1. (10 marks) Sketch the UML diagrams corresponding to the following situations:

a) (5 marks) Every doctor in a hospital has a unique practicing-physician ID number and the object associated with that doctor in the hospital database has a procedure `getPID` for accessing that ID number. There are two specific types of doctors, interns and clinicians. Each intern has a current associated rotation-department (`RotDept`) and an associated clinician-mentor. Each clinician mentors between three and ten interns.

```
Doctor
  PID
  getPID()

Intern
  RotDept

Clinician
  Mentor

3..10
```

b) (5 marks) Each regular ward in a hospital has one or more associated patient-rooms, and each such room has between two and four beds as well as a uniquely-addressed electronic dataport built into one of the room walls. An individual bed is always associated with a particular room, but beds can be moved into different rooms or even different wards; note, however, that no bed can be moved out of the regular wards.

```
WardReg
  Room

WardRegSet
  Bed

Room
  Dataport

1,..,*

2..4
```
2. (40 marks)

All parts of this question refer to the hospital-example handout.

a) (16 marks) Given classes `Doctor`, `Monitor`, `Chart`, `ChartEntry`, `Patient`, `PatientIn`, `PatientOut`, `PatientInReg`, and `PatientInICU` derived from the description in the handout, give a UML diagram showing all of the relationships between these classes implicit in this description.

b) (12 marks) Specify the syntax portion of a MIS for the ICU monitor module described in the handout by filling in the table on the following page. You may assume the existence of classes `Doctor`, `Patient`, `Lifesign`, and `LifesignSet` and exception-types `InvalidDoctor`, `InvalidLifesign` and `InvalidNumDoctors`; you may also assume that all lifesign-readings have floating-point values.
<table>
<thead>
<tr>
<th>Name</th>
<th>Inputs</th>
<th>Outputs</th>
<th>Exceptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitor / init</td>
<td>Patient, LifeSignSet</td>
<td>Monitor</td>
<td>–</td>
</tr>
<tr>
<td>addDoctor</td>
<td>Doctor</td>
<td>–</td>
<td>INVALIDNumDoctors</td>
</tr>
<tr>
<td>delDoctor</td>
<td>Doctor</td>
<td>–</td>
<td>INVALIDDoctor</td>
</tr>
<tr>
<td>addReading</td>
<td>Doctor, Lifesign</td>
<td>–</td>
<td>INVALIDDoctor, INVALIDLifesign</td>
</tr>
<tr>
<td>delReading</td>
<td>Doctor, Lifesign</td>
<td>–</td>
<td>INVALIDDoctor, INVALIDLifesign</td>
</tr>
<tr>
<td>updateReading</td>
<td>Lifesign, double</td>
<td>–</td>
<td>INVALIDLifesign</td>
</tr>
</tbody>
</table>
c) **(12 marks)** Give the **Java** code “skeleton” (class definition with all fields and the header-line of each method) for the **Chart** class relative to the MIS in the handout. All fields must be hidden within the class and all methods should be accessible within other classes. You may assume that function init corresponds to the constructor for this class; you may also assume the existence of classes **ChartEntry**, **Doctor**, **Patient**, and **Date**. Note that your code does not have to take exceptions into account.

**Answer:**

```java
class Chart {

    private ChartEntry[] CA;
    Doctor DP;
    Patient P;

    public Chart(Patient P, Doctor DP) { ... }

    public Doctor getPDoctor() { ... }

    public void changePDoctor(Doctor newDP) { ... }

    public void addEntry(ChartEntry C) { ... }

    public Doctor[] getDoctorsDate(Date df, Date dl) { ... }

    public ChartEntry[] getEntriesDate(Date df, Date dl) { ... }

}
```