CPSC 131: Introduction to Computer Programming II

Program 3: Inheritance and Interface

1 Description of the Program

In this assignment, you will write **five** classes:

- `Student` class: This class inherits from a superclass `Person` (given to you).
- `Instrucor` class: This class inherits from a superclass `Person`.
- `UniqueStudentList` class: This class is used to store a list of non-redundant students.
- `UniqueInstructorList` class: This class is used to store a list of non-redundant instructors.
- `PersonTester` class: This class handles reading dataset into `UniqueStudentList` and `UniqueInstructorList` objects, and printing them out in a sorted order.

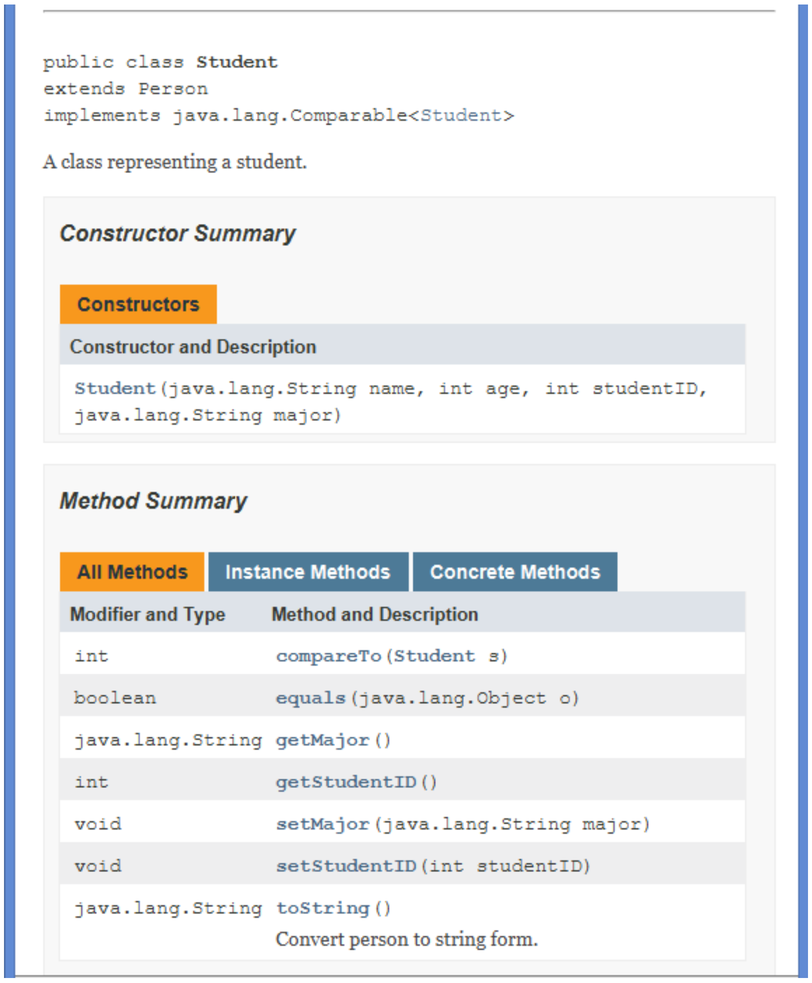
The implementation details are described as follows.

1.1 Student class

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 5. 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.
- A method `equals` that compares this student's information with another object's information. Return `true` if they are same, `false` if they are not.

The summary of the `Student` class is given below.



```
public class Student
extends Person
implements java.lang.Comparable<Student>
```

A class representing a student.

Constructor Summary

Constructor and Description
`Student(java.lang.String name, int age, int studentID, java.lang.String major)`

