Introduction to Java

CS9053

Thursday 6 PM – 8:30 PM

Prof. Dean Christakos

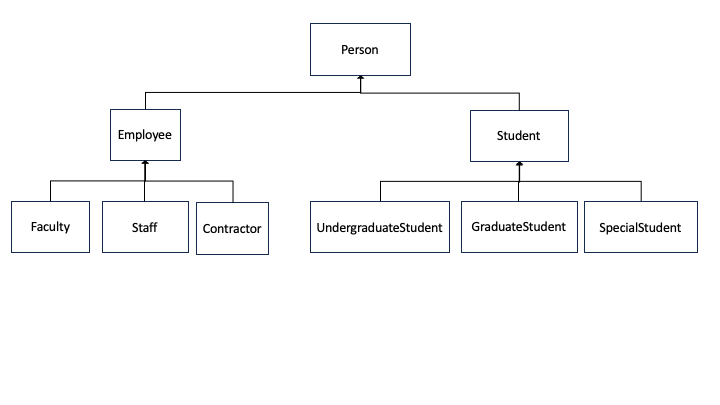
June 14th, 2024

Due: June 21st, 2024 11:59 PM

**Assignment 4**

Part I: Inheritance

1. Here is a class hierarchy diagram for school personnel. I won’t map out the exact UML specifications, because that would make it too easy.



Instead, here’s some descriptions of each class, and you can put everything together:

Person: name, age, gender, and id (auto-generate the ID)

Employee: salary, title, and startDate

Faculty: tenured (true/false), department

Staff: manager

Contractor: contractCompany, endDate

Student: department

UndergraduateStudent: graduationYear

GraduateStudent: advisor (a faculty member)

SpecialStudent: endDate

There should be appropriate getters and setters for each of these fields as well as toString() methods.

1. SpecialStudent sometimes re-register and get double counted in the system, even if they have a new ID. Implement an equals() method for SpecialStudent using its endDate field for comparison, but also implement the equals() methods for the Student class and the Person class so that a SpecialStudent.equals() call will propagate up the super classes and match all fields except ID to check for equality.
2. For GraduateStudent and UndergraduateStudent, implement equals() methods for them as well.

Part II – Arraylists

1. Create a faculty member (this will be important in a moment):

Faulty: James Kaiser, 45, ‘M’, 130,000, ‘Associate Professor’, 2020-09-01, ‘Electrical Engineering’, tenured (ie, true), “Electrical Engineering”

Create some students:

Undergraduate Student: John Smith, 20, ‘M’, “Computer Science”, graduation year 2026

Undergraduate Student: Amy Ferguson, 22, ‘F’, “Chemistry”, graduation year 2024

Graduate Student: Margaret Johnson, 25, ‘F’, “Electrical Engineering”, advisor James Kaiser

Now, you will notice that there is a class called “University.”

You need to update this by creating, as a field, an ArrayList of Student objects. It should only take students.

Then you should implement the method registerStudent. This takes a student object and places them in the ArrayList. Make sure you don’t register a student twice.

Register the students you’ve just created.

(I realize lists aren’t the optimal data structure for this, but we haven’t learned the other types of collections, so go with it)

Next, implement **two** methods called “studentRegistered()” which return a boolean. One should take a Student object and see if it exists in your arraylist field, and the other should take an integer id and see if a student object with that id exists in the ArrayList. That’s an overloaded method.

Implement **two** methods called deregisterStudent that removes a student from the ArrayList. One should take a Student object as an argument, the other takes an integer ID. It returns true if the object exists to be removed and succeeds and false if it does not.

Margaret Johnson has finished her Master’s degree. Deregister her.

Finally, implement printAllStudents() which prints out all the registered students.