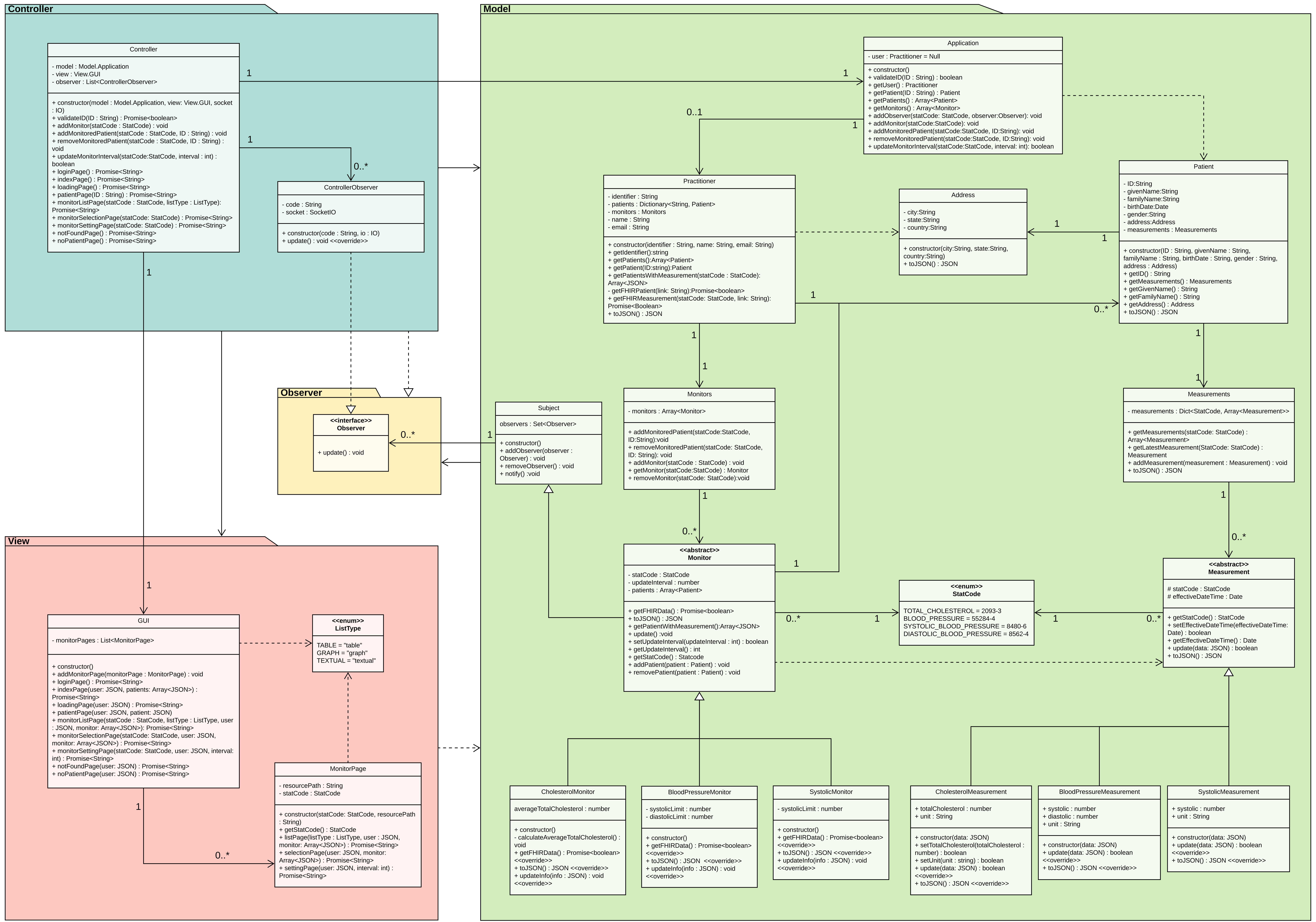
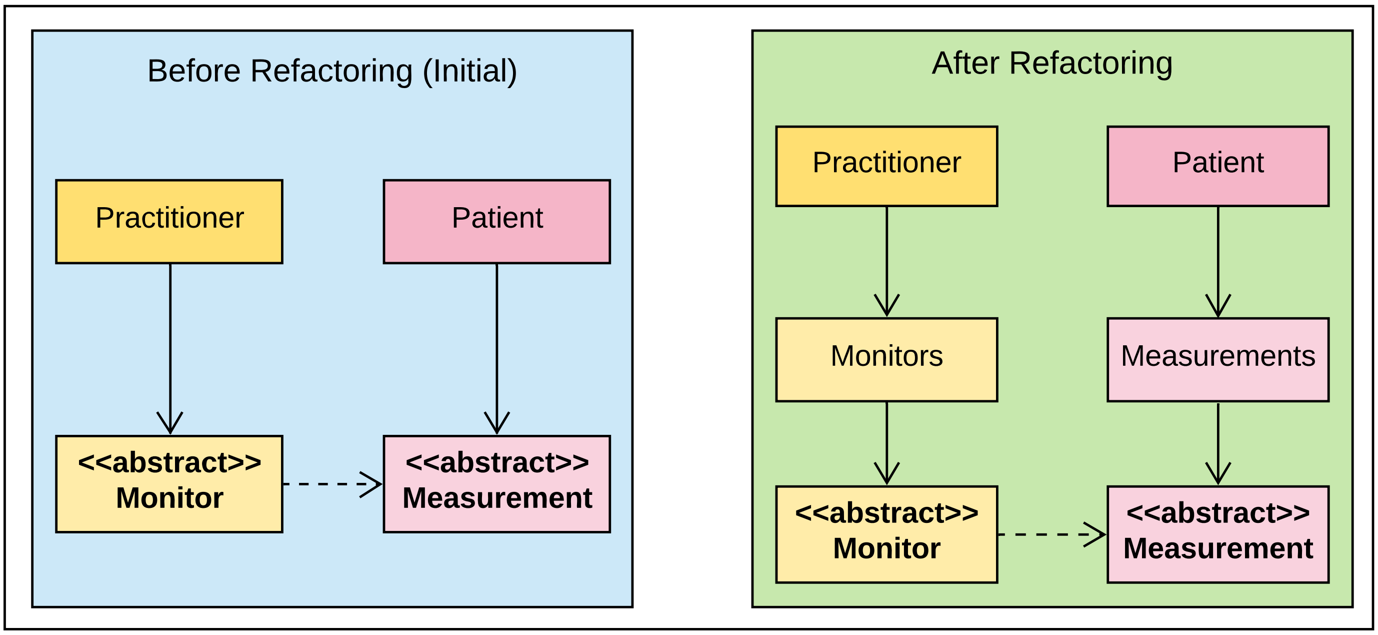

