Name - Soham Balkothe
Student Id - 1001500980
Email - svb0980@mavs.uta.edu

Homework 3

1. Identify design problems you would encounter when developing the software. Briefly explain each of these design problems.

Some of the design problems that can be encountered developing this software are:

1. To design a low-coupled mechanism to add or remove many-to-one dependencies between the registered healthcare providers and the monitoring device as a one-to-many relationship between the monitoring device and the registered healthcare providers is a tightly coupled relationship.

2. When the patient's state changes, multiple registered healthcare providers related to the patient must be updated and notified. A message should be broadcasted to the registered healthcare providers mapped to the patient. This feature requires the monitoring system to perform differently depending on the patient's state and show different behavior i.e. alert the central system rather than just sending the patient's parameters periodically.

3. Different health monitoring may perform the same tasks but may have different interfaces to communicate with different central systems. Our central system will not be compatible with all the health monitoring devices available in the market which are able to record the same parameters.

4. While providing the patient's parameters to the registered healthcare providers, access to the component should be allowed without exposing its internal details. There is a fundamental violation of the SOLID design principles here due to the tight coupling between the registered healthcare providers and the central system as healthcare providers need to make multiple interactions with each service implemented by central system classes.

5. The classes, interfaces, and methods on the central system side can differ from those on the healthcare providers' side which will increase the complexity of the code and make the readability of the code low.

2. Identify exactly two situation-specific (i.e., Gang-of-Four patterns) that can solve some or all of the design problems you identify in the last bullet. Explain how each of the two patterns solves a design problem.

a) Observer pattern:

The observer pattern can be used to solve the design problem of triggering alerts and notifying healthcare providers. The observer pattern defines a one-to-many dependency between objects, where the central system object notifies all the observers or the registered healthcare providers when the patient's state changes. When the central system receives an updated state(outside the desirable parameters) of the patient, it issues and sends a warning message to all the registered healthcare providers mapped with the patient and updates their states. The observers or the registered healthcare providers then can request the patient's data from the central system and process it and also can perform the required actions such as raising an alarm or displaying a message. This all can be achieved by using the observer pattern.

By using the observer pattern while implementing this system, we are able to solve the design problem 1 mentioned in the previous question. It helps us to implement a low-coupled system to add or remove many-to-one dependencies between the registered healthcare providers. It also helps us broadcast a message to multiple observers solving the high-coupling problem between the multiple observers and the central system (one to many high coupling problem). It makes it easier to add or remove a new observer for a given patient. We can add, delete or update any new parameter or functionality in the system without undergoing a lot of code change for multiple observers.

b) Adapter pattern:

The Adapter pattern can be used to address the design problem of data transmission. In this pattern, the small device and the center server use different interfaces or protocols for transmitting data. An adapter class can be used to convert the data from one format to another, allowing the small device to communicate with the central server seamlessly. The Adapter pattern provides a way to integrate incompatible interfaces and ensures that the patient data is accurately transmitted.
The adapter pattern helps us to solve problem number 3 by implementing an adapter class that implements our central system and thus communicates with it easily.

