Rahul Kukreja, Jeffrey Yim, Simon Thorp

rkukreja1@bcit.ca, jyim3@bcit.ca, sthorp@bcit.ca

BCIT CST - Computer Systems Technology Diploma

Welcome!

Welcome to Lab #04: Getting to know the F.A.M.

By the end of today's lab we will:

1. Learn how to plan before writing code by creating a preliminary UML Class Diagram
2. Experience writing code by tackling the most important aspects of a system first by getting specific portions of Assignment 1.
3. Apply the S.O.L.I.D Design Principles when planning your code.

The lab is due **Saturday, 6th of February at 11:59 PM**. Submissions past this time will be penalized by 25%.

You can also walk up to me in the lab itself if you want to be graded and receive feedback in-person (This results in the best feedback and I encourage you to try and finish as much of the lab within the allotted lab time).

Grading

This lab is graded out of 10.

For full marks this week, you must:

- **(2 point)** Upload your work to the Learning Hub (D2L) and put your files in a folder called: 3522_A00(Student#)_Lab4 (If working with a partner, include both student IDs)
- **(2 points)** Draw out a preliminary UML Class Diagram
- **(4 points)** Implement the behavior outlined in the lab below.
- **(2 Mark)** Follow PEP-8 and PEP-257 conventions. Try to eliminate all PyCharm warnings, use appropriate identifiers and best practices (atomic functions, reuse code, etc.).

Read the WHOLE document before starting the lab. Understand the context in which you are writing code. Don't just follow the instructions line by line.

Eat the Frog.

This lab will serve as time for you to work on your assignment. As such I expect you to hit some milestones and start writing code for the **F.A.M.**

If you haven't already go and read the assignment brief on D2L.

Seriously, don't read any more of this document until you have done that.

.

..

...

Finished? Good!

The best-selling (apparently) self-help book "*Eat the Frog*" (If you're interested, read it and let me know if it's any good, I haven't read it) mentions the following:

Mark Twain once said that if the first thing you do each morning is to eat a live frog, you can go through the day with the satisfaction of knowing that that is probably the worst thing that is going to happen to you all day long.

Now, my research online (*read*: google and open the first link) mentions that Mark Twain probably never said anything about eating frogs. Despite that I wholeheartedly agree with this statement.

Your **"frog"** is the biggest and most important task of the day. If you can get that done and out of the way first, then the rest of the day will be a lot less stressful. In line with this philosophy we are going to eat the frog in the assignment. The frog here consists of implementing the foundation upon which we can complete the rest of the assignment. This involves creating a generic user, and recording transactions.

Step 1: Plan *'just enough'* before you code.

For this first task you must sketch out a **preliminary** UML Class Diagram that will represent the complete intended solution to the assignment. **Include this UML diagram as an image (.png, .jpg) in the root directory of this lab**

To do this you must:

- Identify all the classes in your system
- Be able to explicitly state the responsibility of each class.
- Identify any and all important methods and attributes for each class
- Map out the relationships between each of the classes.

You will have to read through the assignment a few times while you do this. Ask if any of the requirements are unclear.

Now I use the word **preliminary** very intentionally. It has been bolded after all. You may:

- Sketch out your UML diagrams on paper.
- It does not need to specify everything the class will have, just the important methods and attributes as per the class's responsibility.

That being said it should still be **legible** and you should still include a scan of the sketch in your assignment folder.

This does not change the fact that your final UML diagram that will accompany the assignment submission will need to be made digitally and specify everything (Attributes and methods) that the class can do.

It is important to plan your code before you start. But it is also important you do not go too far with the planning and spend a lot of time here. Plan just enough so that you have an idea of how all the classes will interact with each other in your system. As you write your code, the design will change and you will get other ideas. This is OK and a good sign.

Recording Transactions for a generic user.

For the rest of the lab you must start working on the assignment. For this lab you will work in the Lab 4 folder of your repo. When you begin working on Assignment 1, copy your code from Lab 4 into your Assignment 1 folder on your repo.

For this lab you must meet the following requirements

- Model the class that represents your users (the child who's account is being monitored). This can be a generalized class that doesn't deal with User Types (yet).
- Implement the `load_test_user()` method/function that initializes and returns an object representing your test user with pre-set hardcoded budgets. You should use this method when working on parts of your system that don't deal with user registration to speed up development.
- Implement the **Record Transaction Menu** and allow the user interacting with your application to enter and save multiple transaction details. Print the recorded transactions.

That's It!

That's all for this lab. I hope this helps you get started on your assignment. Don't hesitate to ask any questions as they come up.