ECE 101: Introduction to Computing Systems
Programming Lab #4 – 100 Total Points
Coverage: Classes

1. College Life of an ECE Major (100 points)

PLEAES READ THIS DOCUMENT CAREFULLY!!!!

In this lab, you are going to be working with python classes to simulate, to some extent, what a typical student goes through from freshman year until he/she graduates. By the end of this exercise, you should be able to understand the following fundamental concepts of Object-Oriented Programming (OOP) in general – designing classes, constructing objects from classes, inheriting attributes and behavior, hiding data, and relationship between child parent classes.

For this project, you have been given a zip file that contains five (5) documents and one (1). The contents of the zip file are listed below, but their purpose will be described in their respective sections.

- Bank.py
- Person.py
- Student.py
- Examples (Folder)
    - Bank0.pyc
    - Person0.pyc
    - Student0.pyc
    - Using Student0.txt
- Finals.txt
- Report.txt

It should be clear at this point of the semester what the purpose of the Report.txt file is, so we will not cover that in this document.

The Finals.txt file contains a string questions. There are twenty (20) questions in total. The answers for these questions, which are either True or False, have not been given to you. Your job is to find out what the answers to these are so that you can use it in your code.

As usual, **your goal is the meet the requirements of this document**. But, you are at liberty to add extra functionality for extra points, as long as your functionality do not deviate from the basic requirements of this document. **You must meet the requirements stated here, before adding extra functionality!** If you decide to add extra functionality to your program, make sure it is well documented in the code as well as the report.txt file.

I. Bank.py File

This python file contains code for one (1) class called **BankAccount**. **You are not allowed to edit the contents of this file.** But as part of this project, you will implement functionality that relies on this file.

Note that the name of the file, Bank, is NOT the same as the name of the class, BankAccount. It is important that you note this distinction.

The class defined in this file gives you the capability to create Bank Account objects. As part of this lab, you are **required** to know/understand what this class does. You are to play around (create an object of this class, display its variables, and call its methods) with this class to find out what it does, **but you are**

not allowed to edit it! If you make changes to this class while you play around with it, you have to re-download the original copy from blackboard to use in the subsequent parts of the lab.

II. Person.py

This python file **will** (this means the code has not been written yet – it is your job to write it) contain code for one (1) class called **Person**. You are **required** to create this class the meets the requirements below. Also, do not add any other functionality to this class, except for the ones listed below!

- One (1) class variable called **population**
  - The initial value of this variable **must** be zero (0)
  - You are to increase the value of this variable **by one every time** a new instance of this class is created
- One (1) constructor
  - You are to set the instance and class variable(s)
  - Four (4) instance variables called; **firstName**, **lastName**, **dateOfBirth**, **gender**
  - Two (2) of the instance variables, **dateOfBirth** and **gender**, must be hidden
- Two (2) methods called; **getDateOfBirth**, **getGender**
  - The class methods are to **return** the values of the respective variables they represent. If you don't know it, figure it out from the name of the variables in this class

III. Examples Folder

This folder contains python byte code for a somewhat working version of this lab. Please not that the example code in this folder does not implement all the requirements. It is only a guide. Run this code to figure out what your expected code must do. For instructions on how to run the example code, see the **Using Studen0.txt file**.

IV. Student.py

The **Student.py** python file **will** (this means the code has not been written yet – it is your job to write it) contain code for one (1) class called Person. Most of your coding will be done in this file. You are to create methods and variables to meet the following requirements

- **Four (4) class variable**
  - **numberOfStudents** (To store the total number of instances of the Student class)
    - The initial value of this variable must be zero (0)
    - You are to increase the value of this variable by one every time a new instance of this class is created
  - **tuition** (How much tuition each student pays to register for classes)
    - The value of this variable must be **$3333**
  - **pricePerBook** (How much each student pays for books)
    - The value of this variable must **122.50**
  - **__finals** (Dictionary data type to store example questions and answers. More on this in the **__populateQuestionsAnswers** method)
    - Note that this variable is a dictionary of dictionaries
    - This variable has five (5) dictionary
      - Populated → Specifies if the variable has been populated with values
      - Freshman → Stores the final questions and answers for a freshman
      - Sophomore → Stores the final questions and answers for a Sophomore
      - Junior → Stores the final questions and answers for a Junior
      - Senior → Stores the final questions and answers for a Senior

