TOMFramework Project

Class Design Document

Version: 0.1 (Draft version)

*The purpose of this document is the draft version of class design that provides you with a guideline for detailing class design and writing the class diagram description for your solution based on design Pattern approach.*

*Points to remember:*

* *Content is important, not the volume.* ***Another team should be able to design in more detail from this document.***
* *Pay attention to overall in description.*
* *Completeness and consistency will be rewarded.*

Contents

[1. Name: ManageMedicalExaminationProcess 4](#_Toc57042056)

[2. Problem 4](#_Toc57042057)

[3. Solution 4](#_Toc57042058)

[3.1. Class diagram 4](#_Toc57042059)

[3.2. Class Descriptions 4](#_Toc57042060)

[4. Consequence 10](#_Toc57042061)

[5. Implementation (code examples for Receive, HTTP and MQTT only) 10](#_Toc57042062)

## Name: ManageMedicalExaminationProcess

## Problem

* The medical examination process goes as follows:

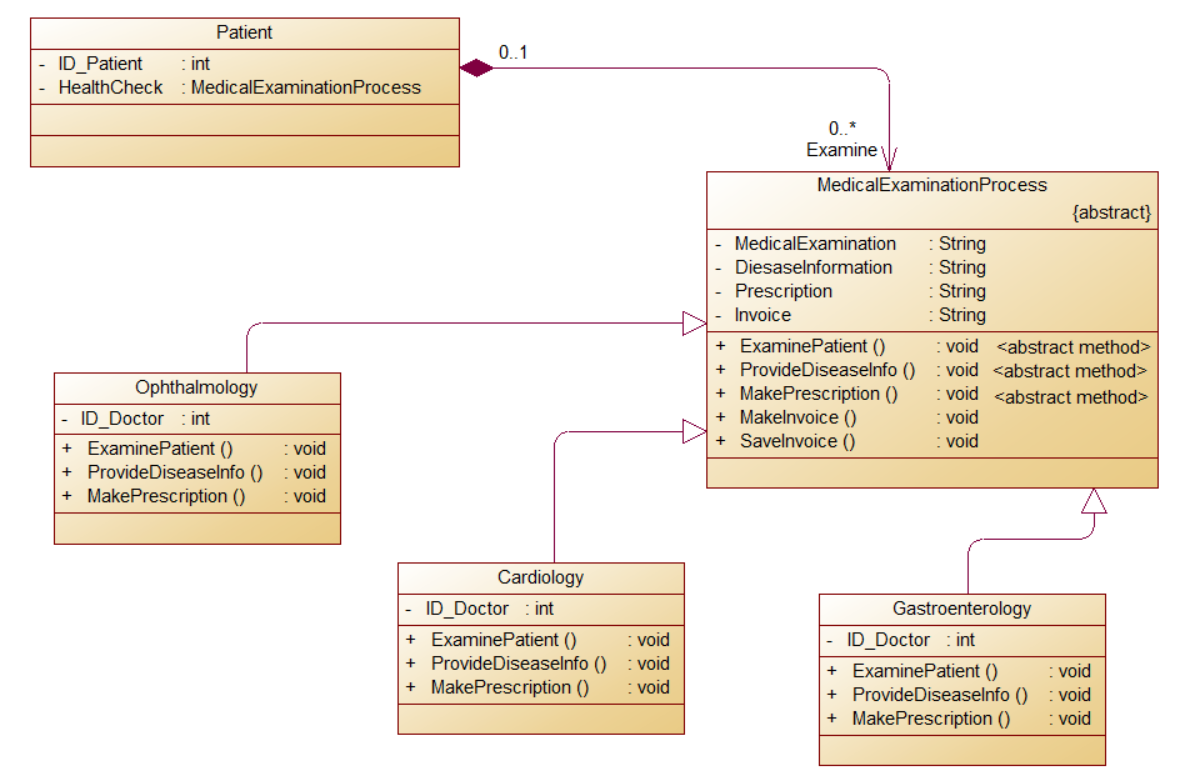
1. Examine for patient
2. Provide disease information
3. Make a prescription
4. Make a invoice
5. Save invoice

* Each patient will be seen by a specialist.
* The examination process in each department is the same, but activities such as examining, providing disease information and making a prescription are different.

## Solution

* Use Template method pattern to design the above module.

### Class diagram



### Class Descriptions

*In this section you would describe in detail each class, its attributes, and its methods. You should logically group classes together. For example, you may use your architecture diagrams to group classes within a sub-system together.*

*Provide a subsection for each class. For each class, briefly describe its purpose, any constraints, (e.g., only single instance) and list the attributes and the methods of each class in the class diagram.*

*For each class, describe each of its attributes with the following details: name, type, a one line description of the attribute if its meaning is not intuitive, and constraints on the attribute (e.g., attribute must have unique value for each object or value range is restricted to positive integers).*

*Each method should be described with the following details: method name, return type and value, parameters, purpose and a brief description of the algorithm used (if it is non-trivial). Pre-conditions and post-conditions should be mentioned here if there are any assumptions about the arguments or the return values. List the attributes read and modified by this method and other methods invoked by this method.*

