Object-Orientation & Inheritance

Table of Contents

1. Session 4 (lab): Object-Orientation & Inheritance (II)

1.1. Class Hierarchies and Polymorphism

You have been asked to code a Windows application to manage the users of the UC3M swimming pool.

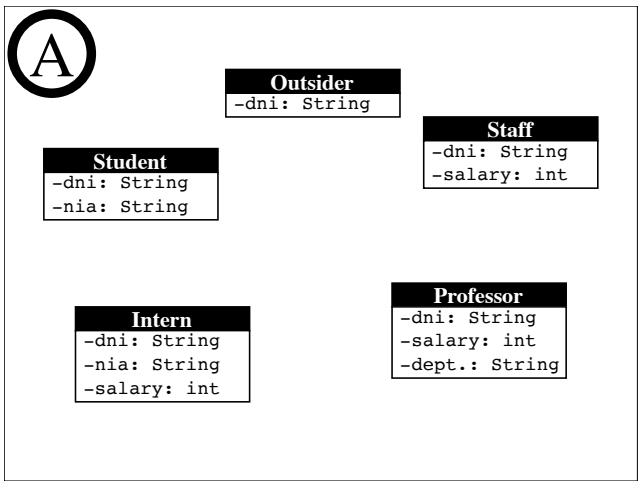
The application should be a very simple database system, which would store the data of all the swimming pool users. It must be capable of listing all the users and their data.

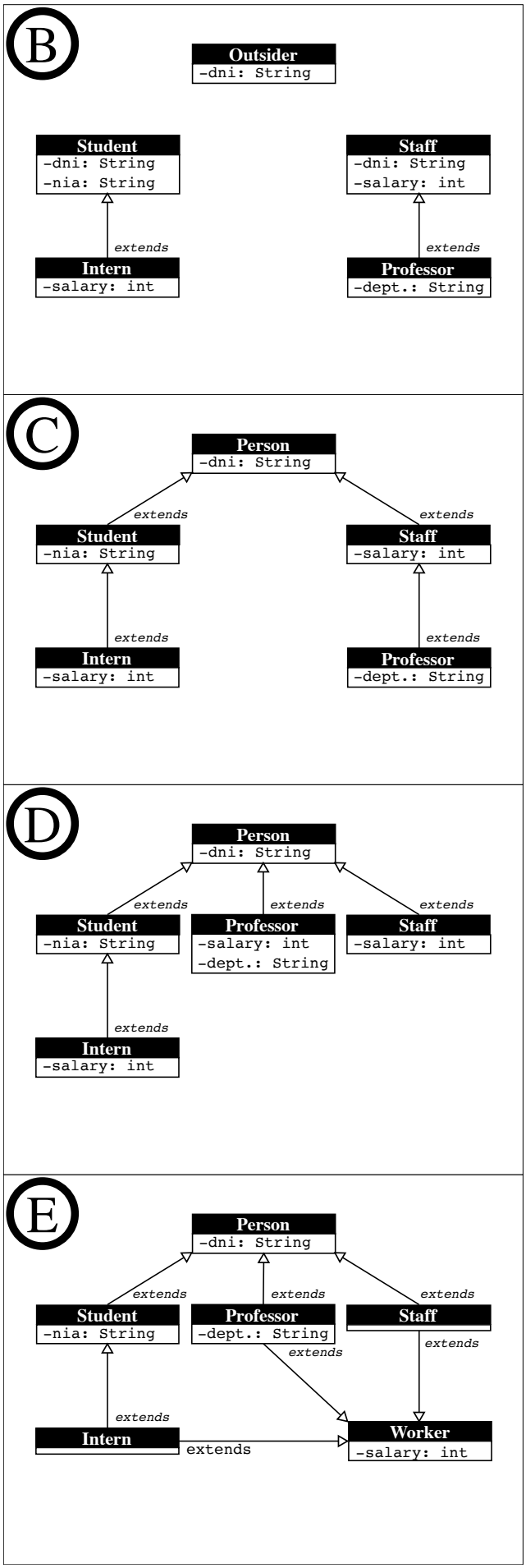
There are 5 types of swimming pool users:

