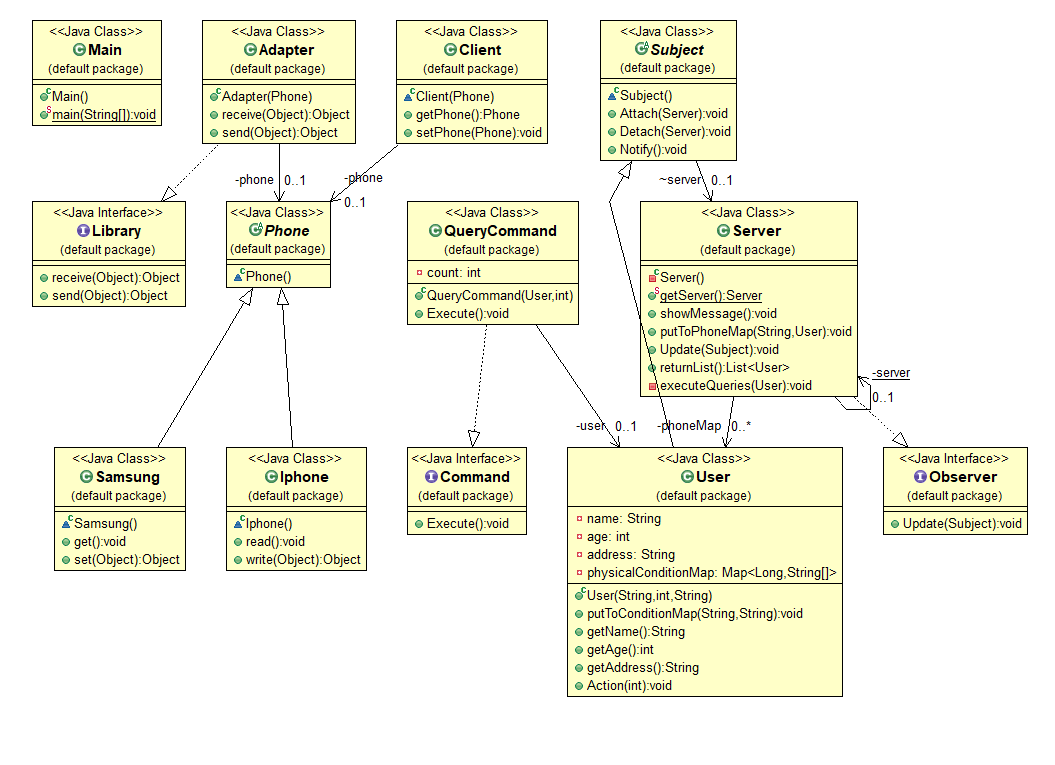
THOUGHT PROCESS

While we had been going through the project description something has been built by time. One thing we have noticed at first was the Health Ministry database should be simulated by us and we have decided that we can call it Server class and implement it in a singleton pattern. Since there will be different brands of phones, we thought that there should be an adapter pattern implementation to use against the default Library class methods. Then, we understood from the server goes over the list of registered phones that we can implement an iterator pattern for this part. Users physical conditions will be monitored by the Server and the Server will be notified from user’s state changes. So, we thought that observer pattern will fit into this scenario. In the project description three different queries are wanted to be executed and we decided that we can use a command pattern implementation.

UML CLASS DIAGRAM



WHAT EACH CLASS DOES?

**Library**: Library interface is the Target in terms of Adapter Pattern and has receive and send methods with an object as the parameter.

**Adapter**: This class is the adapter for different kinds of mobile operating systems which uses different implementations than the Library.

**OperatingSystem**: This is just an abstract class to use for our operating system classes.

**Samsung and Iphone**: These are the operating system classes which have their own interfaces.

**Observer**: Observer is just an interface for our concrete observer which is the Server class in our case and has Update method with a subject as parameter.

**Subject**: Subject class is an abstract class which keeps implementations for our Observer pattern’s subject. The subject will be the User class.

**Command**: This is an interface to define the methods that will be used by the implementers.

**QueryCommand**: This class implements the Command interface, so this class is our concrete command, also will be the connector between the User and the Server objects.

**Server**: Server class stores the list of registered phone numbers and their associated Users. This class is our concrete observer, so it implements the Observer interface. It receives updates from user’s condition changes. Also, we have implemented an Iterator pattern to be able to iterate through registered phones and users associated with them. This class asks the commands that are set in the User class, which is a receiver for our Command pattern, to be executed. This class can only have one instance at runtime, so it fits to Singleton pattern.

**User**: User class is our concrete subject to send updates to the Server. It stores the physical conditions of the users in a map with the timestamps. This class is the receiver for Command pattern that we have implemented in our project. It executes actions called by the Invoker.

**MobilePhone**: This class simulates the mobile phone of the user with the associated mobile operating system.

**Main**: This is our test class which holds the main method. It instantiates the Server, mobile phones, sets the adapter for the operating systems and users. Then, does the specified operations in the Server class.

OUTPUT SCREENSHOTS