1. Application Description

The app serves users with two main functionalities -

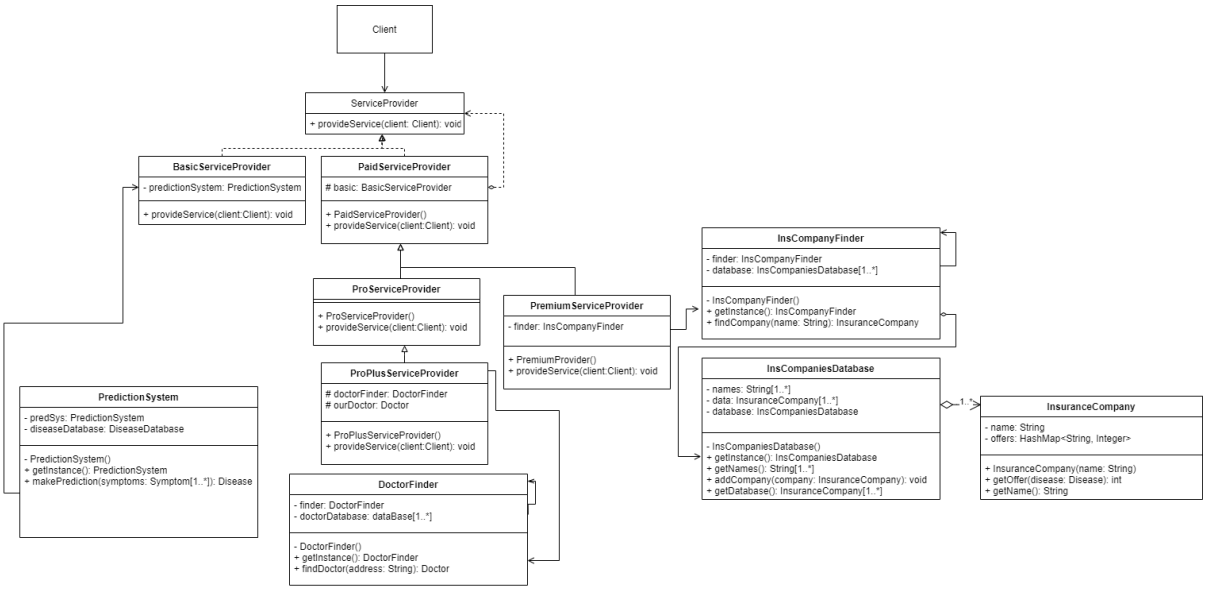
- making predictions about diseases based on symptoms
- finding doctors nearby
- Providing some basic information about diseases
- Searching an insurance company

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 and returns the disease with the highest number of matches. For the doctor finding system, it compares the address of the patient with the address of the doctors inside the database and 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

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.

For implementing the 4 levels of subscription we decided to follow the Decorator pattern. The Decorator Design Pattern is a structural design pattern that allows us το attach new behaviours to objects by placing these objects inside special wrapper objects that contain the behaviours. Since every level of the subscription contains all features of the previous levels, we only need to add new behaviour without editing the old one. Therefore, the Decorator is the right pattern for our project.

3. Program Description

a. Interface ServiceProvider

The ServiceProvider interface is a common interface to all classes that provide services to clients. With a Client object as an argument, the ServiceProvider object will operate according to the client's membership.

b. BasicServiceProvider

BasicServiceProvider is a ServiceProvider. Its main service is predicting the client's disease. The functionality is delegated to a private PredictionSystem field.

c. PaidServiceProvider

An abstract DECORATOR for the BasicServiceProvider class. Derived classes implement the provideService method in such a way that the work is delegated to the super.provideService() while performing additional functionality either before or after the call.

d. PremiumServiceProvider

The PremiumServiceProvider is a concrete decorator created for Premium clients before predicting the disease, which is ensured by super.provideService(). This class verifies the user'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.

e. ProServiceProvider

The ProServiceProvider is a concrete Decorator for BasicServiceProvider serving PRO clients. In addition to disease prediction provided by super.provideService(), this class offers the client a set of advice and recommendations specific to the prediction's result.

f. ProPlusSErviceProvider

The ProPlusServiceProvider is a concrete decorator created for PRO+ clients In addition to disease prediction ensured by super.provideService(), this class finds the nearest doctor according to the client's address. The functionality is delegated to the protected doctorFinder field.

g. Insurance

In order to handle the process connected with the Insurance companies we introduce 3 classes:

- InsCompaniesDatabase
- InsCompanyFinder
- InsuranceCompany

4. Conclusion

Homework 2 presents an extension to the previous prediction system application. The Decorator design pattern was implemented for different subscription plans.