By Aisling O’Reilly and Allen Simbul

OOAD Project - 2017

Table of Contents

[1. PART A 3](#_Toc481972359)

[Assumptions: 3](#_Toc481972360)

[Use Case Narratives: 3](#_Toc481972360)

[Question: h 10](#_Toc481972361)

[Question: i 10](#_Toc481972362)

[Question: j 11](#_Toc481972363)

[2. PART B 12](#_Toc481972364)

[Q1 – Part B 12](#_Toc481972365)

[**i. Draw a class diagram to model the code.** 13](#_Toc481972366)

[**ii. Explain why the code violates the open closed principle.** 13](#_Toc481972367)

[**iii. Rewrite the code so it does not violate this design principle.** 13](#_Toc481972368)

[**iv. Draw a class diagram to model revised code.** 14](#_Toc481972369)

[Q2 – Part B 15](#_Toc481972370)

[**Explanation of Design:** 15](#_Toc481972371)

[3. VP MODEL 16](#_Toc481972372)

[4. Sequence Diagram 18](#_Toc481972373)

[5. System Sequence Diagram 21](#_Toc481972374)

# 

# 1. PART A

## Assumptions:

* Bed has own class and attribute that identifies it
* Both the Clinical Coordinator Nurse and Nurse Manager don’t have specific methods within scope of Hospital System and can be represented as Role names
* An object instantiated from the Test Value class is a value object representing the healthy value for each Test Type
* While a Surgery class was necessary, the inclusion of a surgeon class fell beyond the scope of the project
* There is the concept of a patient stay association class, that contains references to:
  + Ward (which contains reference to Bed)
  + Surgery/(s)
  + Treatment/(s)
  + Test/(s)
  + Team

## Use Case Narratives

Use Case Narrative:

Use Case: **Admit Patient**

Actors: Hospital Administrator

Purpose: Record the admission of a patient to the system.

Overview:

* The Hospital Administrator enter the details of the patient to be admitted, these details would include the name, sex, pps number and if applicable Private Insurant Info.
* If patient has previously been admitted then there data is retrieved via PPS number.
* Based on their sex and availability of beds in a ward the patient info is recorded including the ward they were admitted too.

Pre Conditions: Ward info and Hospital Admin must be known to the system.

Post Conditions: The patient info including the ward he was admitted to was saved in the system.

Flow of Events:

Actor Action (Happy Scenario):

1. The use case begins when a new patient or a previous patient

must be admitted to a ward.

2. HA provides the patients name, sex, dob and PPS number and if applicable a private insurance info.

4. When there are no more patients to be admitted the HA indicates to the system that patient admission is complete.

System Response(Happy Scenario):

3. Depending on the sex of the patient a ward with bed vacancy is chosen and the patient details along with the ward the patient is admitted to is recorded.

Use Case: **Treat Patient**

Actors: Hospital Administrator

Purpose: Records the date/time of treatment was given to the patient.

Overview:

* The HA identifies the patient via PPS num, the doctor that administered the treatment and the start date/time of the appointment.
* The name of the medication/s must be recorded along with the dosage given.

Pre Conditions: The patient, doctor and HA must be known to the system. The treat must have been administered to the patient. The doctor recorded must be in the same team that cares for the patient.

Post Conditions: The medication given to the patient, the dosage of the medicine and the date/time of treatment is recorded by the system.

Flow Of Events:

Actor Action(Happy Scenario):

1. The use case begins when HA wants to record the administration of treatment to a patient.

2. HA enters the PPS number of patient to identify the patient the treatment was given too.

3. HA enter the Doctor name and the Team Code of the doctor.

4. Name of Medication and dosage given to patient is inputted as well.

6. When no more treatment is to be added to a patient, HA indicates to system that this use case is complete.

System Response(Happy Scenario):

5. The system records the date/time of treatment, name of medication and the dosage given to the appropriate patient. A confirmation of this is sent to the HA.

Use Case: **Discharge Patient**

Actors: Hospital Administrator

Purpose:

* Identify Patient
* Produce Invoice
* Record discharge date

Overview:

* The HA enters the patient’s name and identifier. The system finds/identifies the bed and ward assigned to the patient and calculates the cost from the associated price per night and duration of stay. The system produces an invoice and records the discharge date. The status of the bed and ward are updated.

Pre Conditions:

* Patient known by system,
* Patient admitted and assigned bed in ward

Post Conditions:

* Patient marked as discharged and date recorded
* Bed in ward available
* Invoice created

Flow Of Events:

Actor Action(Happy Scenario):

1. The use case begins when a HA desires to discharge a patient.

2. HA enters the patient name and PPS Number (and indicates the patient is to be discharged).

4. The HA receives the patient invoice.

System Response(Happy Scenario):

3. The system identifies the bed and ward assigned to patient, calculates the cost and produces an invoice. The system records the discharge date and updates the ward and bed status.

Alternative Flow of Events

* Invalid patient details are entered. Indicate error and return to step 2.
* No match found. Indicate error and return to step 2.
* Patient already discharged. Indicate error and return to step 2.

Use Case: **Transfer Patient**

Actors: Hospital Administrator

Purpose:

* Identify patient and current bed and ward assigned
* Find and assign appropriate bed in new ward
* Update status of bed and ward previously assigned to patient

Overview:

* HA enters the patient’s name and identifier. The system finds/identifies the patient and bed and ward assigned to the patient. The HA initiates a search for a new ward of the appropriate type. The system displays appropriate wards with a bed available, if any. The HA selects a bed in a new ward. The system removes the patient from their current bed and ward and assigns/adds the patient to a bed in the new ward.

Pre Conditions:

* Patient known by system
* Bed currently assigned to patient
* Bed available in new appropriate ward

Post Conditions:

* Bed in new ward assigned to patient and status updated
* Previously assigned bed and ward status updated

Flow of Events:

Actor Action(Happy Scenario):

1. The use case begins when an administrator

desires to transfer a patient.

2. HA enters the patient name and PPS Number.

4. The HA initiates a search for wards of the appropriate type.

6. The HA receives the list of available wards and selects a suitable bed.

System Response(Happy Scenario):

3. The system identifies the patient and the bed and ward assigned to patient.

5. The system identifies appropriate wards with at least one bed available and returns the list.

7. The system removes the patient from the current ward and assigns the patient a bed in a new ward, updating the respective ward and bed status.

Alternative Flow of Events

* Invalid patient details are entered. Indicate error and return to step 2.
* No free bed available in appropriate ward. Indicate restriction and return to step 4.

Use Case: **List Ward’s Patients**

Actors: Hospital Administrator

Purpose: List all patients admitted to a ward specified by HA

Overview:

* The HA identifies a ward where the system would output the all patients admitted to that ward.
* The patients name, age in years and an indication if Insurance Info is public or private.

Pre Conditions: The ward, patient and HA must be known by the system.

Post Conditions: The info of all patients admitted in the ward specified by the HA is displayed to HA.

Flow of Events:

Actor Action(Happy Scenario):

1. The use case begins when the HA wants to display all the patient info in particular ward chosen.

2. HA enter the ward name to display all patient admitted on the ward.

4. When finished displaying the ward patients the HA indicates to the system that use case is complete.

System Response(Happy Scenario):

3. The system displays all patients admitted in the ward. Patient info include patients name, age in years and an indication if Insurance Info is public or private.

