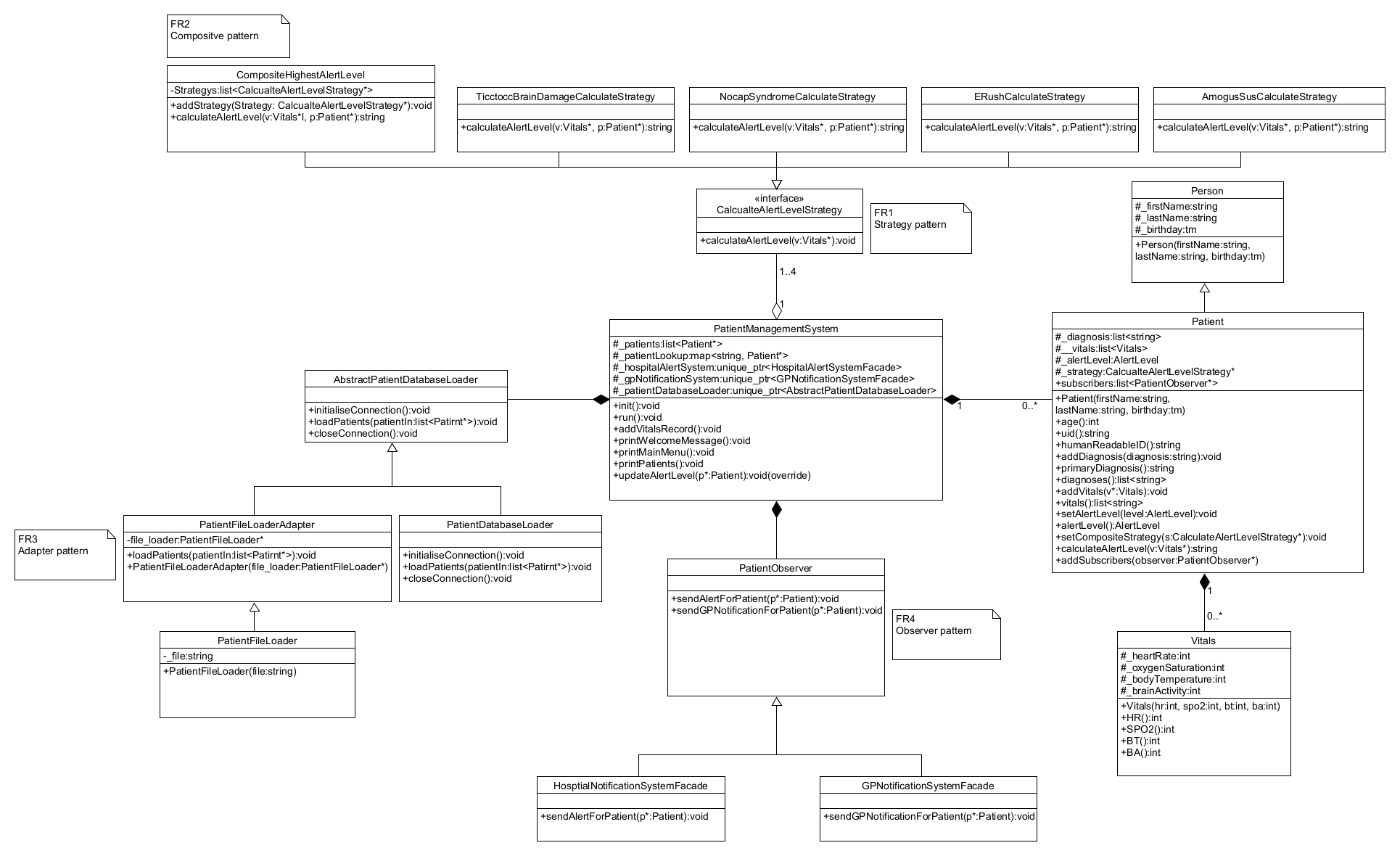
# Design document

Student name: Jingtian Wang

Student ID: 110321808

1. mail ID: [wanjy182@mymail.unisa.edu.au](mailto:wanjy182@mymail.unisa.edu.au)