Controller
Controller
- model : Model.Application
- view : View.GUI
- observer : List<ControllerObserver>
+ constructor(model : Model.Application, view: View.GUI, socket : IO)
+ validateID(ID : String) : Promise<boolean>
+ addMonitor(statCode : StatCode) : void
+ addMonitoredPatient(statCode : StatCode, ID : String) : void
+ removeMonitoredPatient(statCode : StatCode, ID : String) : void
+ updateMonitorInterval(statCode:StatCode, interval : int) : boolean
+ loginPage() : Promise<String>
+ indexPage() : Promise<String>
+ loadingPage() : Promise<String>
+ patientPage(ID : String) : Promise<String>
+ monitorListPage(statCode : StatCode, listType : ListType): Promise<String>
+ monitorSelectionPage(statCode: StatCode) : Promise<String>
+ monitorSettingPage(statCode: StatCode) : Promise<String>
+ notFoundPage() : Promise<String>
+ noPatientPage() : Promise<String>
ControllerObserver
- code : String
- socket : SocketIO
+ constructor(code : String, io : IO)
+ update() : void <<override>>
Observer
<<interface>>
Observer
+ update() : void
Model
Application
- user : Practitioner = Null
+ constructor()
+ validateID(ID : String) : boolean
+ getUser() : Practitioner
+ getPatient(ID : String) : Patient
+ getPatients() : Array<Patient>
+ getMonitors() : Array<Monitor>
+ addObserver(statCode: StatCode, observer:Observer): void
+ addMonitor(statCode:StatCode): void
+ addMonitoredPatient(statCode:StatCode, ID:String): void
+ removeMonitoredPatient(statCode:StatCode, ID:String): void
+ updateMonitorInterval(statCode:StatCode, interval: int): boolean
Practitioner
- identifier : String
- patients : Dictionary<String, Patient>
- monitors : Monitors
- name : String
- email : String
+ constructor(identifier : String, name: String, email: String)
+ getIdentifier():string
+ getPatients():Array<Patient>
+ getPatient(ID:string):Patient
+ getPatientsWithMeasurement(statCode : StatCode): Array<JSON>
- getFHIRPatient(link: String):Promise<boolean>
+ getFHIRMeasurement(statCode: StatCode, link: String): Promise<Boolean>
+ toJSON() : JSON
Address
- city:String
- state:String
- country:String
+ constructor(city:String, state:String, country:String)
+ toJSON() : JSON
Patient
- ID:String
- givenName:String
- familyName:String
- birthDate:Date
- gender:String
- address:Address
- measurements : Measurements
+ constructor(ID : String, givenName : String, familyName : String, birthDate : String, gender : String, address : Address)
+ getID() : String
+ getMeasurements() : Measurements
+ getGivenName() : String
+ getFamilyName() : String
+ getAddress() : Address
+ toJSON() : JSON
Subject
observers : Set<Observer>
+ constructor()
+ addObserver(observer : Observer) : void
+ removeObserver() : void
+ notify() :void
Monitors
- monitors : Array<Monitor>
+ addMonitoredPatient(statCode:StatCode, ID:String):void
+ removeMonitoredPatient(statCode: StatCode, ID: String): void
+ addMonitor(statCode : StatCode) : void
+ getMonitor(statCode:StatCode) : Monitor
+ removeMonitor(statCode: StatCode):void
Measurements
- measurements : Dict<StatCode, Array<Measurement>>
+ getMeasurements(statCode: StatCode) : Array<Measurement>
+ getLatestMeasurement(StatCode: StatCode) : Measurement
