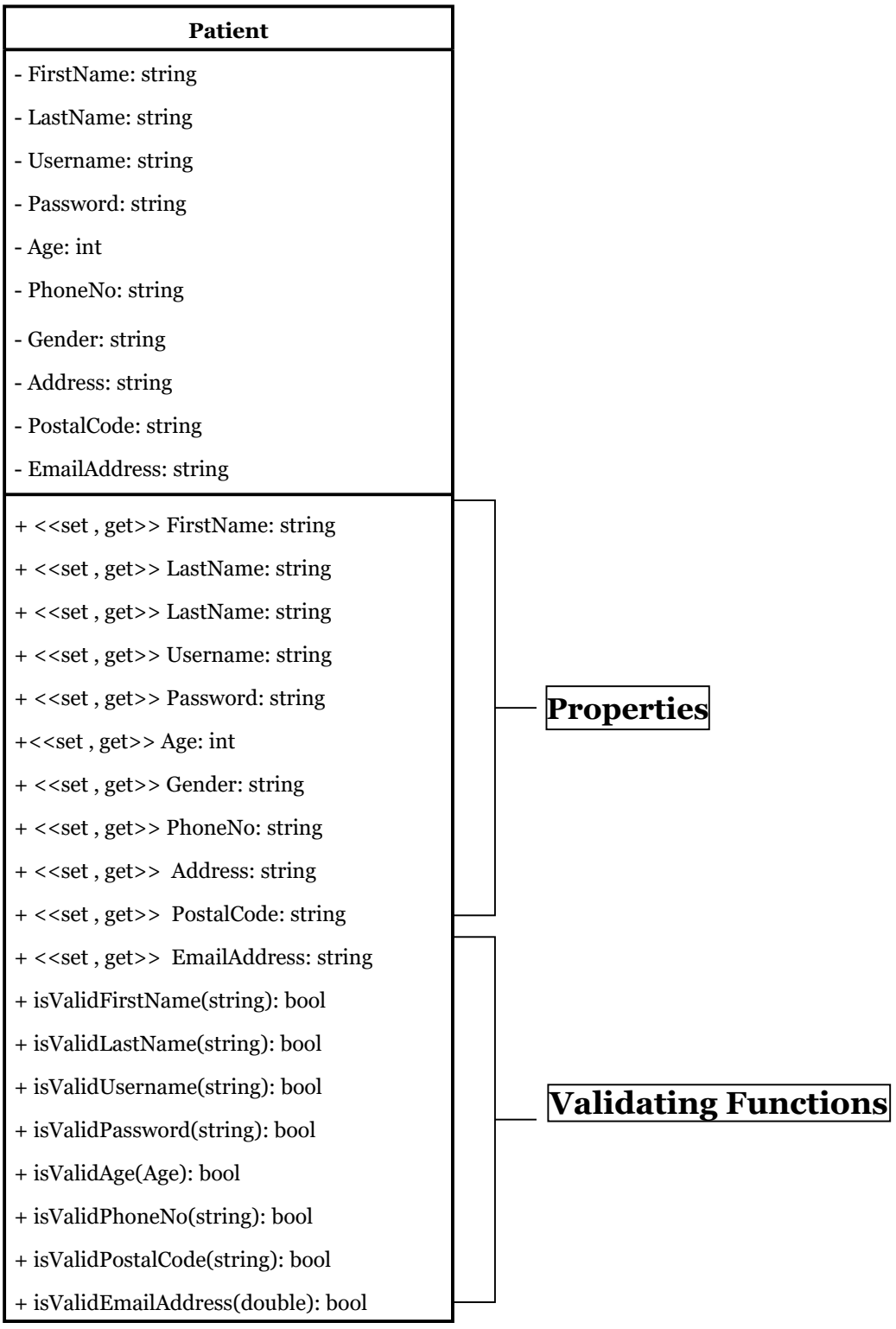
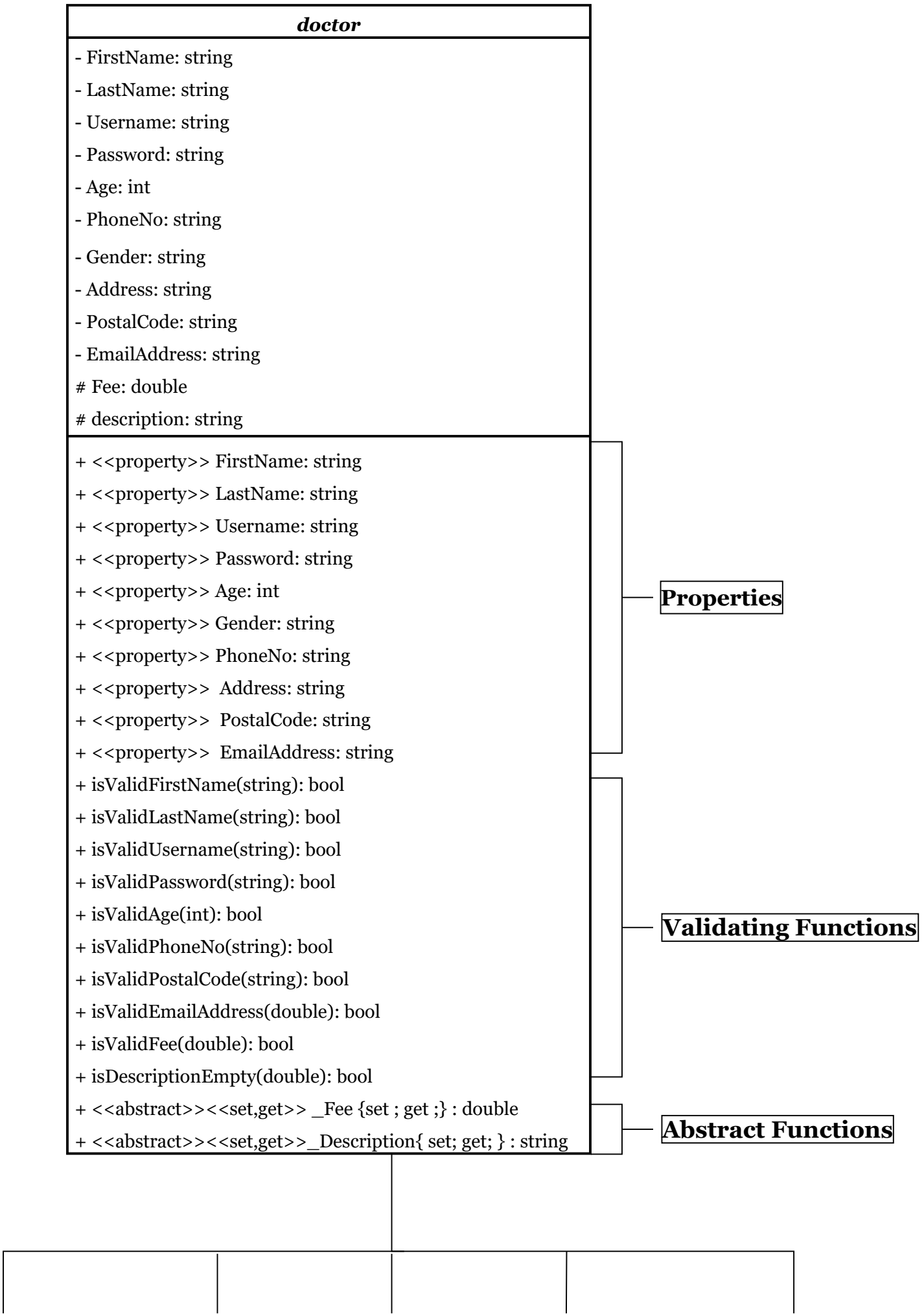