Method Summary

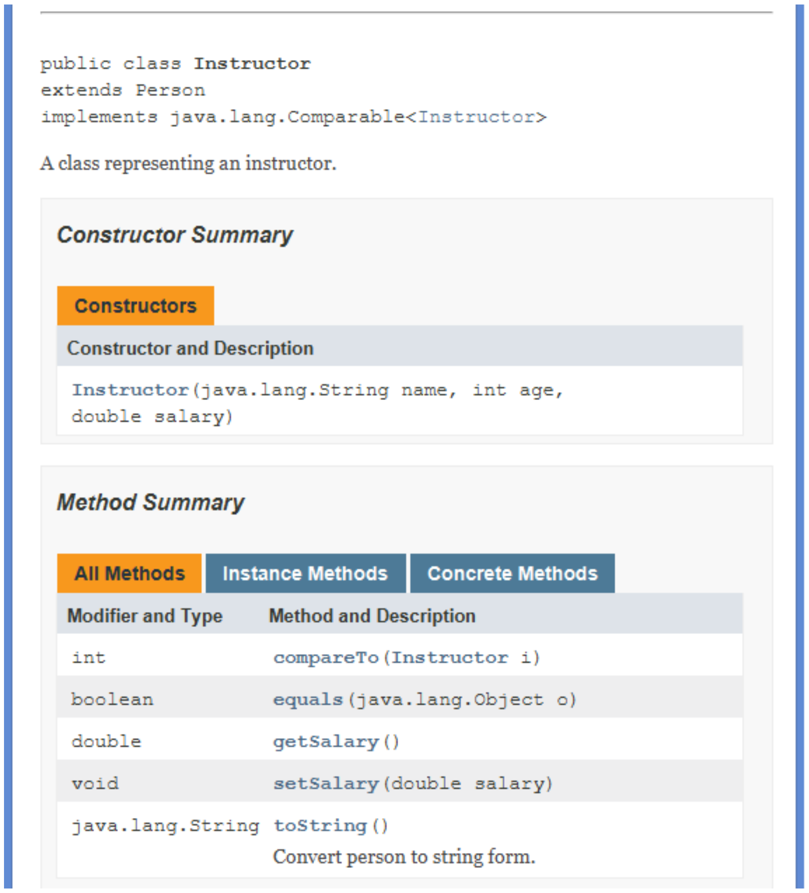
Modifier and Type	Method and Description
`int`	`compareTo(Student s)`
`boolean`	`equals(java.lang.Object o)`
`java.lang.String`	`getMajor()`
`int`	`getStudentID()`
`void`	`setMajor(java.lang.String major)`
`void`	`setStudentID(int studentID)`
`java.lang.String`	`toString()` Convert person to string form.

Figure 1: Summary of the Student Class.

1.2 Instructor class

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 5. Specifically, you need to format


```
public class Instructor
extends Person
implements java.lang.Comparable<Instructor>
```

A class representing an instructor.

Constructor Summary

Constructors

Constructor and Description
`Instructor(java.lang.String name, int age, double salary)`

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
`int`	`compareTo(Instructor i)`
`boolean`	`equals(java.lang.Object o)`
`double`	`getSalary()`
`void`	`setSalary(double salary)`
`java.lang.String`	`toString()` Convert person to string form.

Figure 2: Summary of the Instructor Class.

`salary` value to 2 decimal places, and make them right aligned. You should also override superclass `toString()` method.

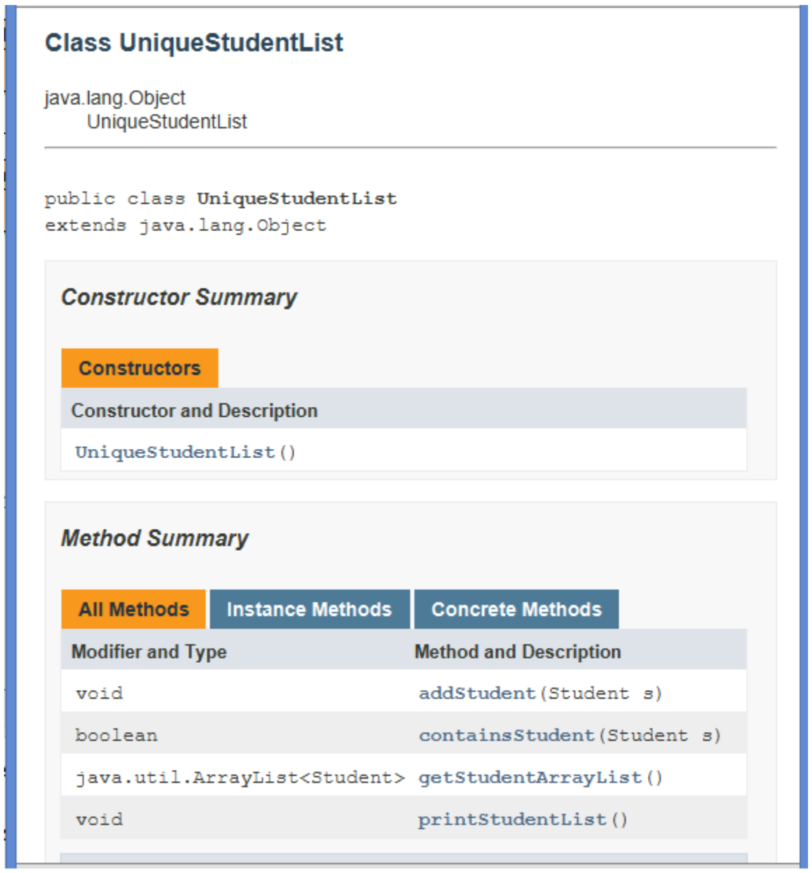
- A method `compareTo` that implements the interface `Comparable`, so that `Instructor` objects can be sorted by `salary` in an ascending order.
- A method `equals` that compares this instructor's information with another object's information. Return `true` if they are same, `false` if they are not.