+ addMeasurement(measurement : Measurement) : void
+ toJSON() : JSON
<<abstract>>
Monitor
- statCode : StatCode
- updateInterval : number
- patients : Array<Patient>
+ getFHIRData() : Promise<boolean>
+ toJSON() : JSON
+ getPatientWithMeasurement():Array<JSON>
+ update() :void
+ setUpdateInterval(updateInterval : int) : boolean
+ getUpdateInterval() : int
+ getStatCode() : Statcode
+ addPatient(patient : Patient) : void
+ removePatient(patient : Patient) : void
<<enum>>
StatCode
TOTAL_CHOLESTEROL = 2093-3
BLOOD_PRESSURE = 55284-4
SYSTOLIC_BLOOD_PRESSURE = 8480-6
DIASTOLIC_BLOOD_PRESSURE = 8562-4
<<abstract>>
Measurement
# statCode : StatCode
# effectiveDateTime : Date
+ getStatCode() : StatCode
+ setEffectiveDateTime(effectiveDateTime: Date) : boolean
+ getEffectiveDateTime() : Date
+ update(data: JSON) : boolean
+ toJSON() : JSON
CholesterolMonitor
averageTotalCholesterol : number
+ constructor()
- calculateAverageTotalCholesterol() : void
+ getFHIRData() : Promise<boolean> <<override>>
+ toJSON() : JSON <<override>>
+ updateInfo(info : JSON) : void <<override>>
BloodPressureMonitor
- systolicLimit : number
- diastolicLimit : number
+ constructor()
+ getFHIRData() : Promise<boolean> <<override>>
+ toJSON() : JSON <<override>>
+ updateInfo(info : JSON) : void <<override>>
SystolicMonitor
- systolicLimit : number
+ constructor()
+ getFHIRData() : Promise<boolean> <<override>>
+ toJSON() : JSON <<override>>
+ updateInfo(info : JSON) : void <<override>>
CholesterolMeasurement
+ totalCholesterol : number
+ unit : String
+ constructor(data: JSON)
+ setTotalCholesterol(totalCholesterol : number) : boolean
+ setUnit(unit : string) : boolean
+ update(data: JSON) : boolean <<override>>
+ toJSON() : JSON <<override>>
BloodPressureMeasurement
+ systolic : number
+ diastolic : number
+ unit : String
+ constructor(data: JSON)
+ update(data: JSON) : boolean <<override>>
+ toJSON() : JSON <<override>>
SystolicMeasurement
+ systolic : number
+ unit : String
+ constructor(data: JSON)
+ update(data: JSON) : boolean <<override>>
+ toJSON() : JSON <<override>>
View
GUI
- monitorPages : List<MonitorPage>
+ constructor()
+ addMonitorPage(monitorPage : MonitorPage) : void
+ loginPage() : Promise<String>
+ indexPage(user: JSON, patients: Array<JSON>) : Promise<String>
+ loadingPage(user: JSON) : Promise<String>
+ patientPage(user: JSON, patient: JSON)
+ monitorListPage(statCode : StatCode, listType : ListType, user : JSON, monitor: Array<JSON>): Promise<String>
+ monitorSelectionPage(statCode: StatCode, user: JSON, monitor: Array<JSON>) : Promise<String>
+ monitorSettingPage(statCode: StatCode, user: JSON, interval: int) : Promise<String>
+ notFoundPage(user: JSON) : Promise<String>
+ noPatientPage(user: JSON) : Promise<String>
<<enum>>
ListType
TABLE = "table"
GRAPH = "graph"
TEXTUAL = "textual"
MonitorPage
- resourcePath : String
- statCode : StatCode
+ constructor(statCode: StatCode, resourcePath : String)
+ getStatCode() : StatCode
+ listPage(listType : ListType, user : JSON, monitor: Array<JSON>) : Promise<String>
+ selectionPage(user: JSON, monitor: Array<JSON>) : Promise<String>
+ settingPage(user: JSON, interval: int) : Promise<String>