1. *Outsider*: people not related to the UC3M, e.g. people living in Leganés. The only datum we know from them is their DNI (the Spanish national identity card).
2. *Staff*: Office workers and support people on the campus, but not teachers. We know their DNI and their salary.
3. *Professor*: UC3M teachers. We know their DNI, their department and their salary.
4. *Student*: We know their DNI and their NIA (Student identification number).
5. *Intern*: Regular students that are also working for the university. We know all their student data and also their salary.

Section 1. Class Hierarchy

Which of the following diagrams best represent the class hierarchy of the application? Discuss the pros and cons of all diagrams with a classmate for 15 minutes.





Section 2. Polymorphism

Implement the application using diagram D as a reference. Code all the classes. Each class must declare all its attributes, a constructor and also a method `String toString()` to generate a textual representation of the known data of each user, according to the format below.

You must also code a test class, with only the `main` method. It should instantiate objects representing each user (listed below) to test the application. You must store all the user objects in a variable that can hold several `Person` objects (any Java collection will do, e.g. an `ArrayList<Person>`...). Then go through all the users in the collection, printing all their data to the standard output.

The expected behavior of your application must be like this (the order in which the users are printed is not relevant):

```
C:\Users\Alberto>java DataBase
DNI: 01100000-A
DNI: 00220000-B
DNI: 00030000-C, salary: 2000
DNI: 04040000-D, salary: 1500
DNI: 50500000-E, salary: 1000, department: mathematics
DNI: 66600000-F, salary: 2000, department: telematics
DNI: 77000000-G, NIA: 777777
DNI: 88080000-H, NIA: 888888
DNI: 90990000-I, NIA: 999999, salary: 400
DNI: 10100000-J, NIA: 101010, salary: 800
```

Section 3.

1. Discuss with a classmate how is it possible to print all user data (data from `Staff`, `Student`... objects) if all of them are handled through references to a simple `Person` object.

2. Discuss with a classmate why do you need to declare the `String toString()` as a public method on `Person` and all its subclasses. Would it be possible to declare that method package-private instead, for example?

Section 4. Last minute changes

Just before submitting your application, the UC3M asks you to implement 2 extra functionalities:

- Add a new user type *Tenured*. They are like standard `Professor`s but have a fixed salary of 2500 Euros. This means that the class constructor should not have a salary argument. Also add these two instances of tenured professors to the database:

  ```
  DNI: 11110000-K, salary: 2500, department: geography
  DNI: 12120000-L, salary: 2500, department: mathematics
  ```

- The application must support a command line argument "`-s`" for generating "short" listings, this is, only the basic data from the users must be printed (only the data available from the class `Person`).

The expected behavior of your application must be like this:

```
; java DataBase
DNI: 01100000-A
DNI: 00220000-B
DNI: 00030000-C, salary: 2000
DNI: 04040000-D, salary: 1500
DNI: 50500000-E, salary: 1000, department: mathematics
DNI: 66600000-F, salary: 2000, department: telematics
DNI: 77000000-G, NIA: 777777
DNI: 88080000-H, NIA: 888888
DNI: 90990000-I, NIA: 999999, salary: 400
DNI: 10100000-J, NIA: 101010, salary: 800
DNI: 11110000-K, salary: 2500, department: geography
DNI: 12120000-L, salary: 2500, department: mathematics
; java DataBase -s
DNI: 01100000-A
DNI: 00220000-B
DNI: 00030000-C
DNI: 04040000-D
DNI: 50500000-E
DNI: 66600000-F
DNI: 77000000-G
DNI: 88080000-H
DNI: 90990000-I
DNI: 10100000-J
DNI: 11110000-K
DNI: 12120000-L
```

1. Estimate how long it will take you to implement each of these new functionalities, in hours or minutes.

2. Implement these two functionalities in a new version of your application and compare the time it took you with your original estimation.