CIT 594 Module 10 Programming Assignment

As we start learning about software design, this assignment focuses on communicating a design using UML class diagrams.

Learning Objectives

In completing this assignment, you will learn how to:

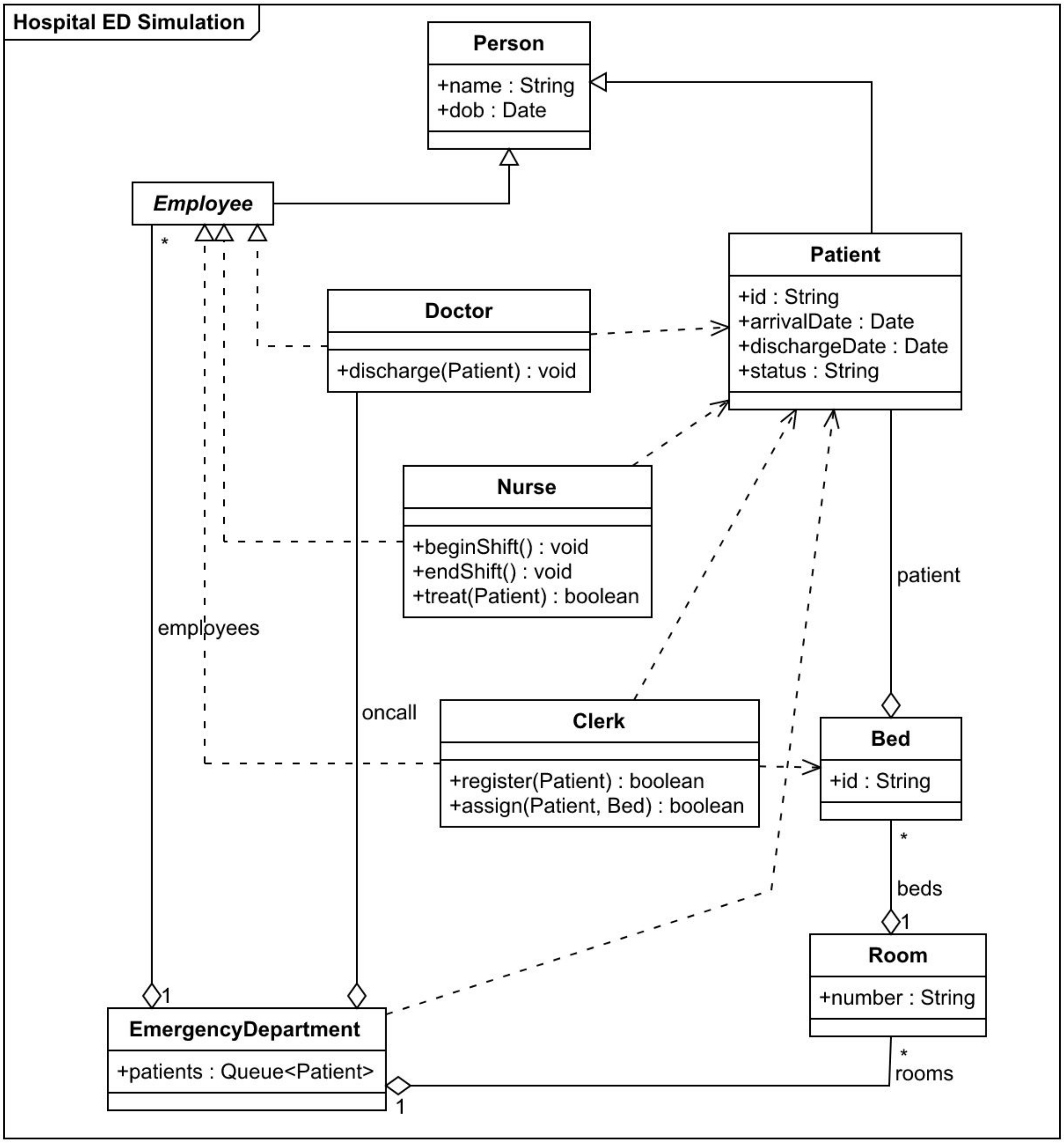
- convert a UML class diagram to Java code

Background

Hospitals and other medical facilities must make important decisions about equipment acquisition, resource allocation, staffing needs, etc. that affect expenditures but also patient care and satisfaction. Unfortunately, though, these medical facilities do not have the luxury of experimenting with these varying factors, as mistakes can prove financially costly and, in the worst case, can endanger patients' lives.

Therefore, medical facilities may use software to simulate the effects of these factors, e.g. by simulating the flow of patients through a hospital Emergency Department (ED) to get an idea of how long they spend in the ED depending on the availability of resources such as clerks, nurses, doctors, rooms, and beds.

The UML class diagram below represents a model of the classes that can be used in such a simulation:


Hospital ED Simulation
Person
+name : String
+dob : Date
Employee
*
Patient
+id : String
+arrivalDate : Date
+dischargeDate : Date
+status : String
Doctor
+discharge(Patient) : void
Nurse
+beginShift() : void
+endShift() : void
+treat(Patient) : boolean
patient
employees
oncall
Clerk
+register(Patient) : boolean
+assign(Patient, Bed) : boolean
Bed
+id : String
*
beds
1
Room
+number : String
1
EmergencyDepartment
+patients : Queue<Patient>
*
rooms
1

Activity

Implement the design in Java by creating classes/interfaces according to the class diagram and following these guidelines:

- You do not have to actually implement the methods, of course, but your code must compile, so have methods return any value as needed.
- You do not need to worry about any sort of main method; just represent the classes as shown above.
- All fields and methods should be **public** (that's what the little plus-sign next to the method names indicates). You do not need to worry about getters and setters.
- For the Date and Queue classes, use the implementations from the java.util package.
- Use arrays to implement multiplicity for aggregation/composition. For instance, if class Puppy has a "zero or more" relationship with class Toy, represent it as a Toy[ ] field in Puppy (as opposed to, say, a List<Toy> or Set<Toy>)

Before You Submit

Please be sure that all classes are in the default package, i.e. there is no "package" declaration at the top of the source code, and that all classes compile.

How to Submit

After you have finished implementing the classes, go to the "Module 10 Programming Assignment Submission" item and click the "Open Tool" button to go to the Codio platform.

Once you are logged into Codio, read the submission instructions in the README file. Be sure you upload your code to the "submit" folder.

To test your code before submitting, click the "Run Test Cases" button in the Codio toolbar.

These test cases will **not** give you detailed information about errors in your implementation; doing so would make the assignment too easy! Rather, the tests will only tell you which classes have problems.

However, if all tests are successful, then you would earn 100% on this assignment.

Assessment

This assignment is scored out of a total of 61 points.

Each class is graded based on the extent to which it adheres to the UML class diagram, including: whether or not it is abstract; what class is its superclass; its fields; and its methods.