Design Rationale

From the previous assignment, we have design the software to be able to handle multiple types of monitors and measurements. By following **Open-Closed Principle (OCP)**, we design the Monitor and Measurement class as an abstract class to reduce the rigidity for any new extension. Therefore, when we want to monitor Blood Pressure (55284-4), we can easily extend (Measurement and Monitor abstract class) and define its specific behaviours without violating the **Liskov Substitutability Principle (LSP)**

There are several refactoring tehniques were applied throughout the whole development process of the software. The major refactoring techniques were applied are **Push Down Refactoring Method**.

- Initially Patient were responsible on keeping track of multiple type of measurements, which result on Patient have a lot of responsibilities and method that mainly used for maintaning different types of measurement.
- We then realize that these responsibilities can be **Push Down** into another class (Measurements class) that are solely responsible for managing different type of measurements for a certain patient.
- The same thoughts and refactoring process are also applied to Practitioner class, and create a Monitors class which are responsible on managing different type of monitors.



Other than Push Down Refactoring Method, we also applied several other refactoring techniques such as:

- Self-encapsulate field refactoring method, there are several cases where duplicate code can be decrease by using self-encapsulate field, which make the code easier to track and maintains.
- Rename refactoring method, some method were renamed to have a clear definition of its responsibilites and return types.