Assignment 2 UML diagram



FR1: Calculate the patient alert level

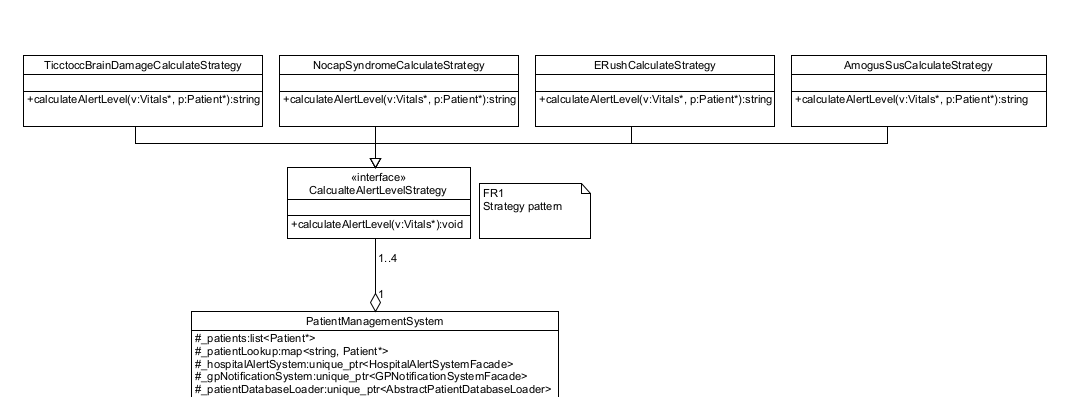


Figure 1: Strategy pattern

**Design pattern:** Strategy pattern

According to the alert level table each different disease will determine alert level according to different vitals, so In fr1 I am using the strategy pattern. The strategy pattern represented in figure 1, based on the table provided I have designed an interface CalculateAlertLevelStrategy and its 4 subclasses AmogusSusCalculateStrategy, ERushCalculateStrategy, NocapSyndromeCalculateStrategy, TicctoccBrainDromeCalculateStrategy, and TicctoccBrainDromeCalculateStrategy. NocapSyndromeCalculateStrategy and TicctoccBrainDamageCalculateStrategy. In these four subclasses, different algorithms are provided to detect the alert level depending on the table disease.

The CalculateAlertLevelStrategy class has only one vintrual function, calculateAlertLevel, and an enumerated variable, alertLevel, which stores the new alert level calculated by each strategy.

**How it works:**

1. The addVitalsRecord method in the PatientManagementSystem iterates over the \_diagnosis of the selected patient.

Depending on the patient's disease, create a corresponding strategy.

2. After the corresponding strategy is created, give the corresponding strategy to the corresponding patient through the setStrategy method of call Patient.

3. Call the patient's calculateAlertLevel method in the patient's addVitals method. Call the patient's calculateAlertLevel method in the patient's addVitals method.

calculateAlertLevel method will try to call the calculateAlertLevel method of the corresponding strategy just set in the PatientManagementSystem to calculate the new alert level.

**Git commits:**

* I first add the CalculateAlertLevelStrategy in commit 5c0cbf.
* I created four subclass in commits f0eb51, 2270a6, 09c7bb, 73b2fb.
* I override vintrual function in commits d65da1, 89d975, 5c7b88, c5256d
* Other changes and fix commits include: bb48c7, d8ed0a, 706186, 52bc99

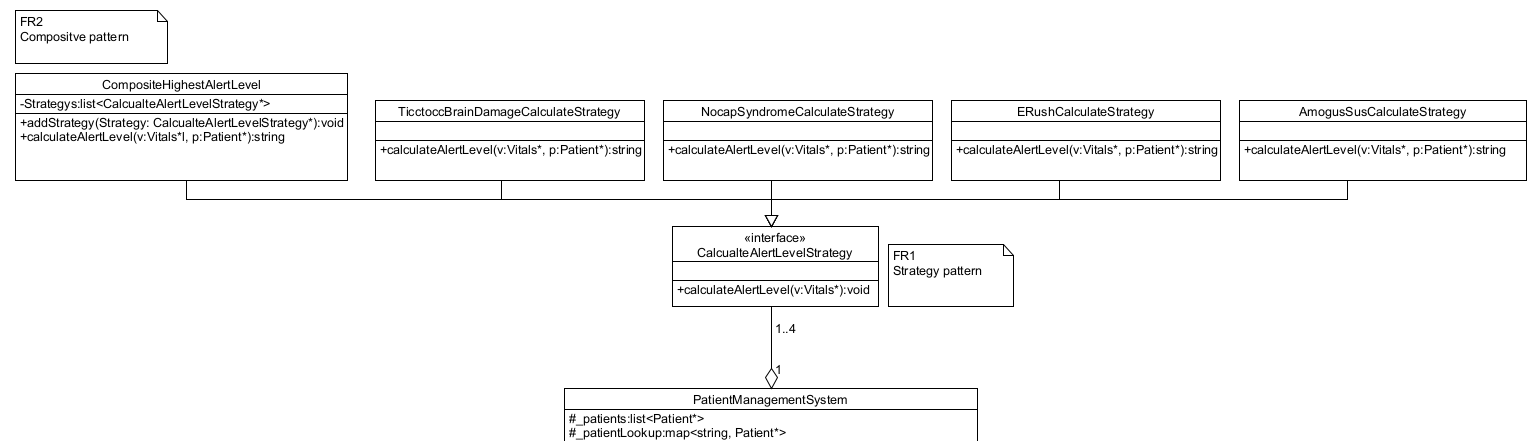
FR2: Calculate the alert level for all diseases a patient has

Figure 2: Composite pattern

**Design pattern:** Composite pattern

In fr2 I used the composite pattern. The Composite pattern represented in figure 2, I created a compositeCalculateStrategy class, which is also a subclass of CalculateAlertLevelStrategy. The difference is the new variable startegys which is a vector of type calculateAlertLevel and method addStrategy which is used to add strategies to the list of strategies.