Use Case: **List Team’s Patients**

Actors: Hospital Administrator

Purpose:

* Identify team

Overview:

* A HA enters the team’s identifier. The system finds/identifies the team and displays each patient assigned to the team by name and respective assigned ward.

Pre Conditions:

* Team registered on system
* Patient’s assigned to the team

Post Conditions:

* Patient’s assigned to the team

Flow of Events:

Actor Action(Happy Scenario):

1. The use case begins when a HA desires to list a team’s patients.

2. A HA enters the team code.

4. The HA receives a list of the team’s patients.

System Response(Happy Scenario):

3. The system identifies the team and produces a list of the patients assigned to the team and their respective wards.

Alternative Flow of Events

* Invalid team identifier entered. Indicate error and return to step 2.
* No patients assigned to team. Indicate status of team and return to step 2.

Use Case: **List Patient’s Treatments**

Actors: Hospital Administrator

Purpose:

* Identify Patient
* List team assigned to patient and team members who treated patient
* List patient’s treatments and drugs prescribed

Overview:

A HA enters the patient’s name and identifier. The system finds/identifies the patient and displays:

* The team and consultant assigned to the patient
* The name and grade of team members that treated the patient
* Any treatments undertaken by the patient and associated prescribed drugs.

Pre Conditions:

* Patient registered on system
* Patient assigned team
* Treatment received by patient

Post Conditions:

* List of patient’s team and treatments created

Flow of Events:

Actor Action(Happy Scenario):

1. The use case begins when a HA desires to list a patient’s team and treatments.

2. A HA enters the patient’s name and PPS Number.

4. The HA receives a list of the patient’s team and treatments.

System Response(Happy Scenario):

3. The system identifies the patient and produces a list of the patient’s team, team members who administered treatment, treatments and prescribed drugs.

Alternative Flow of Events

* Invalid patient identifier entered. Indicate error and return to step 2.
* No team assigned to patient. Indicate status of patient and return to step 2.
* Patient hasn’t received any treatment. Indicate status of patient and return to step 2.

Use Case: **List Ward Nurses**

Actor: Nurse Manager

Purpose: Identify the groups of nurses working in the ward

Overview:

* NM identifies the name of the ward, which displays a list of nurses working in the same ward.
* Display nurse’s name, shift date, shift start time, shift end time.

Pre Conditions: Nurse Manager, Nurse, Ward is known by the system.

Post Conditions: List of nurses with all there info is displayed.

Flow of Events:

Actor Action(Happy Scenario):

1. The use case begins when the NM wants to display all the nurses info in a ward.

2. NM enter the ward name to display all nurse working on the ward.

4. When finished displaying the ward nurses the NM indicates to the system that use case is complete.

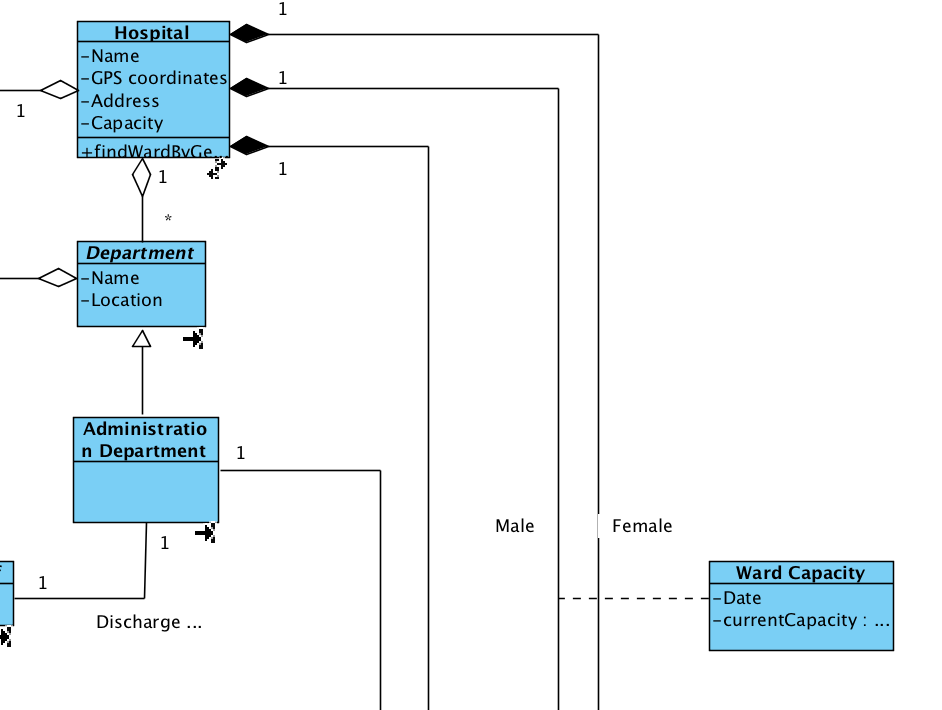
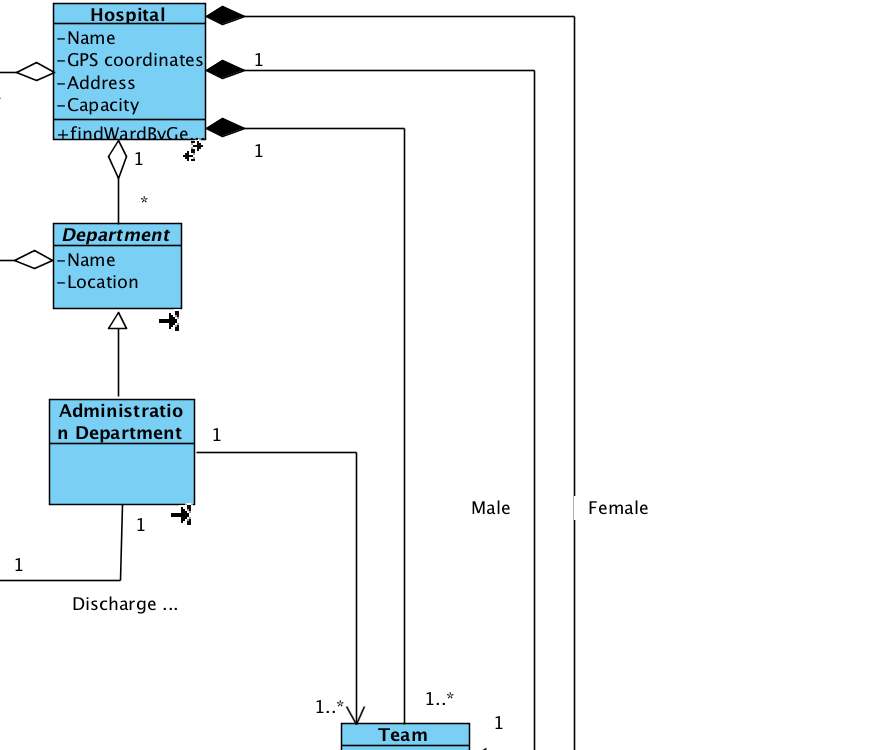
System Response(Happy Scenario):

3. The system displays all nurses admitted in the ward. Nurse info include name, staff number and the start time and end time of the shift.

