CPSC 131: Introduction to Computer Programming II

Program 3: Inheritance and Interface

1 Description of the Program

In this assignment, you will make two classes, `Student` and `Instructor`, that inherit from a superclass `Person`. The implementation of class `Person` is given. You will also need to write a test program to test the methods you write for these two classes. The implementation details are described as follows.

Stage 1: In the first file `Student.java`, you should include the following additional instance variables and methods (other than all instance variables and methods inherited from class `Person`):

- Private instance variables `studentID`, and `major`;
- A constructor takes four inputs (`name`, `age`, `studentID` and `major`);
- Two additional `getter` methods to return each of instance variables (accessor);
- Two `setter` methods to change each of instance variables (mutator);
- A method `toString` that converts a student's information into string form. The string should have the format as shown in Figure 1. You should **override** superclass `toString()` method.
- A method `compareTo` that implements the interface `Comparable`, so that `Student` objects can be sorted by `studentID` in an ascending order.

Stage 2: In the second file `Instructor.java`, you should include the following additional instance variables and methods (other than all instance variables and methods inherited from class `Person`):

- Private instance variable `salary`;
- A constructor takes three inputs (`name`, `age`, and `salary`);
- One additional `getter` method to return the instance variable (accessor);
- One `setter` method to change the instance variable (mutator);

- A method `toString` that converts an instructor's information into string form. The string should have the format as shown in Figure 1. Specifically, you need to format `salary` value to 2 decimal places, and make them right aligned. You should also override superclass `toString()` method.
- A method `compareTo` that implements the interface `Comparable`, so that `Instructor` objects can be sorted by `salary` in an ascending order.

Stage 3: In the third file `PersonTester.java`, you will need to do the followings:

1. You need to read data file "`data1.txt`" into an array of `Student` objects, Specifically,
   (a) The first number in the first line of the file is used to determine the array size.
   (b) The remaining lines are the student records. You will need to create a `Student` object using each line's information, and put it into the array.
2. Do sorting of the initialized array of `Student` objects, and then print them out. The outputs should be nicely labeled and formatted, as shown in Figure 1.
3. Repeat the above steps to read data file "`data2.txt`" into an array of `Instructor` objects, do sorting and print them out. The outputs should be nicely labeled and formatted, as shown in Figure 1. To avoid code redundancy, you may use a loop to handle the repetitive procedure.

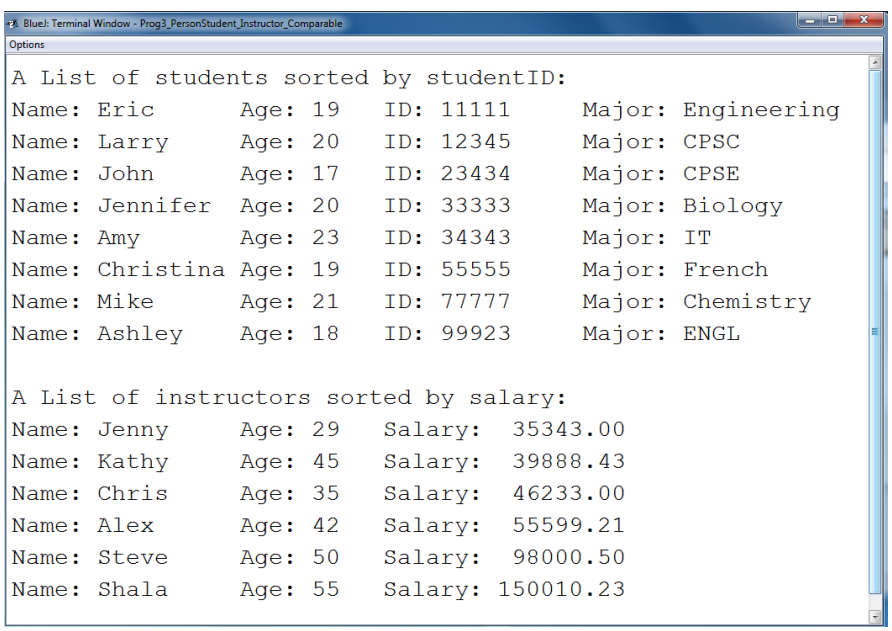

Figure 1: A screenshot of the program output.

2 Submission

Upload the following items on D2L dropbox, including:

1. The source code (`Student.java`, `Instructor.java` and `PersonTester.java`).
2. Screenshot of your program output (Similar to sample output shown in Figure 1).