Lab 12-1 – Student Registration OOP

Goals

- Practice with object oriented programming
- Practice creating classes with methods and attributes
- Practice writing and using getters and setters
- Using instance variables and their associated methods

Setup

- In PyCharm, create a new project or open an existing one (such as Labs)
- Create a new Python file using the following naming convention:
  `ITP115_L12_1_`*`LastName_FirstName`*
  (replace *LastName* with your last/family name and *FirstName* with your first name)
- Your new file must begin with comments in the following format (replace the name and email with your actual information):

```
# Name, USC email
# ITP 115, Spring 2020
# Lab 12-1
```

Requirements

You will be creating a program that simulates a course registration system by modeling students with a **`Student`** class based off the following requirements:

- Create a class called **`Student`** with the following attributes and methods:
  - **Attributes**
    - **`name`**: string representing the student's name
    - **`idNumber`**: integer representing the student's ID number
    - **`courses`**: **`list`** of the courses the student is registered in
  - **Constructor Method**
    - **`__init__(studentName, studentID)`**
      - Parameters (2): name and ID number
      - Return value: none
      - Set the student's name and ID number. Set the courses list to an empty list (a student should not be registered for any courses initially).

- **Methods**
  - **`getName()`**
    - Parameters: none
    - Return value: the student's name
  - **`setName(newName)`**
    - Parameters (1): a new name
    - Return value: none
    - Set the student's name to the new name
  - **`getID()`**
    - Parameters: none
    - Return value: the student's ID number
  - **`setID(newID)`**
    - Parameters (1): a new ID number
    - Return value: none
    - Set the student's ID number to the new ID number
  - **`getCourses()`**
    - Parameters: none
    - Return value: the list of the student's courses
  - **`getNumberOfCourses()`**
    - Parameters: none
    - Return value: the number of courses the student is registered in (this is the length of the **course** list
  - **`addCourse(course)`**
    - Parameters (1): the name of the course being added
    - Return value: a boolean, indicating the success of adding the new course
    - Depending on whether or not the student has registered for the maximum number of courses, add the new course to the student's list of courses.
  - **`__str__()`**
    - Input: none
    - Return value: A string containing a message about the student's name, ID number, and what courses they are enrolled in.
- Additionally, use the following requirements to make create the registration system:
  - Create a global variable, **`MAX_COURSES`**, which has the value 6
  - Define a **`printStudents(studentList)`** function

- Input: a list of student objects
- Return value: none
- Print out the students' names in the list in a menu format by making use of the **`getName`** method for each student.

- Within **`main`**:
  - Create 4 student objects. You may hard code in their information (i.e. you do not need to ask the user to input any values or read in any values from a file to create the students). Put all of these students in a **`list`**.
  - In a **`while`** loop, ask the user to pick a student from the list. Then ask the user for the name of the course the student is adding. Add the course by calling the **`addCourse`** method on the selected student. Depending on whether the addition was successful or not, print out a corresponding message to the user.
  - At the end of the program, print all of the students.

Sample Output

```
Welcome to the student registration system!

Students:
      1) Tiffany
      2) Isaaca
      3) Huy
      4) Brandon
Select a student from the list (1-4): 1
Enter the course the student is registering for: ITP 115
Course registration successful.
Would you like to continue registering? (y/n): y

Students:
      1) Tiffany
      2) Isaaca
      3) Huy
      4) Brandon
Select a student from the list (1-4): 2
Enter the course the student is registering for: ITP 115
Course registration successful.
Would you like to continue registering? (y/n): y
```

```
Students:
      1) Tiffany
      2) Isaaca
      3) Huy
      4) Brandon
Select a student from the list (1-4): 3
Enter the course the student is registering for: ITP 115
Course registration successful.
Would you like to continue registering? (y/n): y

Students:
      1) Tiffany
      2) Isaaca
      3) Huy
      4) Brandon
Select a student from the list (1-4): 4
Enter the course the student is registering for: ITP 115
Course registration successful.
Would you like to continue registering? (y/n): n

Student: Tiffany, ID: 40 enrolled in 1 courses:
      - ITP 115
Student: Isaaca, ID: 41 enrolled in 1 courses:
      - ITP 115
Student: Huy, ID: 42 enrolled in 1 courses:
      - ITP 115
Student: Brandon, ID: 43 enrolled in 1 courses:
      - ITP 115
```

Deliverables and Submission Instructions

- Create a zip file containing your Python code. This cannot be done within PyCharm. Find the file or folder on your computer and compress it.
  a. Windows*:*
     1. Using File Explorer, select your lab file
     2. Right click
     3. Send to ->
     4. Compressed (zipped) folder
  b. Mac OSX:
     1. Using Finder, select your lab file
     2. Right click
     3. Compress "*FileName*"
- Upload the zip file to your Blackboard section:
  1. On Blackboard, click on the Labs item in the course menu on the left.
  2. Click on the specific item for this assignment (starts with L and a number).
  3. Click on the Browse My Computer button and select your zip file.
  4. Click the Submit button.