## Question: h

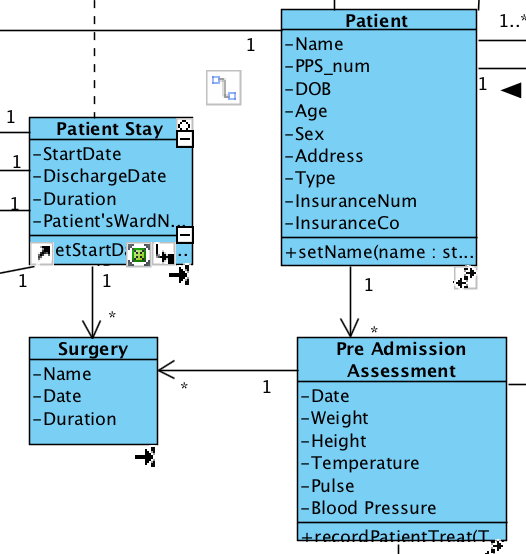
i. A manager would like to determine the number of patients staying in the hospital on a particular date.

This changed was added to the class diagram.



*before after*

ii. A doctor wants to view blood test results taken for a patient during a particular stay in the hospital.



## Question: i

i. Give an overview how the object model would be mapped to a relational model.

The object-oriented model utilises the concepts of:

- Classes of objects

- Attributes (which can be used as identifiers)

- References to objects to implement relationships between classes of objects

The equivalent concepts in a relational model are:

- Tables with records

- Columns (one of which will be used as a Primary Key)

- Foreign Keys

- Classes in the object-oriented model map to Tables in the relational model

- A data attribute of a class will map to zero or more columns in a relational database. Not all attributes are persistent, some are used for temporary calculations.

- A row of a table in the relational model corresponds to the attributes of an object in the object-oriented model

- Rows in tables are uniquely identified by primary keys and relationships between rows are maintained using foreign keys.

- From a relational database perspective, the only difference between association and aggregation/composition relationships is how tightly the objects are bound to each other

- To implement one-to-one and one-to-many relationships, include the key of one table in the other table.

- To implement many-to-many relationships, the concept of an associative table, a data entity whose sole purpose is to maintain the association between two or more tables in a relational database is necessary. The attributes contained in an associative table are traditionally the combination of the keys in the tables involved in the relationship.

## Question: j

j. Give an overview on how you would map design class diagram to code.

Source code must be written for Class Design Diagram and class definitions must also be created for them.

Department, Staff and Doctor class would be Interfaces. This done is done so that it would follow the OCP design principle. Anytime a new type of staff must be added to the system it can just inherit from the Staff interface so that the majority of the source code remains unchanged and it doesn’t affect the rest of the working functionality.

Since the Department, Staff and Doctor classes are interfaces, classes associated with them must inherit and implement their operations. Classes should only inherit from these interfaces if they have an “IS A” relationship with them. So classes in the Design Class that have a Generalization Association to the interfaces would be the one to implement the interfaces operations.

To keep with the encapsulation principle of OOP the majority if not all the operations listed in the Design Class Diagram would be set to private. This would also apply to the attributes in the classes.

The code for the operations of the classes must also be defined. The code for the operation must be written as such so that it promotes both low coupling and high cohesion.

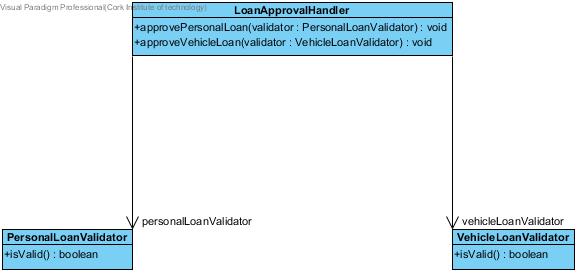
The multiplicity of the associations must be also be considered when mapping the design class diagram to code. Classes that have \* at the other end of the association would suggest that these classes have many references of the other class. This would suggest that they would have an arraylist of references of the other class. For example a Patient Class can have multiple Wards. Translated into code The Patient Class would have an ArrayList of Wards references as an attribute. The same goes for Ward, a Ward can have multiple Patient so it too would have an ArrayList of Patient references as an attribute.

# 

# 2. PART B

## Q1 – Part B

### **i. Draw a class diagram to model the code.**



### **ii. Explain why the code violates the open closed principle.**

The OCP states that classes should be open to extension without modifying the source code.

This means changing what the classes do without editing the source code.

The above UML diagram violates the OCP because each time a new a type of Validator is created the LoanApprovalHandler source code must be modified to access the isValid methods of the LoanValidator classes.

Solution:

A abstract class called Validate must be created and the VehicleLoanValidator and PersonalLoanValidator must extend this abstract class.

Now the LoanApproveHandler does not have to be modified and only have one method called approveLoan(Validate validate) to access the isValid() method of the two existing classes and any other class if more LoanValidator is added.

### **iii. Rewrite the code so it does not violate this design principle.**

Public class LoanApprovalHandler(){

approveLoan(Validate validate){

validate.isValid()

}

}

Public abstract class Validate(){

Public abstract Boolean isValid()

}

Public class PersonalLoanValidator extends Validate{

Public Boolean isValid(){

//do function

}

}

Public class VehicleLoanValidator extends Validate{

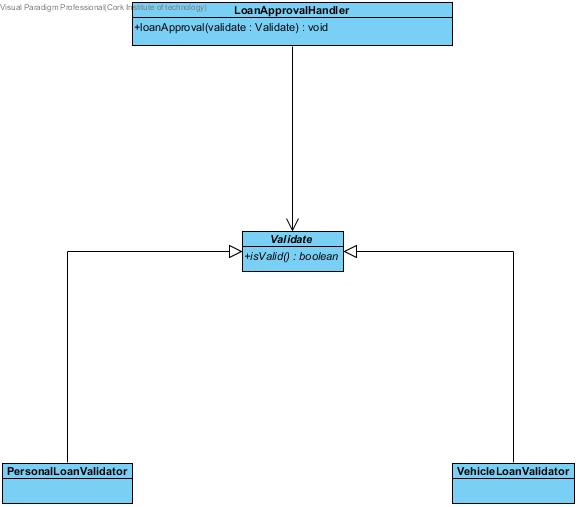
Public Boolean isValid(){

//do function

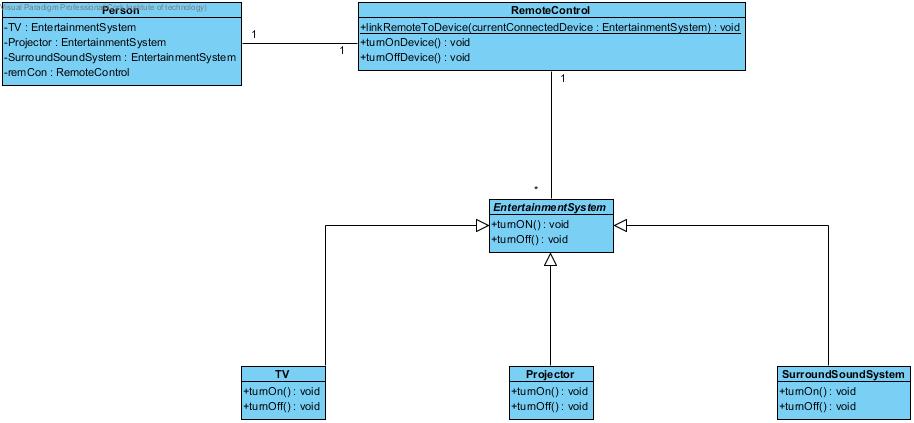
}

}

### **iv. Draw a class diagram to model revised code.**



## Q2 – Part B



### **Explanation of Design:**

1. Relationships:

- RemoteControl has multiple EntertainmentSystem classes. This is because more devices can be added to the Remote Control settings, it can manage more devices in the future.

