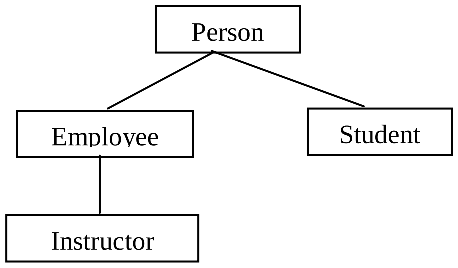
# CIS 459.23 Lab 2

Due Oct 30 (Sunday)

OSU wants you write some classes for their Personnel Record System. To make it simple, consider only 4 classes: Person, Employee, Instructor and Student. The following figure illustrates the relationship between these 4 classes. The Person class is the parent class of the Employee class and the Student class. The Employee class is the parent class of the Instructor class.



The following are the tasks you need to complete for each of these classes.

- Create appropriate fields for each class. Necessary fields are listed. Add your own fields if needed. Some fields need to have appropriate constraint. Use your own way to make sure that these constraints are satisfied.
  - **Person**
    - *ID*: int, starting from 1 and should be unique
    - *Name*: String
  - **Employee**
    - *Salary*: double and should not be negative
  - **Student** (For simplicity, assume that a student has at most 1 teacher)
    - *TeacherID*: int. It's his/her instructor's ID. 0 if no instructor is given
    - *TeacherName*: String
  - **Instructor**:
    - *StudentIDArray*: int array. An array of students' IDs of this instructor. Set the array size to be 10, initially all 0s, assuming an instructor won't have more than 10 students.
- All the above fields are private and only accessible through the access methods.
- A "toString()" method for each class to print out all the available information about the current object. In Person class "toString()" is declared as abstract.
- A static "findStudents(Person[] personArray)" method in the Instructor class to fill an instructor object's students ID array, and the corresponding students' TeacherID fields. See the test program for better understanding.
- Person should be declared as abstract class.
- Provide multiple constructors/methods if needed. Check the test.java program to see what constructors/methods are necessary and what actions they should do.
- If a class can use the parent class method and constructor, use "super" to call it to reduce the redundant code.
- Make sure this test.java program can work with your class.
- sample output. From this sample output, you'll know what information you should print out for a specific object.

NOTE: the sample output is not the unique output format of the test program. The real output format depends on how you design the toString() methods in each class. But make sure that your program will print out as much information about each object's fields as possible, including the inherited fields and the fields defined in its own class.

HINT:

- There is NO main method in any of these 4 classes
- To make sure ID is unique across the objects, declare a static “LAST_ID” in the Person class.
- **Read descriptions in test.java VERY CAREFULLY for better understanding!**

Submit your Person.java, Emloyee.java, Student.java and Instructor.java files

Appendix 1: Test Program

```
/*
 * Lab 2 Program to test the Person, Employee, Student, and Instructor classes.
 */

public class Lab2_Test
{
   public static void main(String[ ] args)
   {
      // uncommenting the following line should produce a compile error.
      // This is for testing of an abstract class.
      // Person p = new Person("George");

      final int MAX_HEADCOUNT = 20;
      Person[] person_array = new Person[MAX_HEADCOUNT];

      // A student named Peter
      person_array[0] = new Student("Peter");

      // An instructor named Peter
      person_array[1] = new Instructor("Peter");

     // An instructor named Sandy and her salary
      person_array[2] = new Instructor("Sandy", 25000);

     // A janitor named Bob
     person_array[3] = new Employee("Janitor Bob");

     // A student named Tom and his instructor is Peter.
     // The constructor needs to do three things:
     // 1: sets this student’s “TeacherName” field to be “Peter”,
     // 2: finds out the ID of the 1st instructor
     //     who exists in the person_array so far and named "Peter",
     //     and assign it to this student's “TeacherID” field.
     //     Set it to be 0 if no instructor named Peter is found in the person_array so far
     // 3: records this student’s ID in the instructor’s StudentArray if such an instructor is found
     // right after executing the following statement
     // person_array[4].TeacherID = 2
     // person_array[4].TeacherName = “Peter”
     // person_array[1].StudentArray[0] = 5
     person_array[4] = new Student("Tom", "Peter", person_array);

     // A student named Maggie and her instructor is Susan
     // right after executing the following statement
     // person_array[5].TeacherID = 0
     // person_array[5].TeacherName = “Susan”
     person_array[5] = new Student("Maggie", "Susan”, person_array);
```

```
        // An instructor named Susan and her salary
         person_array[6] = new Instructor("Susan", 40000);

        // After all objects are created,
        // instructors need to fill their students arrays,
        // and some students need to fill their TeacherIDs now,
        // since there may exist cases that when a Student object is created with instructor’s name,
        // the corresponding Instructor object hasn’t been created and is not in the person_array.
        // For example, person_array[6] is created after person_array[5].
        // You need to record person_array[5]’s ID in person_array[6]’s studentArray field,
        // and record person_array[6]’s ID in person_array[5]’s TeacherID field.
        // Note: if there are more than one Instructor objects
        //          having the same names as a Student object’s TeacherName,
        //          it’ll always be the first one’s ID assigned to the Student object’s TeacherID
        Instructor.findStudents(person_array);

        System.out.println("ID and name of all personnel in the array");
        for (int i = 0; i < Person.getMaxID(); i++)
        {
            System.out.println(person_array[i].getID() + ":" + person_array[i].toString());
         }
    }
}
```

Appendix 2: Sample Output

```
ID and name of all personnel in the array
1:Peter is a student
2:Peter is an instructor earning a salary of 0.0 with 1 student in his/her class
   Student[0]: 5
3:Sandy is an instructor earning a salary of 25000.0 with 0 student in his/her class
4:Janitor Bob is an employee earning a salary of 0.0
5:Tom is a student whose instructor is Peter, and his/her instructor's ID is 2
6: Maggie is a student whose instructor is Susan, and his/her instructor’s ID is 7
7: Susan is an Instructor earning a salary of 40000 with 1 student in his/her class
    Student[0]: 6
```