University of Texas at El Paso - Department of Computer Science
CS1101: Introduction to Computer Science Lab
Spring 2019 Comprehensive Lab 1

Due Dates –

1. **Document containing pseudocode-Tuesday, March 5th by 1:30pm via Blackboard. Review your pseudocode with a member of the instruction team in labs or office hours no later than the March 1st.**
2. **Java Program and Tracing document-Tuesday, March 12th by 1:30pm via Blackboard. Demo to a member of the instruction team during labs or office hours no later than March 16th.**

1. Objective

This challenge lab will enable you to apply, in a practical scenario, the topics we have introduced in the class so far: input/output including files, variables, conditionals and loops.

2. Learning outcomes

After completing this assignment, you will be able to:

- Analyze problems and express a solution algorithm using pseudocode.
- Implement a pseudocode algorithm in a high-level language, including the correct use of arithmetic and logical expression and simple input/output operations.
- Use the syntax and semantics of a high-level language to represent: basic variable, assignment and arithmetic operations and basic control structures (if-then, for-loop, while-loop).

3. Your Challenge – A Secure, Personal Banking App

You have been hired by CS 1301 and CS 1101 instruction team to create a program that enables a user to do online banking transactions, called *minerBank*.

3.1. *Verifying credentials*

Your first task is to verify credentials for each user by asking the user for their username and PIN as follows:

```
Enter username:
Enter PIN:
```

You will verify the user credentials by reading the information of each user from the file *users.txt* (included for this assignment). This file will have on each row the following user information: username, pin, checking account balance, savings account balance.
For example, the file *users.txt* may have the row provide below with information about Alice. This row indicates that user name is `alice`, her password is `1234`, her checking account balance is $`500`, and her savings account balance is $`400`.

```
alice 1234 500 400
```

You will verify that the username and PIN entered by the user in 3.1. matches the username and PIN obtained from the users file. A user only has 3 opportunities to enter the correct password, otherwise the program will exit with the following message:

```
Your username and/or password is not correct. Goodbye!
```

3.2. *Display menu*

If the user provides valid credentials, *minerBank* will have a simple menu at the beginning. This menu will be displayed as many times as desired by the user until she/he presses 5 where your program will end:

```
                  1. Check Balance
                  2. Withdraw Money
                  3. Deposit Money
                  4. Transfer Money
                  5. Exit

Welcome to minerBank! Enter the menu option you want (1 to 5):
```

3.3. *Option 1: Check balance*

This option will further ask the user to choose between checking account and savings account.

```
     Select one of the following options:
                    1. Checking Account
                    2. Savings Account
```

Your program will display the balance for the corresponding account.
If the user selects option 1, in **`alice`**'s example it would display $**`500`**, otherwise, if the user selects option 2, your program would display $**`400`**.

3.4. *Option 2: Withdraw money*

This option will:

a. Ask the user to choose between checking account and savings account (similar as shown in 3.3).
b. Ask the user the amount of money they would like to withdraw from the selected account. The user is not allowed to withdraw more than $200 or more than what their current balance is.
c. Subtract the amount from the corresponding account.
d. Print the final balance from the corresponding account.

3.5. *Option 3: Deposit money*

This option will:

a. Ask the user to which account they would like to make a deposit (checking or savings, refer to 3.3).
b. Ask the user the amount of money they would like deposit, no more than $1200 is allowed.
c. Print the final balance from the corresponding account.

3.6. *Option 4: Transfer money*

This option will:

a. Ask the user from which account they would like to withdraw from (checking or savings, refer to 3.3).
b. Ask the user the amount of money they would like to transfer where no more than $200 or more than what their current balance has is allowed.
c. Transfer from the selected account to the other.
d. Print the final balance of both accounts.

3.7. *Option 5. Exit*

Only by pressing this menu option the program will end, otherwise it will continuously display the menu to allow the user to continue selecting other menu items. However, if this option is pressed it will print a goodbye message created by you.

Notes:

- Assume that usernames provided in the file users are unique.
- The final balances for a user's account will not be recorded (you are expected to read from the file but not write to it) in the file, just displayed in the screen.

4. Testing Case

Assume Alice wants to transfer 150 from her checking to her savings account. Her information in the file is as follows:

```
alice 1234 500 400
```

Step 1. Alice will enter the username `alice` followed by `1234`.

Step 2. Alice will input option `4` for transferring money, followed by the option `1` for checking account. Alice will be prompted to input the transfer amount. Alice will input `150`.

```
Input the transfer amount:150
```

```
The final balance of your checking account is: $350
The final balance of the savings account is $550
```

Step 3. Alice is presented with the main menu, where she types option 5 (Exit). The good-bye message is printed, and the program ends.

