Langara
THE COLLEGE OF HIGHER LEARNING.

Department of Computing Science & Information Systems
CPSC 1181

Lab#3
Sep 29, 2022

Objectives:

- Create subclasses
- Use visibility modifiers to maintain encapsulation during inheritance
- Override toString() methods from super classes in subclasses

Preparation:

Read chapter 9 of the textbook.

Due date:

Due Date: 11:00 PM on Wednesday Oct 5, 2022

Where to upload:

Zip your files into yourstudentID.zip where yourstudentID is your student number, and upload it to dropbox lab3 in D2L.

Part A: Tutorial

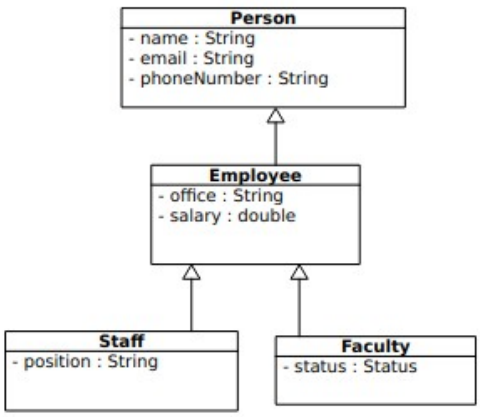
Do tutorial . Due date 11:00 PM Thursday Sep 29, 2022. Late submission will not be accepted.

You should finish tutorial and upload your files to tut2 drop box in D2L by due date.

There is no extension for tutorials. After 11:00PM, the drop box will be closed.

Part B:

In this lab assignment you are going to be implementing the inheritance hierarchy shown below:



B1)

enum Status:

enum Status{TEMPORARY,PART_TIME, FULL_TIME};

We will use this enum structure in our code. If you have not used enum before, refer to this tutorial.

Class Person has following instance fields:

name: String

email: String

phoneNumber: String

Class Employee extends class Person with following instance fields:

office: String

salary: double

The default office and salary for Employee are “A101”, and 40000 respectively;

class Faculty extends class Employee with following instance field:

status: Status

The default status for Faculty object is TEMPORARY

Class Staff extends class Employee with following instance field:

position: String

The default position for a staff object is “general”.

The detail UML class diagram of the classes is shown on the next page.

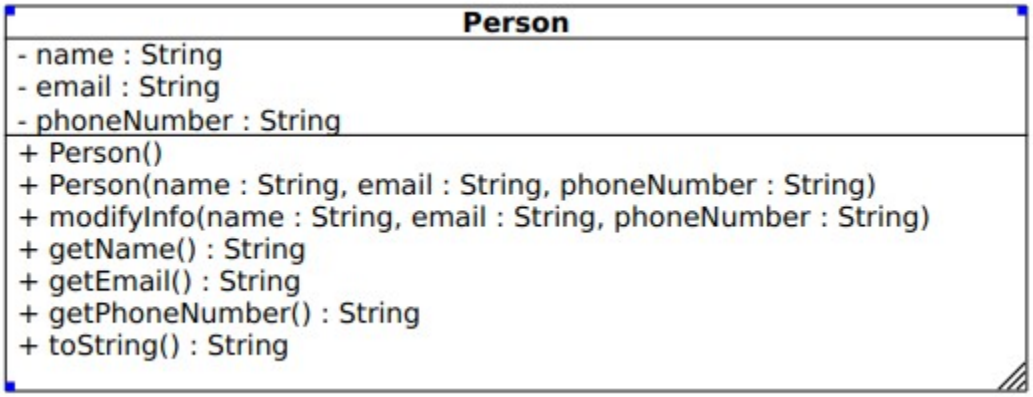
You should develop all methods shown in the UML diagram.

- sign in UML stands for private

+ sign in UML stands for public

B2)

Develop class Person. The details of the class Person is shown below:



Class includes two constructors,

A mutator method modifyInfo(….) that set new values for the instance fields.

Three access methods, getName(), getEmil(), and getPhoneNumnber()

Override the toString() method of the class. For the format of the toString() method refer to the sample run of the program at the end of this document.

B3)

Develop class Employee. The details of the class Employee is shown below:

Employee
- office : String - salary : double
+ Employee() + Employee(name : String, email : String, phoneNumber : String) + Employee(name : String, email : String, phoneNumber : String, office : String, salary : double) + setOffice(office : String) + setSalary(salary : double) + getSalary() : double + getOffice() : String + toString() : String

Class includes three constructors,

Two mutator methods setOffice(….) and setSalary(…) that set new values for the instance fields.

Two accessor methods, getSalary() and getOffice()

Override the toString() method of the class. For the format of the toString() method refer to the sample run of the program at the end of this document.

B4)

Develop class Faculty. The details of the class Faculty is shown below:

Faculty
- status : Status
+ Faculty() + Faculty(name : String, email : String, phoneNumber : String) + Faculty(name : String, email : String, phoneNumber : String, office : String, salary : double, status : Status) + setStatus(status : Status) + getStatus() : Status + toString() : String

Class includes three constructors,

One mutator method setStatus(….) that set new values for the instance field.

One accessor method, getStatus()

Override the toString() method of the class. For the format of the toString() method refer to the sample run of the program at the end of this document.