- It can also be said that a remote control may not have any devices associated with it, if there are none at the moment.

- The Entertainment System would only know one remoteControl. The reason being is that according to the spec, devices would only work with one “universal” remote control at a time.

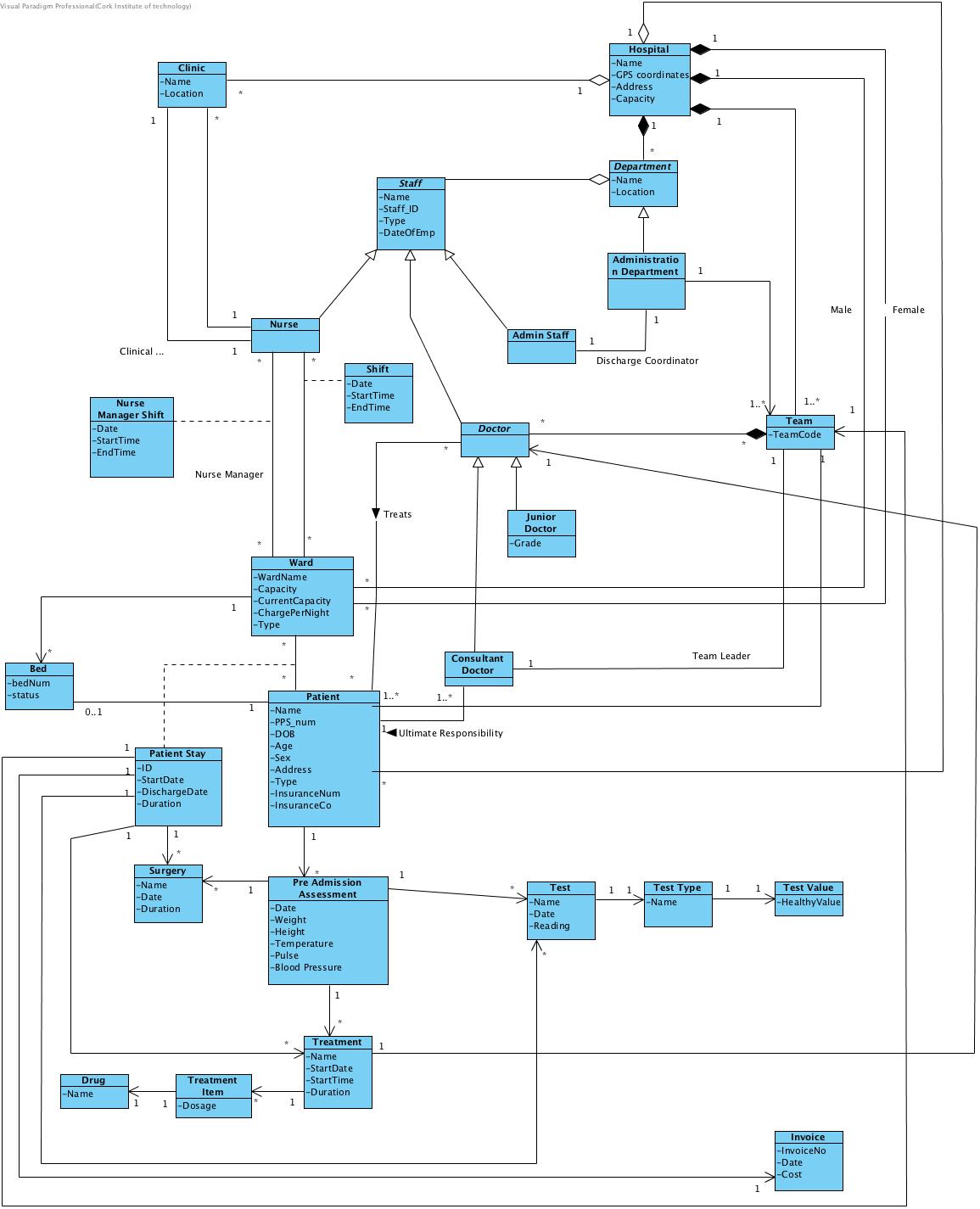
2. OCP Design

- As stated in the spec a new device must be added to the system without breaking the existing code. This would suggest that new classes must be added to the Home Entertainment system.

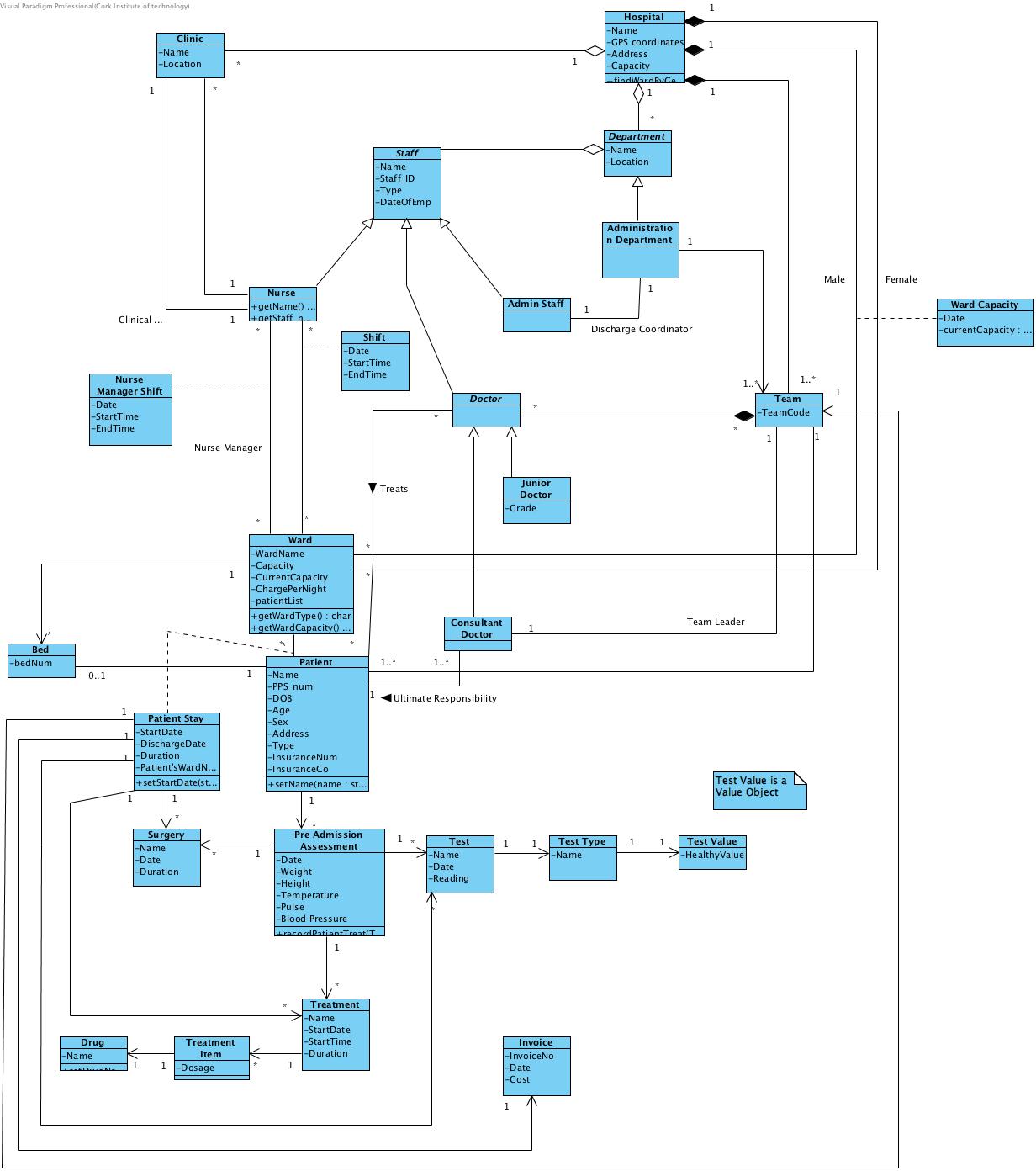
- This is the reason why EntertainmentSystem was made as an Abstract Class. So that new classes can be added to the system without having to modify the RemoteControl class.