Therefore, if you find that a patient has more than one case when traversing the \_diagnosis of the selected patient in the addVitalsRecord method in the PatientManagementSystem, you can add more than one strategy to an instance of the compositeCalculateStrategy class, and if there is only one disease, the compositeCalculateStrategy can also handle only one disease. This design is consistent with the concepts and expectations of the composite pattern.

**How it works:**

1. In the patient class I added a new variable \_strategys and initialised it in the constructor, which will be used to store all the disease strategies for the patient.

2. In the addVitualsRecord method of the PatientManagementSystem I first create an instance of the compositeCalculateStrategy class to store the possible multiple conditions, and as in fr1, in the addVitalsRecord method I iterate through the selected patient's The \_diagnosis is traversed in the addVitalsRecord method as in fr1 and the corresponding strategy is added to the \_strategys via the addStrategy method of the compositeCalculateStrategy class.

3. Unlike fr1, which sets a single disease to the patient, here we set the real column of the composite strategy to the patient. The next step is the same as the third step of fr1, but the difference is that the calculateAlertLevel of the compositeCalculateStrategy class traverses the entire \_strategys and calls the corresponding strategy in the addStrategy method of the compositeCalculateStrategy class. strategygys and call each strategy's calculateAlertLevel to compute a new alert level for each corresponding strategy, and then compare these alert levels to select the highest alert level.

**Git commits:**

* I first add CompositeHighestAlertLevel class in commit da1c7e.
* I update calculateAlertLevel method in CompositeHighestAlertLevel class in commit e145dd.
* I create calculateAlertLevel method in Patient class in commit 1719dc.
* I update a looping in addVitalsRecord method of PatientManagementSystem class to find current patient’s diagnoses in commit 86afb8.
* Other changes and fix commits include: aab704.

FR3: Load patients from file

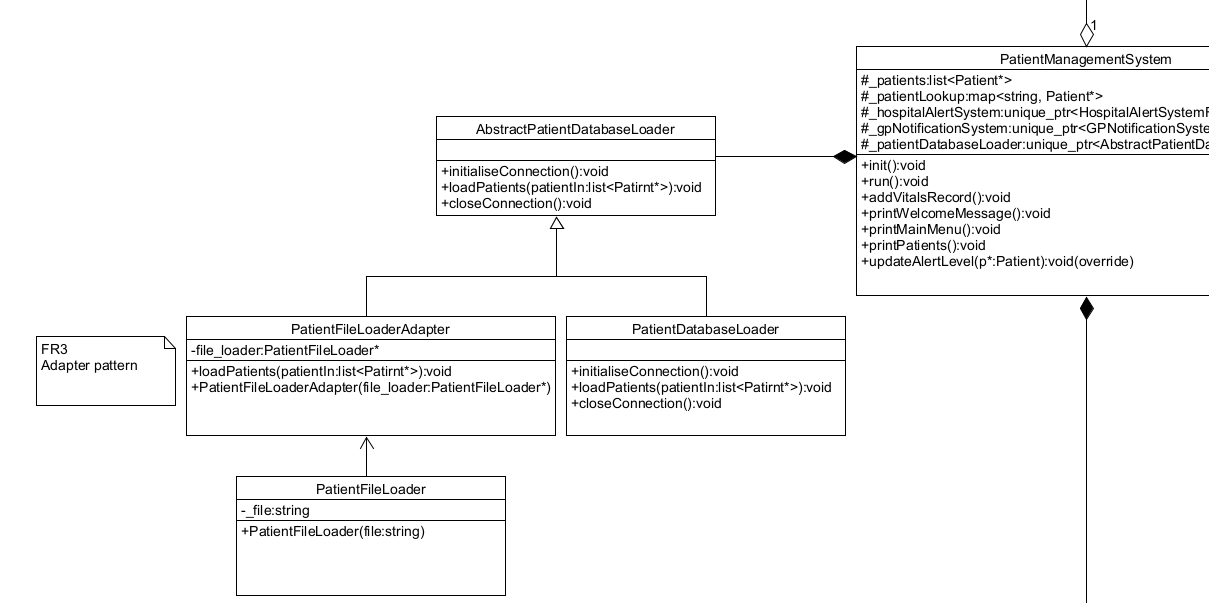


Figure 3: Adapter pattern

**Design Pattern:** Adapter pattern

PatientManagermentSystem now not only need to load data from database, and also need to try to load patient information through txt file, so I use adapter pattern to try to use PatientFileLoaderAdapter to adapt to the PatientFileLoader. As you can see in Figure 3, not only is PatientDatabaseLoader a subclass of AbstractPatientDatabaseLoader, but PatientFileLoaderAdapter is also a subclass of AbstractPatientDatabaseLoader. It is used to transform the PatientFileLoader and load data from it.