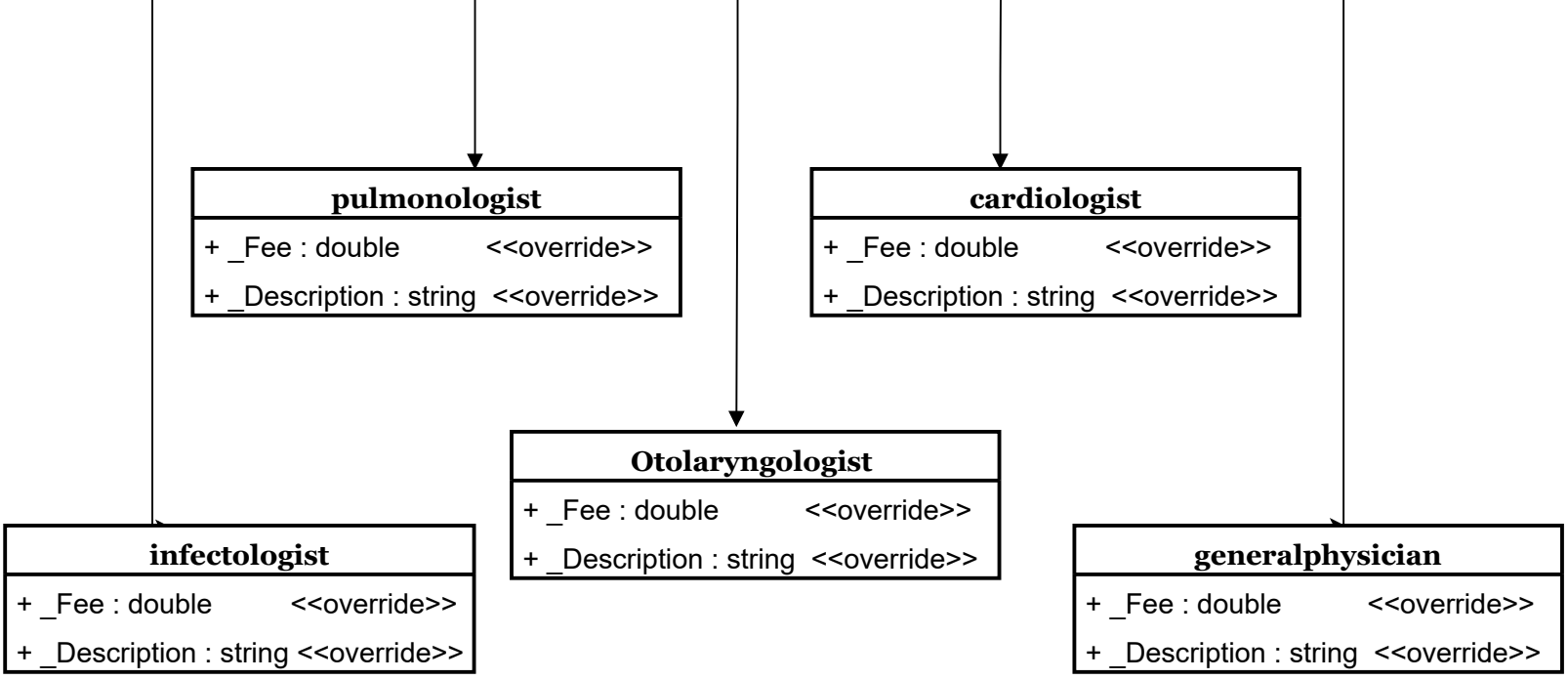