- For example, if a new class was to be added like Console the OCP design can be achieved by making the console class depend on the EntertainmentSystem Abstract Class. Doing it in this way means that the source code wouldn’t need to be modified to accommodate this change.

# 3. VP MODEL

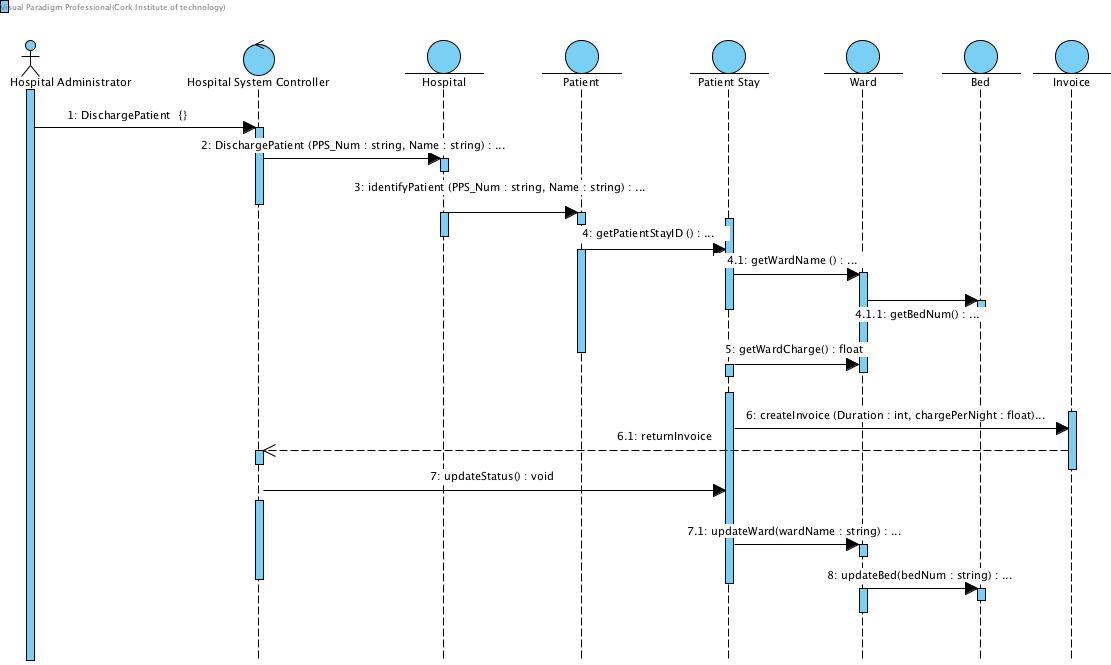


*Domain Class Diagram*

