Lab 05- 40 points

For today's lab you will be implementing two of your own inheritance hierarchies. It is a smaller/simpler lab, with an earlier deadline. The expectation being that you will also use this time to start your Project #1.

Part 1 (20 Points)

Implement a superclass `Person`.
Make two classes, `Student` and `Instructor`, that inherit from `Person`.

- A person has a name and a year of birth (2 points)
- A student has a major (2 points)
- An instructor has a salary (2 points)

Write

- The class declarations (1 point each x 3 = 3 points)
- The appropriate constructors (2 points each for sub x 1, 1 point for super= 5 points)
  - Make one for each class that initializes all the data and utilizes the "Super(…)" constructor appropriately.
    - That is, the student and instructor constructors will call the super constructor with the appropriate data to initialize the person's instance variables.
    - For example, the student and instructor constructors will need the information necessary (parameters) that includes person information
- The (appropriate and overridden) `toString` methods for all classes. (3 points)
  - For each class' implementation of `toString`, use (extend) behavior from each super class' toString's method in order to avoid duplicate code (Negative points for duplicate code)!
- Getters and Setters (Negative points if done wrong)

Supply a single test program that tests these classes and methods, including the constructors. (2 points)

Part 2 (20 points)

- Make a class `Employee` with a name and salary. (2 points)
- Make a class `Manager` that extends `Employee`.
  - Add an instance variable for type `Manager`, named `department`, of type `String`.
    (4 points)
- Make a class `Executive` inherit from `Manager`.
  - An Executive has a String that represents their office location.
    (3 points)
- Implement the appropriate constructors (2 points each for sub x 1, 1 point for super= 5 points)
  - Make one for each class that initializes all the data and utilizes the "super(…)" constructor appropriately.
  - That is, the Manager constructor will call the super constructor with the appropriate data to initialize the Employee's instance variables. Same for executive.
- Supply appropriate `toString` methods for all classes. (4 points)
  - For each implementation of `toString`, use (extend) behavior from each super class' toString's method in order to avoid duplicate code. (Negative points if not done)
  - For example, override the method `toString` to print the Manager's name, department, and salary.
  - The Executive toString will include Manager and Employee information as well.
- Include Getters and Setters (Negative points if done wrong)

Supply a single test program that tests these classes and methods, including the constructors. (2 points)

Tips and Additional Requirements

1) You do not have to test your getters and setters.

2) You do have to test your constructors. You can do that by

   1. Creating the object
   2. Testing that the instance variables were set correctly by the constructor. You can do this using the getters and setters.

You need not use the automated/generated "toString" text that Eclipse provides. You can change it in your various classes to be something more aesthetically pleasing, for example,
“I am a student majoring in X. I am also a Person whose name is A and was born in B.”
All we will be checking for is that you output all the information at each level and use the super call appropriately.

Submission

Turn in a .zip of your (4+4) 8 .java files to Canvas by your respective deadline. Ensure you have actually submitted (received confirmation and checked online), since no late labs will be accepted.

We will base grading on your conforming to the standards we reviewed in class, as well as your ability to follow the requirements of this lab. Points will be deducted for unnecessary duplicate code, file/class naming errors, poor commenting, and style (always auto format).