**How it works:**

1. The constructor of PatientFileLoaderAdapter accepts a parameter file\_loader which is a pointerof PatientFileLoader of type PatientFileLoader. PatientFileLoaderAdapter itself includes a private variable file\_loader Its type is pointer of PatientFileLoader.

2. The parameter file\_loader is given to the variable file\_loader through the constructor to complete the constructor, and there is also a destructor to free the memory of the file\_loader pointer.

3.PatientFileLoaderAdapter will override AbstractPatientDatabaseLoader's method loadPatients(std::vector<Patient\*>& patientIn), and in this method will call variable file\_loader's loadPatientFile() method, this method will read infomation from current txt file and then returns a list of pointers of type Patient and gives it to loadPatients' parameter patientIn, thus completing the conversion of the PatientFileLoader into the PatientDatabaseLoader.

4.Modify the following code in PatientManagementSystem: \_patientDatabaseLoader(std::make\_unique<PatientDatabaseLoade>(),

to:

\_patientDatabaseLoader(std::make\_unique<PatientFileLoaderAdapter>(new PatientFileLoader(‘patients.txt’))),

We replace <PatientDatabaseLoader> with <PatientFileLoaderAdapter> and add the parameter PatientFileLoader(‘patients.txt’) to initialise a pointer to the PatientFileLoader class and try to open the file ‘patients.txt’, and finally calling loadPatients mrthod to load the data.

**Git commits:**

* I first create the PatientFileLoaderAdapter in commit 27ee5a.
* I finished PatientFileLoader in commit d67ae4 af3099.
* Other commits include: 7990eb, 3cbf65.

FR4: Alert the Hosptials and GPs

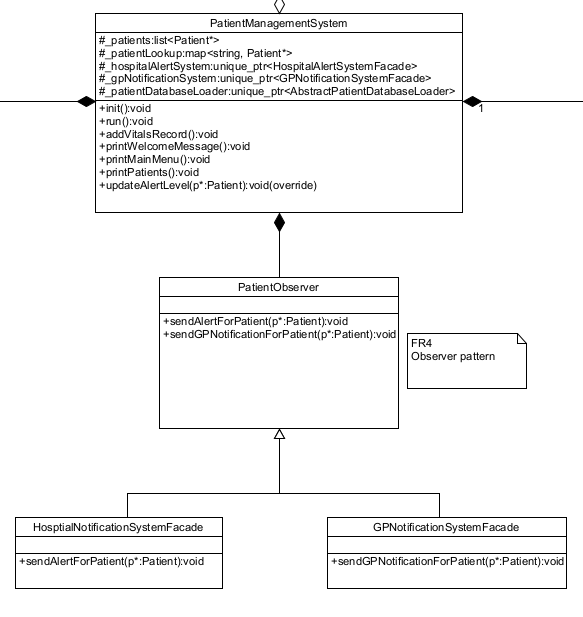


Figure 4: Observer pattern

**Design pattern:** Observer pattern

Whenever the patient's alert level changes, the system will try to notify the hospital (alert level = red) or the GP (alert level >= orange) so here you can design it using observer pattern. The figure 4 represented observer pattern. The observation object is patient and the observer is HospitalNotificationSystemFacade and GPNotificationSystemFacade. I created a new class called PatientObserver now HospitalNotificationSystemFacade and GPNotificationSystemFacade are subclasses of this class to make it easier to call their methods sendAlertForPatient and sendGPNotificationForPatient consistently.

**How it works:**

1. Now the Paitent class has an additional public variable subscribers of type PatientObserver, which is used to store each patient's observer. And the Patient class has an additional method addSubscribers to add observers to variable subscribers.

2. The init method in PatientManagementSystem iterates through each patient and use addSubscribers method to add instances of HospitalNotificationSystemFacade and GPNotificationSystemFacade to each patient as subscribers.

3. Iterate through the list of \_subscribers for the current patient in the addVitalsRecord method of the PatientManagementSystem and for each observer in the subscribers call sendAlertForPatient and sendGPNotificationForPatient. Therefore each time record the vital or the alert level change immediately, the notification will send when equal or greater than orange(GP) or red(hospital).

**Git commits:**

* I first create the PatientObserver in commit 326e66.
* I finished logic in PatientManagementSystem in commit 486477.
* Other commits include: 4aa834, 96f51f, ad1d80.