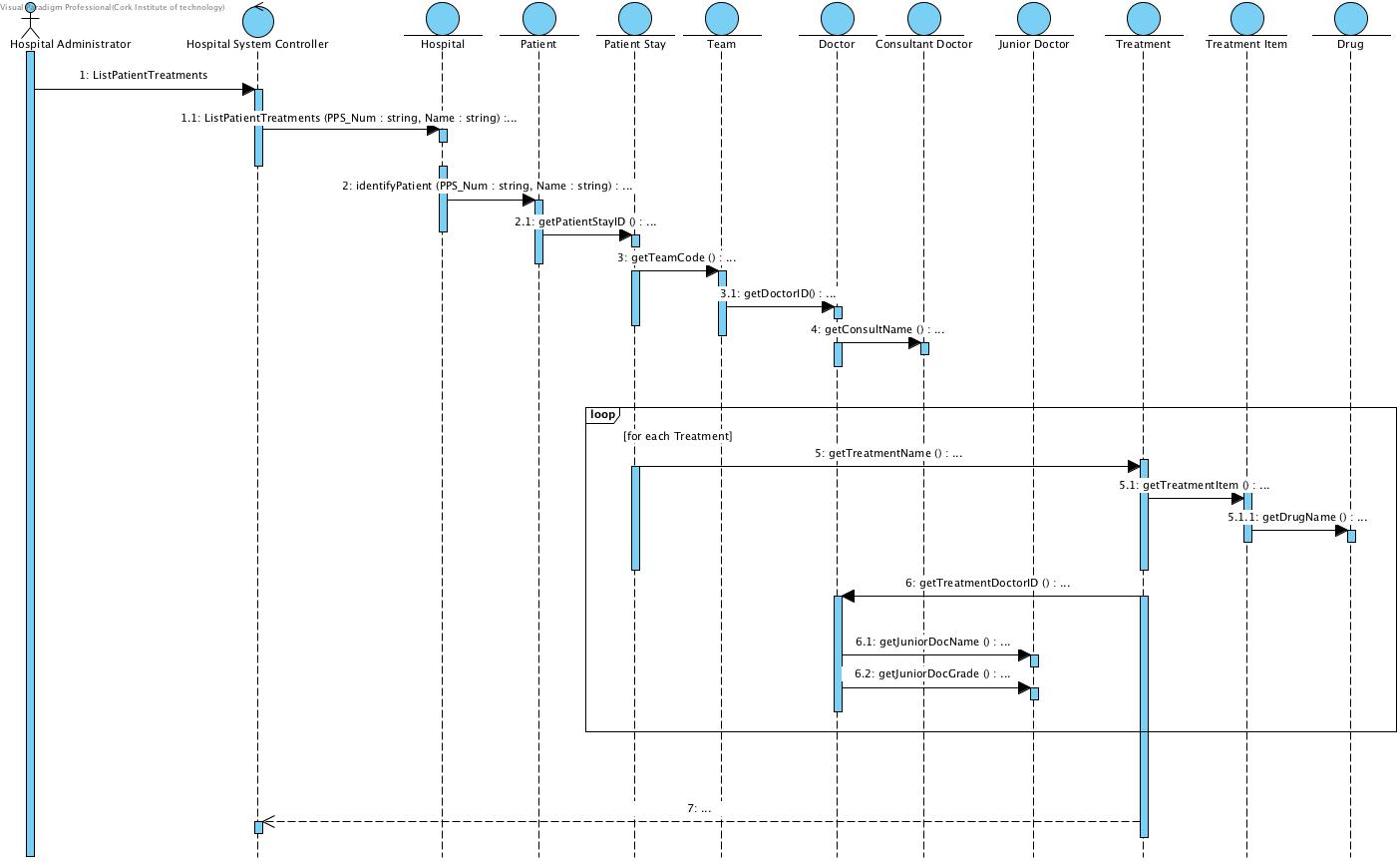


*Design Class Diagram*

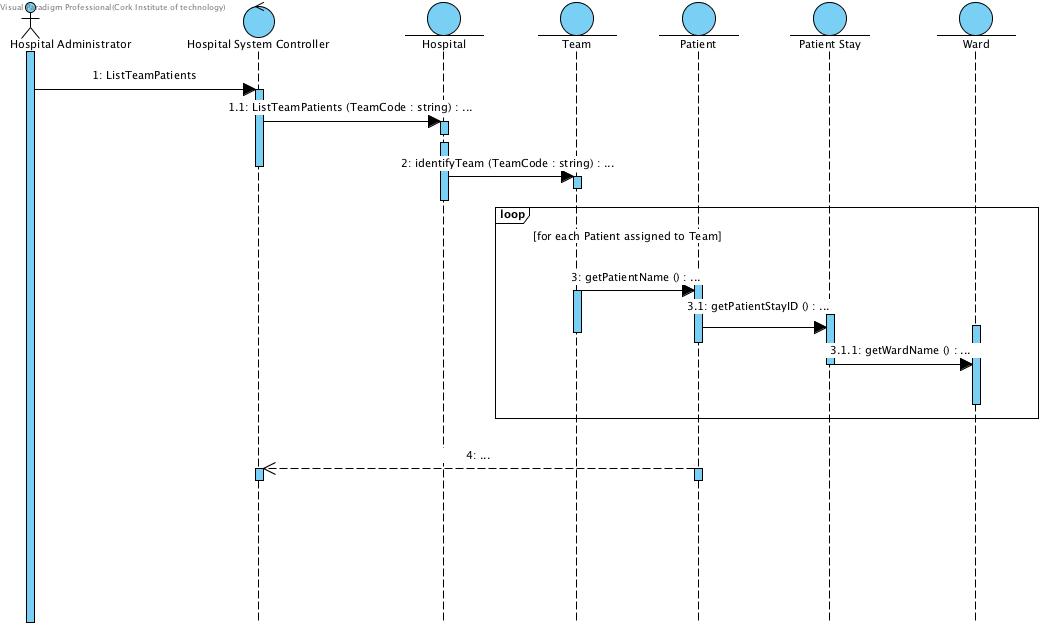
# 4. Sequence Diagram



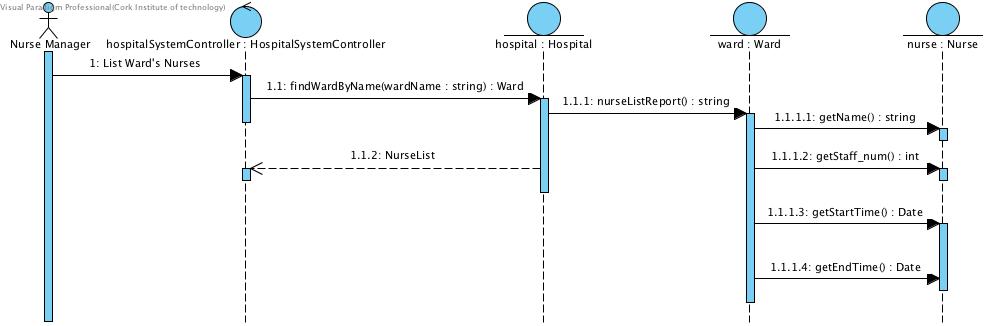
*Discharge Patient*

**

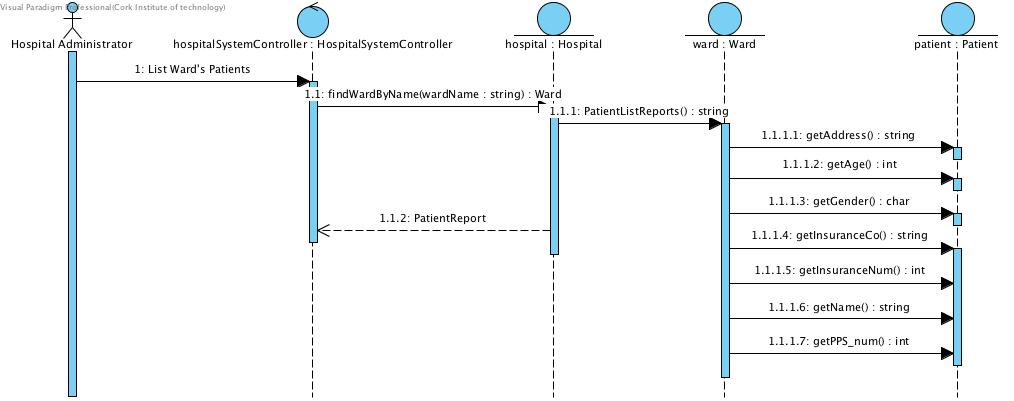
*List Patient’s Treatment*

**

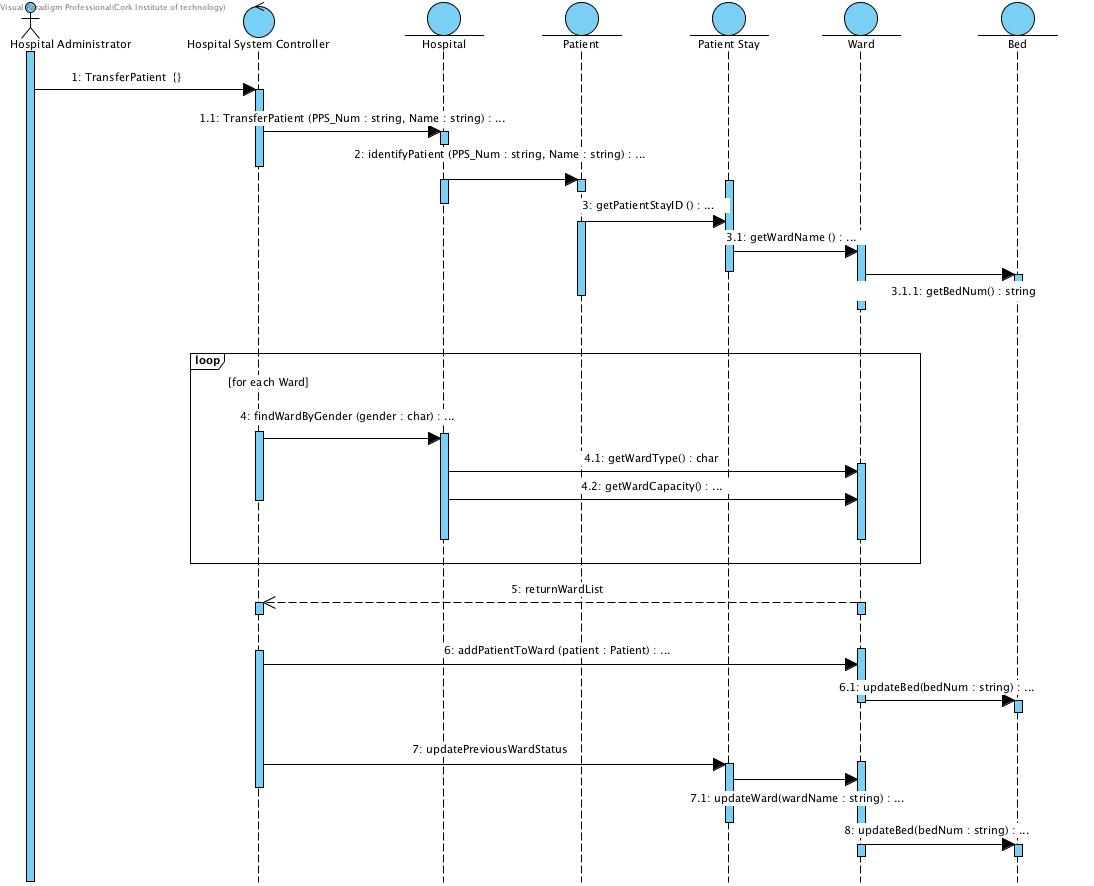
*List Team’s Patient*

**

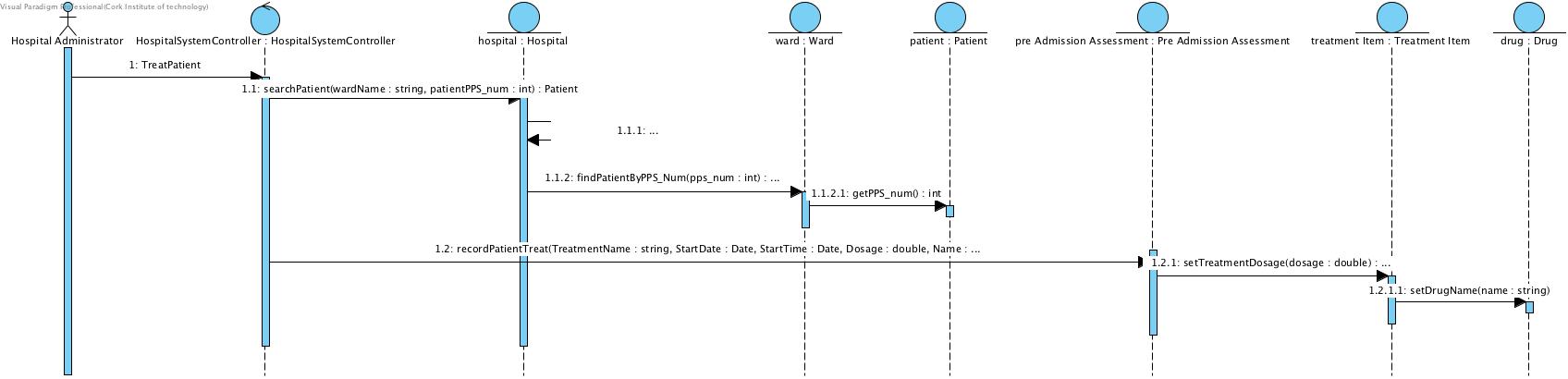