        - The keys Freshman, Sophomore, Junior, and Senior have corresponding values that are dictionaries
            - The dictionary has the following keys
                - Questions
                - Answers
            - The value for the Questions key will contain a **list** of strings
            - The value for the Answers key will contain a tuple of Booleans
        - The contents of this variable will be given to you in the file called finals, and your job is to populate this variable using the **__populateQuestionsAnswers** method described later
- **Eleven (11) instance variables**
    - The following four (4) of these instance variables will be inputs to the constructor
        - **firstName**
        - **lastName**
        - **dateOfBirth**
            - The only sequence of string for this variable must be of the format of two-digits month, two-digits day, and four-digits year. Example 01-01-2010
            - You are therefore required to check this input value to make sure that it follows the format above
            - To validate the dateOfBirth value, you can import and use python's **datetime.strptime** funtion
        - **Gender**
            - The value of this variable can only be **'M'** or **'F'**
            - You are therefore required to check this input value to make sure that it meets the format above
    - The following remaining seven (7) must be generated
        - **__studentID**
            - This variable must be a unique number of arbitrary number of digits. You must import the uuid to create data for this variable
        - **__major**
            - This variable can be whatever string you prefer. Example **ECE**
        - **__gpa**
            - The initial value of this variable must be set to **four (4)**
        - **__studentType**
            - The initial value of this variable must be set to **Undergraduate**
        - **__emailAddress**
            - The value for the variable must be created from the input **firstName** and **lastName** variables
            - In addition, you must add a string specifying the students major and the university name.
            - As an example, if the students major is **ECE**, and the name of the university is **Temple**, then the student's email address will be of the format; **<firstName>.<lastName>@ece.temple.edu**
        - **__studentYear**
            - This variable can take five (5) possible values; Freshman, Sophomore, Junior, Senior, and Graduate
            - But the initial value for this variable must be **Freshman**
        - **__currentClasses**
            - This is a **list** of classes the student is currently taking
            - The initial value for this variable must be an empty list, **[]**

- **Constructor Method**
  The input to the constructor must be the variables outlined in the instance variables section above. When your class is called, it must do the following;
  - Validate dateOfBirth and gender variables. If any of there variables does not meet the requirement, you must display a message and return
  - Initialize other instance variables
  - Create a Person object for this student using the Person class
    - The input variables to the Person class must be; **firstName**, **lastName**, **dateOfBirth**, and **gender**
  - Create a BankAccount object for this student using the BankAccount class
    - The input variables to the BankAccount class must be; the string '**Checking'**, the float **5000.00**, and the variable **__emailAddress**

  A call to your class must print out information that looks something like figure 1 below.

```
>>> me = Student('R','A','01-01-2014','M')
Congratulations R, for been accepted to the ECE program in <some university>!
Please wait while we create your student account
19632

Thank you for your patients R.
Your student ID number is: "19632",
Your email address is: R.A@ece.some.university.edu

As part of university policy, a Bank Account is been created for you. Please enter login credentials!
Password: ra

Please remember your login credentials;
Username: R.A@ece.some.university.edu
Password: ra

Do you want to log into your student account? (Y/N): Y
You are logged into your student account!
```

Figure 1: Constructing a student object

- **Display Method**
  You are also required to provide a method that display information about your objects of your Student class. An example is shown in figure 2 below.

```
>>> print( me )

********** Personal Information **********
Name: R A
DOB: Wednesday, January 01, 2014
Gender: Male

********** School Information **********
Major: ECE
GPA:4.0
Student: Undergraduate
Email: R.A@ece.some.university.edu
Year: Freshman
Courses Taken:[]

***** Account Information *****
Account Number: 4066330554
Account Type: checking
Balance: 500.0
```