DESCRIPTION:

Patient class is designed to depict patient's entity. The attributes present in the patient class are related to the general info of a patient such as first name,last name,age, gender, address etc. This class contains all the attributes that a patient must provide while sign up. All the member variables are private and for accessing and mutating public properties have been created. Bool returning validating functions have been made for each class member variable, so that it restrict user from entering meaningless information. For E.g. (First Name and Last Name can only have upper or lower characters; Mobile No must be in 03xx-xxxxxxx format and cannot have characters and cannot exceed 12 digits, username cannot have spaces etc)


doctor
- FirstName: string
- LastName: string
- Username: string
- Password: string
- Age: int
- PhoneNo: string
- Gender: string
- Address: string
- PostalCode: string
- EmailAddress: string
Fee: double
description: string
+ <<property>> FirstName: string
+ <<property>> LastName: string
+ <<property>> Username: string
+ <<property>> Password: string
+ <<property>> Age: int
+ <<property>> Gender: string
+ <<property>> PhoneNo: string
+ <<property>> Address: string
+ <<property>> PostalCode: string
+ <<property>> EmailAddress: string
+ isValidFirstName(string): bool
+ isValidLastName(string): bool
+ isValidUsername(string): bool
+ isValidPassword(string): bool
+ isValidAge(int): bool
+ isValidPhoneNo(string): bool
+ isValidPostalCode(string): bool
+ isValidEmailAddress(double): bool
+ isValidFee(double): bool
+ isDescriptionEmpty(double): bool
+ <<abstract>><<set,get>> _Fee {set ; get ;} : double
+ <<abstract>><<set,get>>_Description{ set; get; } : string
Properties
Validating Functions
Abstract Functions



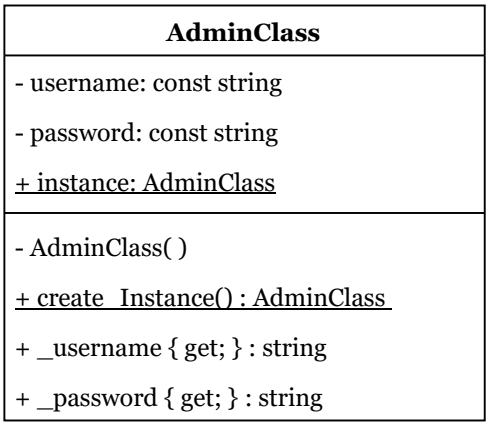
Parent Class: doctor

Doctor class clearly depicts doctor's entity. The attributes present in the doctor class are related to the general info of a doctor such as first name, last name, age, gender, address etc. This class contains all the attributes that a patient must provide while sign up. All the member variables are private except fee and description which have protected access level. For accessing and mutating class variables public properties have been formed. This Class has been made abstract containing abstract properties of Fee and description, whose implementation is been provided in the child classes. Bool returning validating functions have been made for each class member variable, so that it restrict user from entering meaningless information. For E.g. (First Name and Last Name can only have upper or lower characters; Mobile No must be in 03xx-xxxxxxx format and cannot have characters and cannot exceed 12 digits, username cannot have spaces etc)