Bonus: Think about a bonus feature you could include that will allow you to practice using input/output operations, a sequence of steps, conditionals or loops, and testing cases. You should include the code of your bonus feature and explain this feature during the demo to the instruction team. Possible bonus features will be discussed during the labs, think about how to make your code more robust, or add additional features such as calculating a fee for using the app.

5. Task 1: Write pseudocode

Create a word/pdf document named “LastName_FirstNameInitial_CompLab1_**Pseudo**”. In this document, you will write a description of your approach to the challenge using the pseudocode notation reviewed in lectures. Your document must include the following sections: Task Description, Variables, Pseudocode, Tracing and Assumptions.

5.1. *Task description*

In simple words, describe the task that was given to you.

5.2. *Variables*

Define the variables and their corresponding datatype required to store the data needed by the *minerBank* system. Make sure you use meaningful names. For each variable, include a sentence that describes how you’re going to use the variable, e.g., variable to store the balance of a checking account.

5.3. *Pseudocode*

Using the pseudocode notation used in the lecture, write the pseudocode required to implement the functionality of *minerBank*. Make sure your pseudocode includes instructions for the following:

- Asking and retrieving the information you need from the user, e.g., username and PIN.
- Implement a conditional when necessary, e.g., Is the PIN correct?
- Implement an iterative process when necessary, e.g., finding user’s information in a file.

5.4. *Assumptions*

If something was not clear about your challenge and you make an assumption, please include a description of your assumption in this section of your word document.

6. Task 2: Implement minerBank in Java and trace your code.

You will implement the pseudocode that you created and submitted using Java. Your implementation should match for the most part your pseudocode. It is OK if after doing some implementation you refine your pseudocode and will by consequence end up looking a bit different.

6.1. *Download and annotate Java file.*

Download the minerBank.java provided with this assignment through Blackboard. Modify the first lines of the code to include your name as follows:

```
/* CS1101 – Intro to Computer Science
Instructor: Aguirre OR Akbar OR Villanueva
Comprehensive Lab 1
By including my name below, I confirm that:
-  I am submitting my original work.
-  If I include code obtained from another source or I
received help I am giving attribution to those sources as
comments.
-  This submission does not incur in any academic
dishonesty practice as described in the course syllabus.
Modified and submitted by: [YOUR NAME GOES HERE]
*/
```

6.2. *Modify the Java file.*

Using the pseudocode notation used in the lecture, write the pseudocode required to implement the functionality of *minerBank*. Make sure you follow best practices when writing your code such as adding documentation when needed and using proper indentation.

6.3. *Run your program.*

Run your program and make sure that it's working as expected, e.g., by entering the inputs you used in your tracing example, you will see the expected outcomes.

6.4. *Tracing*

Create a document (PDF/Word) named "LastName_FirstNameInitial_CompLab1_**Trace**" where you will trace the values from Alice's example on section 4. Trace the value of your variables using the notation used in lectures.

7. Task 3: Submit your solution.

Using Blackboard to submit the following:

- A Word/PDF document to turn in on **Monday, March 4th by 10am** and named as the following: "*LastName_FirstNameInitial_CompLab1_**Pseudo***".
- The modified **JAVA file** called minerBank.java and the tracing document called "*LastName_FirstNameInitial_CompLab1_**Trace***". On **Monday, March 11th by 10am**

8. Grade percentage breakdown

25% - Pseudocode
10% - Appropriate use of input/output operations (in Java)
15% - Appropriate use of conditional (i.e., if-then) statements (in Java)
15% - Appropriate use of iterations (i.e., for-loop, while-loop) statements (in Java)
10% - Appropriate documentation (in Java)
5% - Appropriate notation and indentation (in Java)
10% - Program compiles, runs and contains the functionality required.
10% - Student answers all questions during demo.
Up to 20% - Bonus feature

Penalizations:
7.5% - Every 24 hours for up to 72 hours. For example, if you submit 36 hours later your maximum percentage is 85%.
10% - Not following the instructions for submission.
Refer to the course syllabus for policies on academic dishonesty.

9. Tips

- **Start** early.
- **Plan** to work 3 to 5 additional hours outside the lab to complete this assignment.
- **Ask** clarifying questions to the instruction team if something is not clear.
- Plan to submit at least 1 hr. before the deadline to deal with potential Blackboard bugs.
- **Have fun** and keep it simple!