*List Wards Nurses*

**

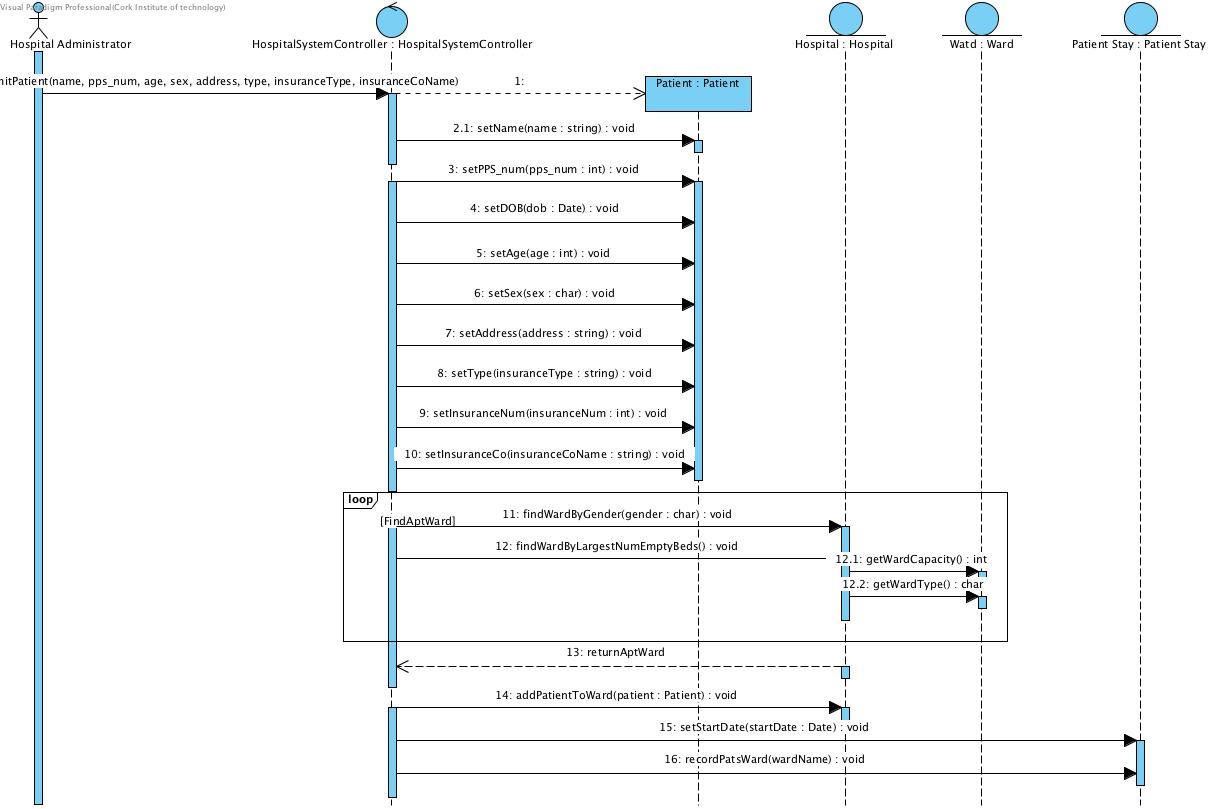
*List Wards Patient*

**

*Transfer Patient*

**

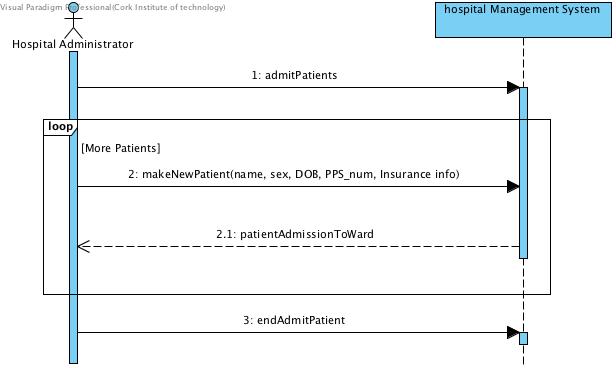
*Treat Patient*

**

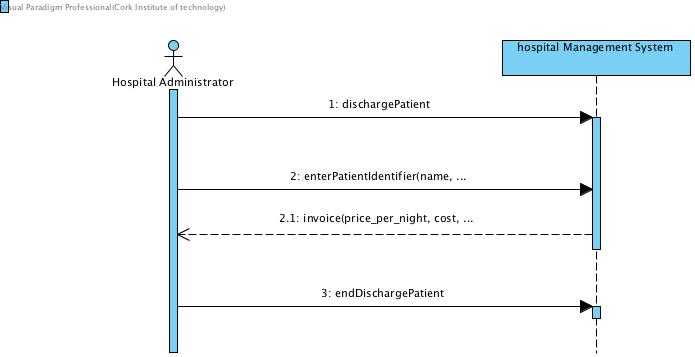
*Admit Patient*

# 

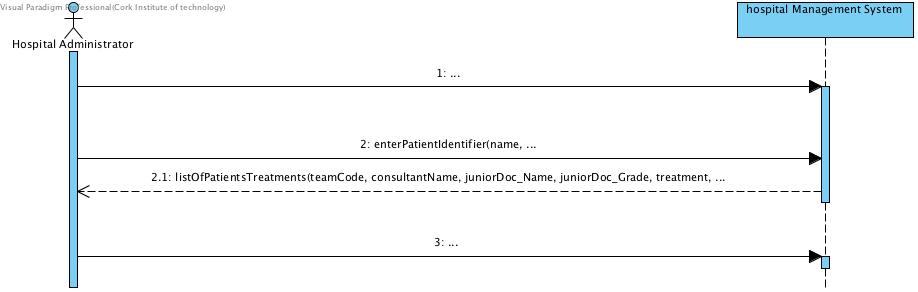
# 5. System Sequence Diagram



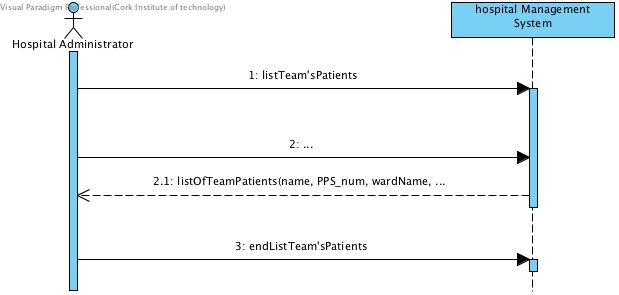
