes.

8 Grading

If your project does not compile, it receives a grade of zero. If you do not document your program according to the documentation guidelines, the graders have been instructed to deduct **up to 25%**.

Level 1 (35%): Complete the Contact class, BudgetCategory class and Transaction class. You should create a main method that will instantiate, populate and view the contents of the objects using the toString method.

Level 2 (50%): Partially complete the Client class -- you do not have to implement the loadClientFile, saveClientFile, getTransactionReport and getBudgetReport methods and the Tracker class. Implement the Budget class.

Level 3 (70%): Complete the loadClientFile, saveClientFile and getBudgetReport

Level 4 (85%): Complete the Transaction class, Tracker class and the getTransactionReport method. Update the Client class to include all the properties.

Level 5 (100%): Implement the GUI. See the GUI addendum for details

Extra Credit: Talk to me after you finish Level 4 for extra credit ideas.

9 Submission

Create a file named Readme in the root directory of your project that describes what grade level you feel that you completed. If there is any other information the grader should know about your assignment, you should include it in this file. Archive/zip your entire NetBeans project with a zip tool and uploaded it using the assignment upload link.