***The following is provided as an example:***

***Pay attention: only describe Instrument and its subclass (Handy and Station)***

#### **Class: Patient**

* Purpose: To model the relevant aspects of the Patient
* Constraints: *None*
* Persistent: *No*

Attribute Descriptions

1. Attribute:ID\_Patient

Visibility: private

Type: int

Description: ID of patients

Constraints:non-negative

1. Attribute:HealthCheck

Visibility: private

Type: MedicalExaminationProcess

Description: Manage medical examination process for each patient by each department

Constraints:non-negative

Method Descriptions

1. Method: Patient (Creater method)

Visibility: public

Return Type: Object Patient

Parameters: ID and Medical examination process of patients

1. Method: getID

Visibility: public

Return Type: int

Parameters: Return ID of patients

1. Method: getME

Visibility: public

Return Type: MedicalExaminationProcess

Parameters: Return Medical examination process of patients

#### **Abstract Class: MedicalExaminationProcess**

* Purpose: Implement the invariant parts of the Medical Examination Process in an abstract class
* Constraints: *None*
* Persistent: *No*

Attribute Descriptions

1. Attribute:MedicalExamination

Visibility: private

Type: String

Description: Steps in ophthalmic examination

Constraints:non-negative

1. Attribute:DiesaseInformation

Visibility: private

Type: String

Description: Patient’s diesase information

Constraints:non-negative

1. Attribute:Prescription

Visibility: private

Type: String

Description: Prescription of patient

Constraints:non-negative

1. Attribute:Invoice

Visibility: private

Type: String

Description: Invoice of patient

Constraints:non-negative

Method Descriptions

1. Method: ExaminePatient <abstract>

Visibility: public

Return Type: void

1. Method: ProvideDiseaseInfo <abstract>

Visibility: public

Return Type: void

1. Method: MakePrescription <abstract>

Visibility: public

Return Type: void

1. Method: MakeInvoice

Visibility: public

Return Type: void

Parameters:

Pre-condition:

Post-condition:

Attributes read/used:

Methods called:

Processing logic:

Test case:

1. Method: SaveInvoice

Visibility: public

Return Type: void

Parameters:

Pre-condition:

Post-condition:

Attributes read/used:

Methods called:

Processing logic:

Test case:

#### **Class: Ophthalmology**

* Purpose: to model the steps to perform Medical Examinations in ophthalmology
* Constraints: *None*
* Persistent: *No*

Attribute Descriptions

1. Attribute:ID\_Doctor

Visibility: private

Type: int

Description: ID of doctor in ophthalmology

Constraints:non-negative

Method Descriptions

*Create, getters and setters methods*

1. Method: ExaminePatient

Visibility: public

Return Type: void

Parameters:

Pre-condition:

Post-condition:

Attributes read/used:

Methods called:

Processing logic:

Test case:

1. Method: ProvideDiseaseInfo

Visibility: public

Return Type: void

Parameters:

Pre-condition:

Post-condition:

Attributes read/used:

Methods called:

Processing logic:

Test case:

1. Method: MakePrescription

Visibility: public

Return Type: void

Parameters:

Pre-condition:

Post-condition:

Attributes read/used:

Methods called:

Processing logic:

Test case:

#### **Class: Cardiology**

* Purpose: to model the steps to perform Medical Examinations in cardiology
* Constraints: *None*
* Persistent: *No*

Attribute Descriptions

1. Attribute:ID\_Doctor

Visibility: private

Type: int

Description: ID of doctor in cardiology

Constraints:non-negative

Method Descriptions

*Create, getters and setters methods*

1. Method: ExaminePatient

Visibility: public

Return Type: void

Parameters:

Pre-condition:

Post-condition:

Attributes read/used:

Methods called:

Processing logic:

Test case:

1. Method: ProvideDiseaseInfo

Visibility: public

Return Type: void

Parameters:

Pre-condition:

Post-condition:

Attributes read/used:

Methods called:

Processing logic:

Test case:

1. Method: MakePrescription

Visibility: public

Return Type: void

Parameters:

Pre-condition:

Post-condition:

Attributes read/used:

Methods called:

Processing logic:

Test case:

#### **Class: Gastroenterology**

* Purpose: to model the steps to perform Medical Examinations in gastroenterology
* Constraints: *None*
* Persistent: *No*

Attribute Descriptions

1. Attribute:ID\_Doctor

Visibility: private

Type: int

Description: ID of doctor in gastroenterology

Constraints:non-negative

Method Descriptions

*Create, getters and setters methods*

1. Method: ExaminePatient

Visibility: public

Return Type: void

Parameters:

Pre-condition:

Post-condition:

Attributes read/used:

Methods called:

Processing logic:

Test case:

1. Method: ProvideDiseaseInfo

Visibility: public

Return Type: void

Parameters:

Pre-condition:

Post-condition:

Attributes read/used:

Methods called:

Processing logic:

Test case:

1. Method: MakePrescription

Visibility: public

Return Type: void

Parameters:

Pre-condition:

Post-condition:

Attributes read/used:

Methods called:

Processing logic:

Test case:

## Consequence

* Easily manage the patient's health examination if the patient comes to see many times
* Code reuse for the invariant parts of medical examination process
* Customization is restricted

## Implementation (code examples for Receive, HTTP and MQTT only)