The summary of the `Instructor` class is given below.

1.3 UniqueStudentList class

In the third file `UniqueStudentList.java`, you may include the following instance variables and methods:

- Private instance variable `studentArrayList` (type: `ArrayList<Student>`);
- A default constructor to initialize `studentArrayList` ;



Class UniqueStudentList

java.lang.Object
    UniqueStudentList

```
public class UniqueStudentList
extends java.lang.Object
```

Constructor Summary

Constructor and Description
UniqueStudentList()

Method Summary

Modifier and Type	Method and Description
void	addStudent(Student s)
boolean	containsStudent(Student s)
java.util.ArrayList<Student>	getStudentArrayList()
void	printStudentList()

Figure 3: Summary of the UniqueStudentList Class.

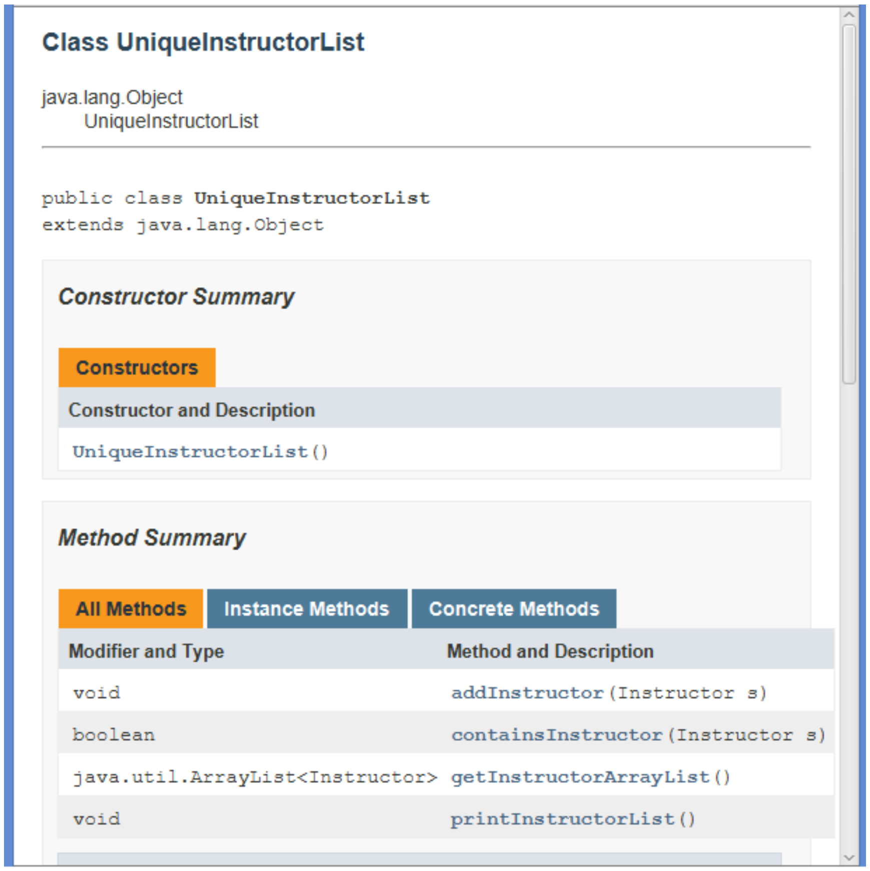
- One `getter` method to return the instance variable (accessor);
- A method `addStudent` that add a student into `studentArrayList`. Make sure you only add it if `studentArrayList` does not contain this new student object. Otherwise, you will not add it to the list.
- A method `containsStudent` that is used to check whether the input student is in `studentArrayList` or not. This method may be called within the `addStudent` method.

  Note: You really don't have to implement this method if you don't want. You can do the checking process within the `addStudent` method. It makes your implementation cleaner if adding this method.