*Admit Patient*

**

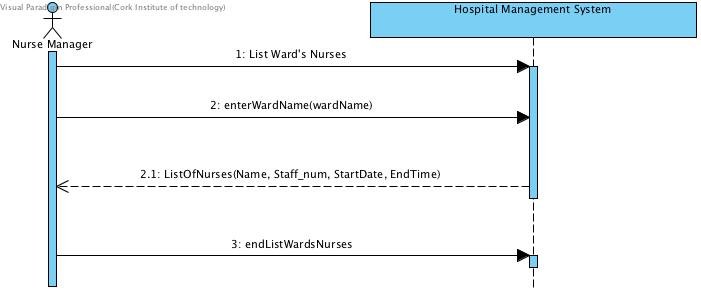
*Discharge Patient*

**

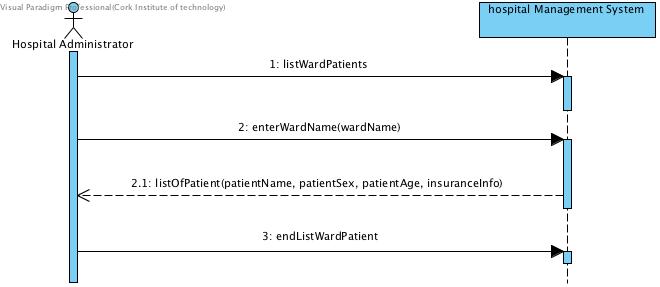
*List Patients Treatment*

**

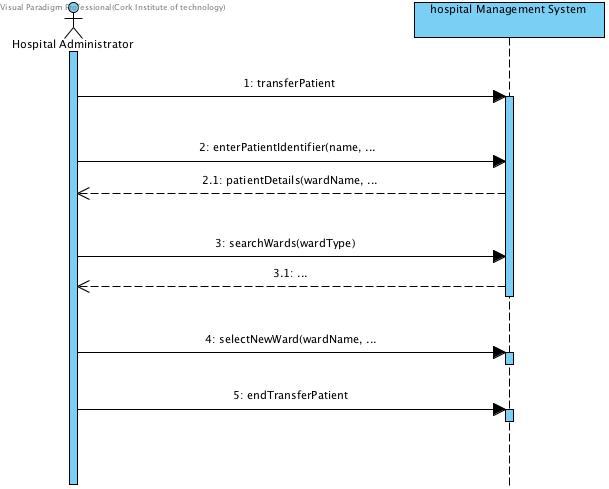
*List Team Patients*

**

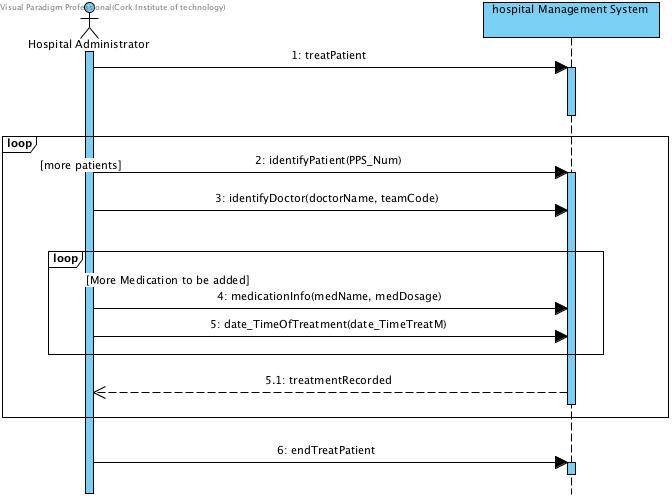
*List Ward Nurses*

**

*List Ward Patient*

**

*Transfer Patient*

**

*Treat Patient*