B5)

Develop class Staff. The details of the class Staff is shown below:

Staff
- position : String
+ Staff() + Staff(name : String, email : String, phoneNumber : String) + Staff(name : String, email : String, phoneNumber : String, office : String, salary : double) + setPosition(position : String) + getPosition() : String + toString() : String

Class includes three constructors,

One mutator method setPosition(….) that set new values for the instance fields.

One accessor method, getPosition()

Override the toString() method of the class. For the format of the toString() method refer to the sample run of the program at the end of this document.

B6)

Download sample test harness and output result for the set of the classes; TestLab3.java , out.txt

Add more test cases as required.

The output of the test harness is shown below. Discover and follow the correct toString() format of the classes.

Important Notes:

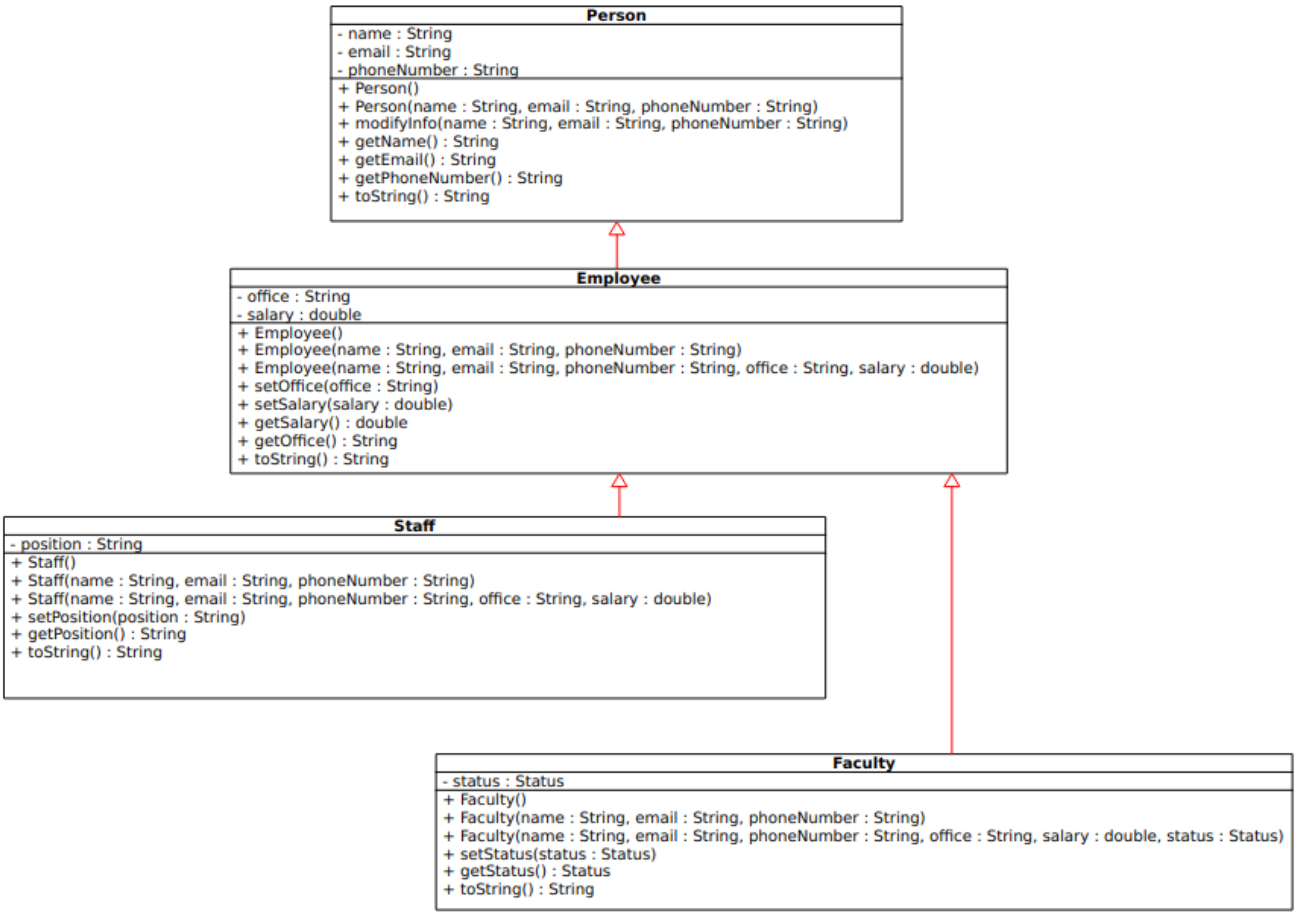
1. Use principal of inheritance and encapsulation. You will get poor mark if you do not used them properly, even if your program produces correct output.

2. Do not develop redundant code
3. Use appropriate methods of the super classes in all cases including toString() methods.
4. You will be marked based on correctness and quality of your code.
5. You do not need to document test cases since we have learned it in the first lab assignment. Just create appropriate test cases.
6. Follow Java style code in your coding.

What to submit

1. Comment your classes and methods appropriately using the javadoc notation.
2. Comments about your assignment if needed. These comments are not the comments documenting your code but rather something you need to convey to us about your assignment
3. Submit to D2l Dropbox: lab3

Complete Class diagram:



Total Mark: 50