Figure 2: Display information about student object

- **Getter Methods**
  Since most of the instance variables about your object are hidden, you would need methods to get these variables. These methods are called getter methods. Create the following methods to return the values corresponding to the name of the methods below.
  - **getStudentID**
  - **getMajor**
  - **getGpa**
  - **getStudentType**
  - **getEmailAddress**
  - **getStudentYear**

  In addition to the methods above, you would also need two (2) additional methods to get the instance variables **__dateOfBirth** and **__gender** from the Person object. Please note that the Person class already has these methods called **getDateOfBirth** and **getGender**. But your job is create these exact methods inside of the Student class. So, you need to find a way to call the Person class methods, and modify the values they return in the Student class. To call these methods you would need a keyword called **super.**

  Hint: <value> = super(<Name of Calling Class>, self).<name of method>(<any inputs>)

  - **getDateOfBirth**
    When this function is called, it must get the date of birth from the Person class, and display the date in words. Example **01-01-2014** will be formatted to **Wednesday, January 01, 2014.** Import datetime from datetime and use the **strftime** method.

    Hint: <value>.strftime('%A, %B %d, %Y')

  - **getGender**
    When this function is called, it must get the gender from the Person class, and "Male" or 'Female" depending on if the original input is 'M' or 'F'

    Figure 3 below shows the output from these methods.

```
>>> me.getDateOfBirth()
'Wednesday, January 01, 2014'
>>> me.getGender()
'Male'
```

Figure 3: Calling Functions from parent

- **registerForClasses Method**
  This method will assign values to the **__currentClasses** instance variable. When this method is called, you have to assign classes to this student based on their year as shown below. Note that the course code must be matched to the student's major. In this example, we are using ECE major.
  - Freshman → ECE 101 and ECE 102
  - Sophomore → ECE 201 and ECE 202
  - Junior → ECE 301 and ECE 302
  - Senior → ECE 401 and ECE 402

  Your method must also do the following

- Each time you register a student for classes, you must deduct some amount of money from their bank account.
- You must make sure that a student has enough money in their account to register for classes.
- You must request and verify the student's ID number before they can register for a class

- **parent Method**
  When a student runs out of money in their bank account, you must call this method and request for more money. You must provide an input value to this method. This value specifies how much money you need. Your method must then make a deposit to the students account. The amount deposited must be greater or equal to the amount requested.

  It would be nice to have an actual class called Parent to do this!!!

- **displayClasses Method**
  This method displays the classes this student is currently taking
- **__populateQuestionsAnswers Method**
  This method populates the **__finals** variable. The method must be called **only once**, the first time an instance of the Student class takes a final. When this happens, you are to set the value of the **Populated** key True, so that the variable is not re-populated.

  Remember, population of the **__finals** variable with values must be done only once.

- **takeFinals Method**
  When a student deices to take the finals for his/her year you must do the following
  - Make sure they have registered for classes
  - Makes sure they have not already graduated
  - Make sure to request for their student id and validate it
  - Call the **__populateQuestionsAnswers** if the finals have not already been populated
  - Get the students answers to the questions (5 questions per year)
  - Score the students answers, one point for each correct answer
  - Calculate a new gpa for the student, using whatever averaging formula you want
  - If the student scores more than half the allowed points of 5, promote them to the next year. If not, ask them to retake the class
  - Delete the contents of the **__currentClasses** variable
  - Advance the student to the next year. If the student passes the exam for the Senior year, promote them to Graduate

V. Logging In

Inside the BankAccount class, there is a method called **promptLogin.** Use this class to validate that the user has logged into their account. To do this, call this method first in all the methods of the Student class.

If the user has not already logged in, they would be prompted to login using the **login** method in the BankAccount class.

VI. Goodluck!!!!!