SSAD: Assignment 3

REPORT

Team Project number 21: Smart health prediction system

Adela Krylova, Ayhem Bouabib, Made Oka Resia Wedamerta, Vladimir Zelenokor

27. 11. 2021

1. Application Description

The app serves users with two main functionalities -

- making predictions about diseases based on symptoms
- finding doctors nearby
- Providing some basic information about diseases
- Searching an insurance company

The app uses a simple database consisting of Diseases and their symptoms. In order to predict the illness, it matches the symptoms with the database. It returns the disease with the highest number of matches. The doctor finding system compares the address of the patient with the address of the doctors inside the database. It returns the nearest found doctor. Similarly, the system also finds the Insurance company.

Moreover, there are 4 levels of subscription plans, which differ in the number of services.

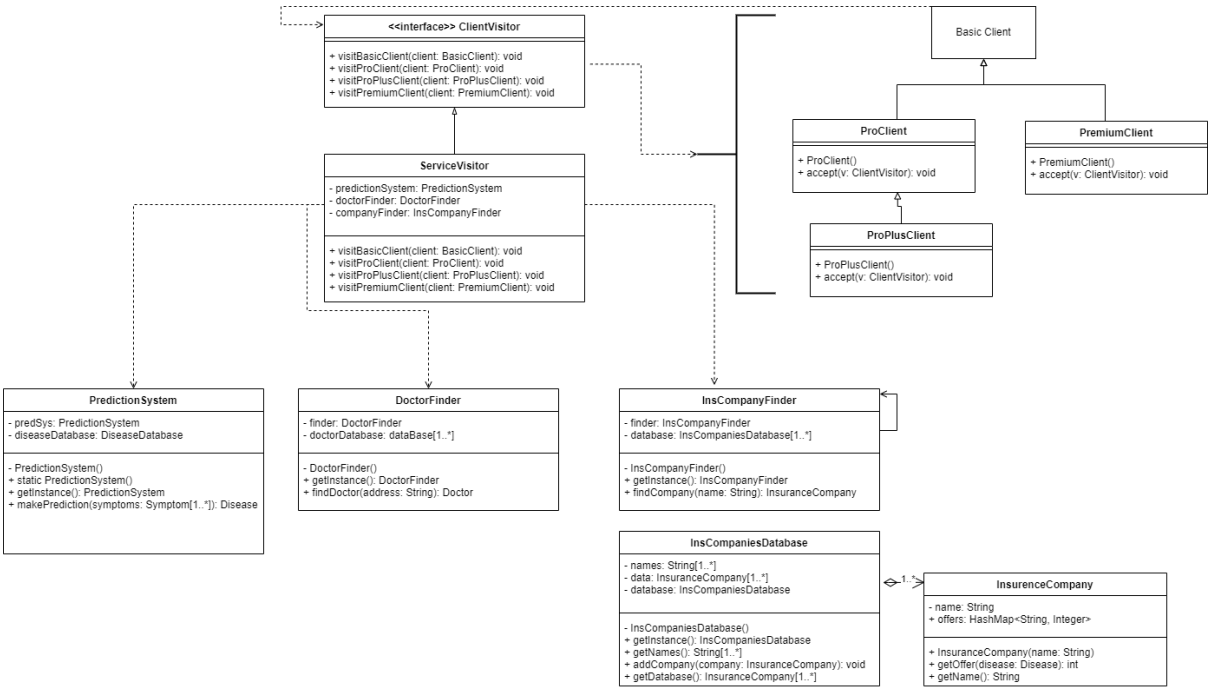
- **Standard** - allows user to make a disease prediction
- **Pro** - disease prediction + advises on the disease treatment
- **Pro+** - disease prediction + find doctor + advises on the disease treatment
- **Premium** - disease prediction + insurance verification

Updates:

Some changes were made, in particular:

- The Visitor pattern was implemented so that dealing with different types of users and updating them was easier and clearer.
- The naming convention was fixed according to the Java coding style recommendation

2. UML Diagram



Please click here to see the diagram in full size.

Remark: In order to make the diagram easy to read some classes from the previous assignment were not drawn.

Visitor Design Pattern

In order to update the users in a more effective manner we decided to implement the *Visitor pattern*. Visitor design pattern is a behavioral pattern which separates algorithms from the objects on which they operate. In our case, the services will be provided to the clients class without exposing the hidden details of the different functionalities. We decided to implement this pattern because it helps us to perform the same operation(e.g. providing service to all subscribing user classes without exposing the hidden details).
Therefore, we do not call every method separately on every object but we can use the *accept(Visitor)* method for every user class. Moreover, implementing the Visitor helps us in case of future modification of the code. The Visitor DP enables us to implement the new feature independently of the user hierarchy. Consequently, If a new functionality was to be added to the system, the Client classes

would not be modified. That prevents the code from potentially malicious modifications and helps us to keep the code clear.

3. Program Description

a. Interface ClientVisitor

The ClientService is a common interface to all possible visitors that could possibly interact with different types of clients (BasicClient class and its derived classes).

b. ServiceVisitor

The ServiceVisitor class is a concrete ClientVisitor. It is responsible for providing the service that corresponds to the Client's membership/level. Such a task is assured through the following methods:

c. visitBasicClient(BasicClient client)

This method predicts the client's disease which is the main service provided by the Smart Health System. The prediction functionality is delegated to a private PredicitionSystem field.

d. visitProClient(ProClient client)

This method provides services for Pro clients. In addition to the disease prediction assured by calling the visitBasicClient(), this method offers the client a set of advice and recommendations specific to the prediction's result.

e. visitProPlusClient(ProPlusClient client)

This method provides services to PRO+ clients. In addition to disease prediction and recommendations ensured by calling visitProClient(), the method finds the nearest doctor according to the client's address. The functionality is delegated to the private doctorFinder field.

f. visitPremiumClient(PremiumClient client)

This method serves Premium clients. This method verifies the client's insurance company. If the latter is one of the application's partners, a percentage of the treatment cost is covered. The functionality is delegated to the private

InsCompanyFinder field. The disease prediction is provided by calling visitBasicClient().

4. Conclusion

Homework 3 presents an extension to the previous prediction system application. The Visitor pattern was added so that the code was easily maintainable.