CIS611
Individual Practice Programming Assignment: PA06
Total Points: 20

Inheritance, Polymorphisms, and abstract classes

The purpose of this programming assignment is to:

- Develop subclass from super class through inheritance
- Override instance methods in the subclass
- Define abstract classes and classes that implement abstract methods

Q1 (20 points)**:**

Code an Address.java class that has an address, city, state, and zipCode attributes.

Code a Person.java class, which is an abstract class, means an instance of Person.java cannot be created. The Person.java has abstract methods, which need to be implemented by the two concrete sub-classes (by Sudent.java, and Faculty.java).

Code a Student.java class that has a status, and the values of the status are ((1) Freshman, (2) Sophomore, (3) Junior, or (4) Senior.) and also a subclass of Person. The status attribute is private with public getter and setter. Also code a toString() method by overriding the toString() form Person class, and concatenate its return from the parent class with the additional fields from the current subclass data fields.

Code a Faculty.java class that has a date hired (a reference data field to class `MyDate`.), has a rank ((1) Lecturer, (2) Assistant Professor, (3) Associate Professor, or (4) Professor.), and is a subclass of Person. A MyDate class needs to be coded with a java.util.date attribute, which is the hire date. MyDate will have myDate attribute with public getter and setter. Also code a toString() method by overriding the toString() form Person class. Also code a toString() method by overriding the toString() form Person class, and concatenate its return from the parent class with the additional fields from the current subclass data fields.

For the classes above, the attributes have to be private. The getter and the setter, and toString() methods are `public`.

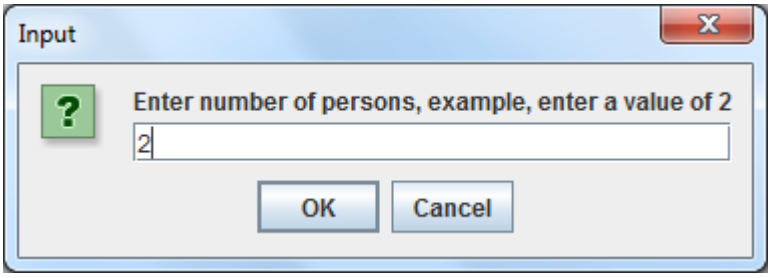
`Part 1 – Test Your Code by Hard-Coding Values (You will not be graded on Part 1)`

This assignment has two parts. The part 1 is testing your program by hard-coding the data inside a `TestByHardCoding`.java program. A partial implementation of `TestByHardCoding`.java is provided in the assignment folder and the students can use it for testing the code. Consider this for your practice and learning, and you will not be graded on Part 1.

Part 2 – Get Information from User Input (You will be graded on Part 2 only)

After you are satisfied with the part one (hard-coding using the `TestByHardCoding`.java), you will do the part two of this assignment, which will get the necessary information using the JOptionPane. For the part two, you will write a separate Java program (TestWithInput.java), which will accept the all the pieces of information using JOptionPane only (instead of hard-coding, as you did in part one). Following user input needs to be obtained using separate JOptionPane:

1. Number of Person. This number (n) will be used to set the size of `personArray. It is expected that the students test the code with at least with size 2, where one Faculty instance and one student instance is populated.`



2. For each person, get the following pieces of information for each Person (person could be either a faculty, or a Student) using JOptionPane:
    a. Information on Address (using this information, you will create an instance of Address
    b. Information on Date (using this information, you will create an instance of MyDate)
    c. Information on Faculty, or a Student (using this information, you will create an instance of Faculty). You may design your program to give a prompt to the user to decide on whether to enter the Student information, or the Faculty information. Based on the selection of the user, either the Faculty information, or the Student information will be entered by the user. And using the information entered, an instance of Faculty (or Student) will get created and will be used to populate an element of personArray.

Next, you will iterate through the elements of personArray and display the information of the person (you will use toString) in a new JOptionPane. Use line separator to separate the information of each person.

The program should not crash if input values are kept empty, or empty space is entered in Part 2. There is no need to code to check for any format of the phone number email address (they cannot be blank, or empty).