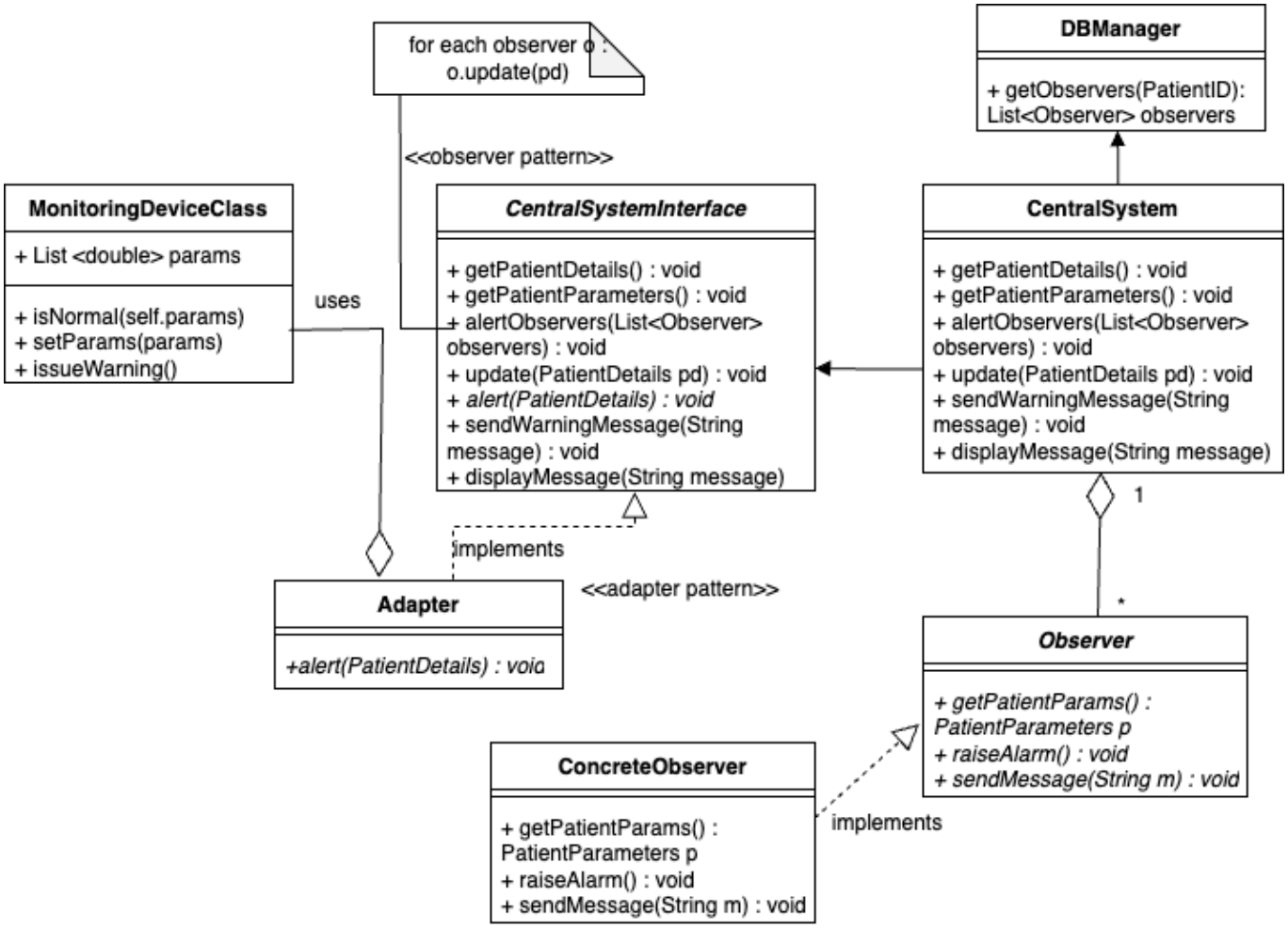
The other patterns that can be used to solve the other mentioned problems are (Please judge on the above 2 patterns not these ones):

1) State pattern:
The state pattern can be used to solve the design problem of comparing incoming patient data with the set limits. The state pattern allows an object to alter its behavior when its internal state changes. In this case, the patient monitoring device would have a state machine that would transition between different states depending on the incoming patient data. When a patient's parameter falls outside the set limits, the device would transition to a "warning" state, which would trigger the alert system.

2) Facade pattern:

This pattern can be used to provide a simple and easy-to-use interface for healthcare providers to set limits, view patient data and take appropriate action. The complex system of collecting, transmitting, processing, and storing data can be hidden behind a facade, which presents a simplified interface to the user. This can improve usability and reduce the risk of errors.

3. Draw a design class diagram (in UML) to show the classes of the patient monitoring software including the participants of the patterns identified in the bullet. Describe the roles and responsibilities of the classes in the design class diagram.



Roles of the classes :

CentralSystemInterface: This class defines the functions for the central system class.