1.4 UniqueInstructorList class

In the fourth file `UniqueInstructorList.java`, you may include the following instance variables and methods:

- Private instance variable `instructorArrayList` (type: `ArrayList<Instructor>`);


Class UniqueInstructorList

java.lang.Object
    UniqueInstructorList

```
public class UniqueInstructorList
extends java.lang.Object
```

Constructor Summary

Constructors

Constructor and Description
`UniqueInstructorList()`

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
`void`	`addInstructor(Instructor s)`
`boolean`	`containsInstructor(Instructor s)`
`java.util.ArrayList<Instructor>`	`getInstructorArrayList()`
`void`	`printInstructorList()`

Figure 4: Summary of the UniqueInstructorList Class.

- A default constructor to initialize `instructorArrayList` ;
- One `getter` method to return the instance variable (accessor);
- A method `addInstructor` that add an instructor into `instructorArrayList`. Make sure you only add it if `instructorArrayList` does not contain this new Instructor object. Otherwise, you will not add it to the list.
- A method `containsInstructor` that is used to check whether the input instructor is in `instructorArrayList` or not. This method may be called within the `addInstructor` method.

  Note: You really don't have to implement this method if you don't want. You can do the checking process within the `addInstructor` method. It makes your implementation cleaner if adding this method.

1.5 PersonTester class

In the fifth file `PersonTester.java`, you will need to do the followings:

1. You need to read data file "`data1.txt`" into the `UniqueStudentList` object, Specifically, for each line you read, you need to create a `Student` object, and then add the student into the student list.

2. Do sorting on the student array list, and then print out a sorted student list (sorted by `studentID`). The outputs should should include the followings:

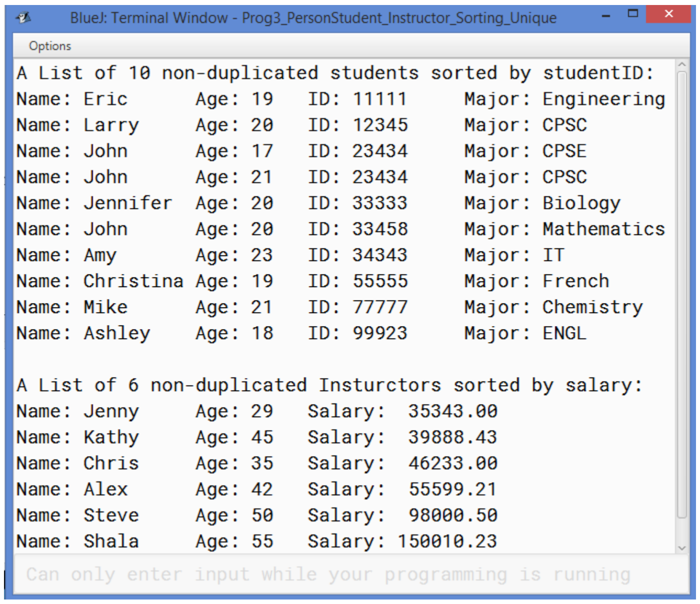
   (a) Student list information: including **number** of non-duplicated students;

   (b) Detailed student list through calling the method of `printStudentList`.

   be nicely labeled and

3. Repeat the above steps to read data file "`data2.txt`" into `UniqueInstructorList` object.

Your final program output should look like Figure 5.


BlueJ: Terminal Window - Prog3_PersonStudent_Instructor_Sorting_Unique
Options

```
A List of 10 non-duplicated students sorted by studentID:
Name: Eric      Age: 19   ID: 11111     Major: Engineering
Name: Larry     Age: 20   ID: 12345     Major: CPSC
Name: John      Age: 17   ID: 23434     Major: CPSE
Name: John      Age: 21   ID: 23434     Major: CPSC
Name: Jennifer  Age: 20   ID: 33333     Major: Biology
Name: John      Age: 20   ID: 33458     Major: Mathematics
Name: Amy       Age: 23   ID: 34343     Major: IT
Name: Christina Age: 19   ID: 55555     Major: French
Name: Mike      Age: 21   ID: 77777     Major: Chemistry
Name: Ashley    Age: 18   ID: 99923     Major: ENGL

A List of 6 non-duplicated Insturctors sorted by salary:
Name: Jenny     Age: 29   Salary:  35343.00
Name: Kathy     Age: 45   Salary:  39888.43
Name: Chris     Age: 35   Salary:  46233.00
Name: Alex      Age: 42   Salary:  55599.21
Name: Steve     Age: 50   Salary:  98000.50
Name: Shala     Age: 55   Salary: 150010.23
 Can only enter input while your programming is running
```

Figure 5: A screenshot of the program output.

2 Submission

Upload the following items on D2L dropbox, including:

1. A zipped file containing the source code of all java files).

2. Screenshot of your program output (Similar to sample output shown in Figure 5).