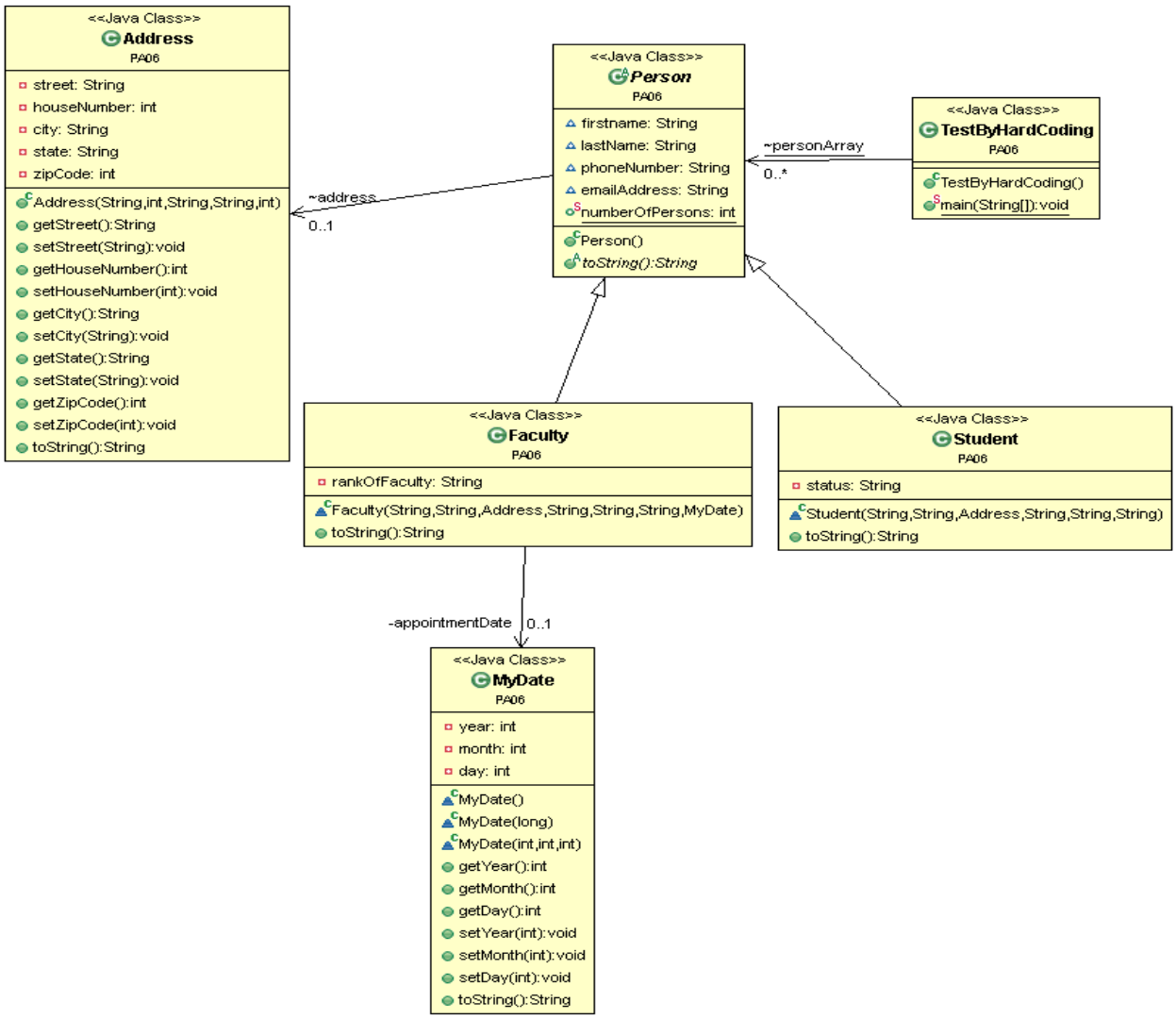
It is left to the creativity of the students the use of JOptionPane, how many JOptionPane to code, the user interface, and how to design the various prompts. As long as the students can obtain the necessary information from the user input using JOptionPane, it will suffice the assignment requirement. For example, the only difference between the Faculty information and the Student information is, Faculty has a Date of Appointment and Rank; while Student has Status. All other

pieces of information (Address, firstName, lastName, phoneNumber, email) are common to both. The program can be coded so that common pieces of information is re-used for both Faculty and Student, and when prompted for Faculty information, only Faculty specific information needs to be entered. It is left to the discretion of the students, the decision to design these various prompts.

Finally, you will submit both `TestByHardCoding`.java and TestWithInput.java, along with the other java classes (e.g., Address.java, Person.java, Student.java, Faculty.java, MyDate.java). A partial implementation of `TestByHardCoding.java is provided in the assignment folder, which the student can use.`

You will be graded only on Part 2 (where we expect the input to come from the users), although you are required to submit both Part 1 and 2.

Class Diagram:



Evaluation Criteria:

- The program must compile cleanly (no compile errors, but compile warnings are sometimes accepted)
- The program should not crash while running and it should terminate gracefully
- All tasks (requirements) in this assignment must be completed in order to receive credit
- The correct understanding and implementation (coding) of the requirements (programs should behave as anticipated):
    - The program must terminate with proper/correct outputs
    - All the logical relationship between classes should be performed correctly

Submission: (This is an individual Assignment!)

Copy the .java source files from the *src* folder in your *work space* to another folder that should be named following the provided naming format in this course, then zip and upload the file under this assignment answer in Canvas.

File Name: *FLLLLPA06.zip (F = first letter in your first name and LLLL = your last name)*

Grading Rubric PA06

Student Name: _____

Question 1

Requirements	Comment	Max Points Allowed	Points Earned
General Code Structure: Proper naming convention used for file (0.25) Comments used in the code to explain the purpose of the code (0.25) Indentation of the code for better readability (0.25) Good choice of the variable names (0.5)		1	
Input, Output, User Interface: Prompt the user to enter the number of persons using JOptionPane (2) Prompt the user to enter either the Faculty data, or Student data using JOptionPane (2)		10	

If the Faculty data, then get the Date, Address, and the Faculty information using JOptionpane (2) If the Student data, then get the Address, and the Student information using JOptionpane (2) Display the Faculty and Student in a JOptionPane, and put a line separator between each Student/Faculty information (2)			
General Algorithm and Logic: Coding of the Person, Student, Faculty, Address, MyDate, classes with the subclass hierarchy (1) Override the toString() method in Faculty.java and Student.java (1) Set the size of Code personArray from the user input of the person size (1) Populate the personArray with instances of Faculty (2) Populate the personArray with instances of Student (2) Iterate through personArray and display the information of Faculty and Person in JOptionPane. (2)		9	
Total		20	

Total ____/20