# 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<br>-lastName : String<br>-contact : Contact<br>-budget : Budget<br>-tracker : Tracker | Client first name<br>Client last name<br>Client contact information<br>Client budget<br>Client monthly transactions |
| +Client()<br>+Client(fname:String, lname:String)<br>+//Mutator and Accessor methods<br>+loadClientFile( fileName : String) : void<br>+saveClientFile( fileName : String) : void<br>+getBudgetReport() : String<br>+getTransactionReport() : String<br>+toString() | No-arg constructor– use Java default values.<br><br><br>Load the data from the file.<br>Save the client data using the given filename.<br>Returns a formatted budget report.<br>Returns a formatted transaction report<br>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<br>-phone : String<br>-email : String | Entire home address<br>10 digit phone number<br>Email address |
| +Contact()<br>+Contact( hmAddress : String, phone:String, email:String)<br>+//Mutator and Accessor methods<br>+toString() | Use Java defaults |

## 2.3 BudgetCategory Class

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

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

## 2.4 Budget Class

Budget information of the client.

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

## 2.5 Transaction Class

A monthly expenditure.

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

## 2.6 Tracker Class

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

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

```
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*

# 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.

# 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. **Due 2/15/2025**

**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. **Due 2/21/2025**

**Level 3 (70%):** Complete the loadClientFile, saveClientFile and getBudgetReport. **Due 2/24/2025**

**Level 4 (85%):** Complete the Transaction class, Tracker class and the getTransactionReport method. Update the Client class to include all the properties. **Due 2/26/2025**

**Level 5 (100%):** Implement the GUI. See the GUI addendum for details. **Due 2/28/2025**

**Extra Credit:** Talk to me after you finish Level 4 for extra credit ideas. **Due 2/28/2025**

# 9 Submission

You will submit your assignments through GitHub. You will create a branch to work on a level. When you feel that the level is complete, you will make a pull request for one of the teaching assistants. They will give you feedback or give you a grade on the level. After receiving a grade, you can move on to the next level. More information is found in the repository README file.