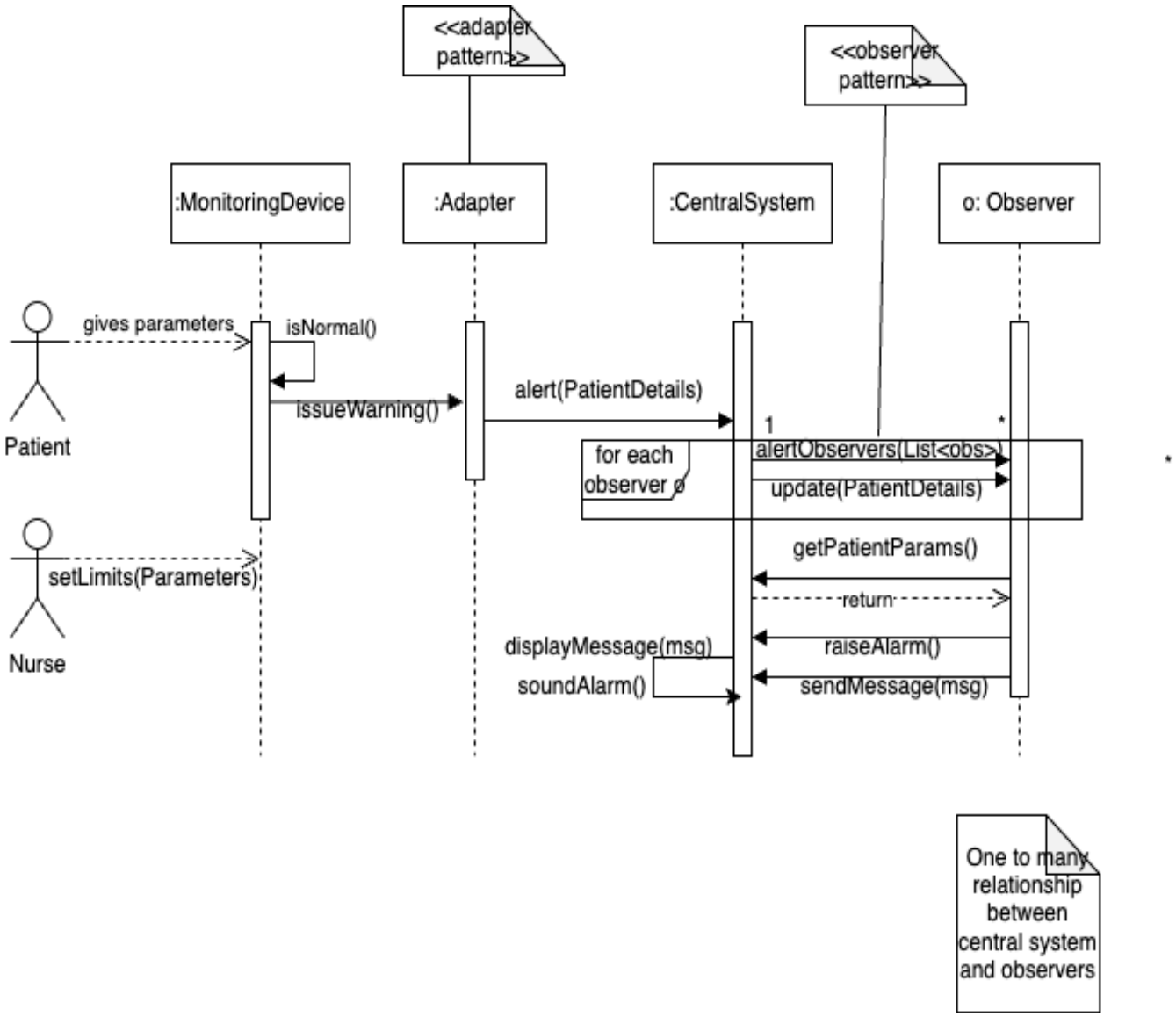
Observer : This class defines the methods for the observer interface.

ConcreteObserver : This classes extend the CentralSystemInterface class to add their own behaviour such as getPatientParams, raiseAlarm, sendMessage methods.

Adapter: The adapter implements the method from classCentralSystemInterface so that the interfaces of monitoring system and central system becomes compatible and can communicate efficiently.

MonitioringDeviceClass: This class is the implementation f the monitoring device. It contains a list of parameters of the patient which it stores and updates regularly. The method isNormal() checks if the parameters are in the limits set by the nurse using setParams() method. It issues a warning message to central system using issueWarning() method. The observer pattern has been implemented while alerting the observers and the adapter pattern is used when the MNS communicates with the CNS.

4. Draw a design sequence diagram to show object interaction in the patient monitoring system. Based on the design sequence diagram, explain how objects interact with each other to accomplish the functions of the patient monitoring system.



Explaination:

The patient sends the params regularly to the CNS(Central System). If the parameters are out of limits set by the nurse using setParams(params) method, the MNS (monitoring system) issues a warning to the CNS using the adapter method. The CNS retrieves all the related observers from the DB and alerts them using the alertObservers() method. Different observers then may demand different types of actions to be performed such as get patient health details using getPatientParams() method, raise an alarm using raiseAlarm() method and display a message using sendMessage() method. The observer pattern has been implemented while alerting the observers and the adapter pattern is used when the MNS communicates with the CNS.