Child Classes: Pulmonologists, Cardiologists, Infectologist, Otolaryngologist and General Physician

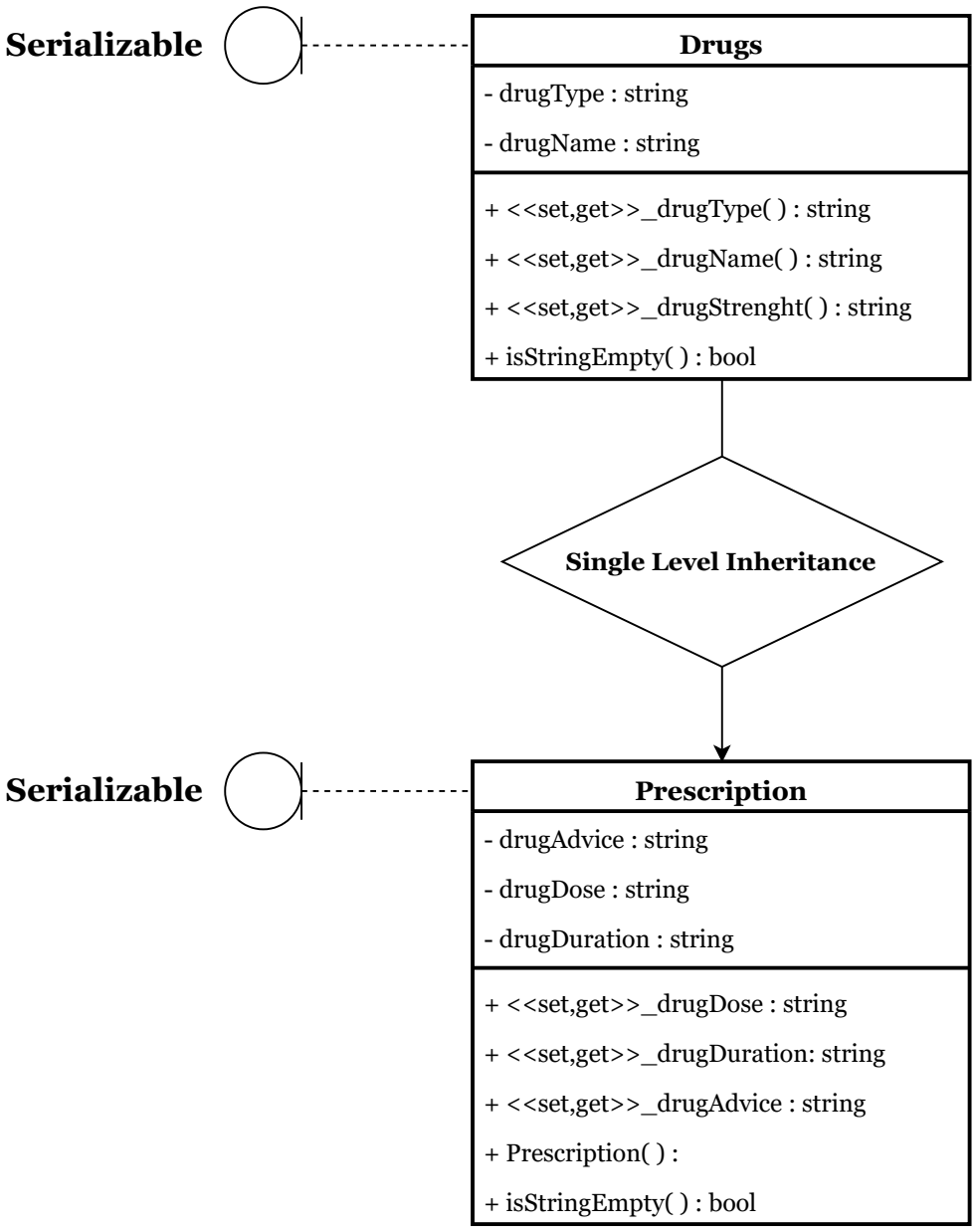
These classes are child classes of doctor class all each having a single inheritance, but overall there is a hierarchal inheritance between the base and child classes. Above mentioned classes are different specialties of a doctor. The abstract properties (fees and description) are implemented by these classes. The fees and general description of a particular doctor is set in the constructor based depending upon the type of specialist.

Singleton Design Pattern



Description:

Singleton design pattern have been used for the admin class because we want to have only one admin running the application i.e. only one instance of admin can be created throughout the entire application. Admin class serves the purpose of saving the username and password of admin required at the time of login. Also, both username and password variables are made constant so, so that their value cannot be altered throughout the program.

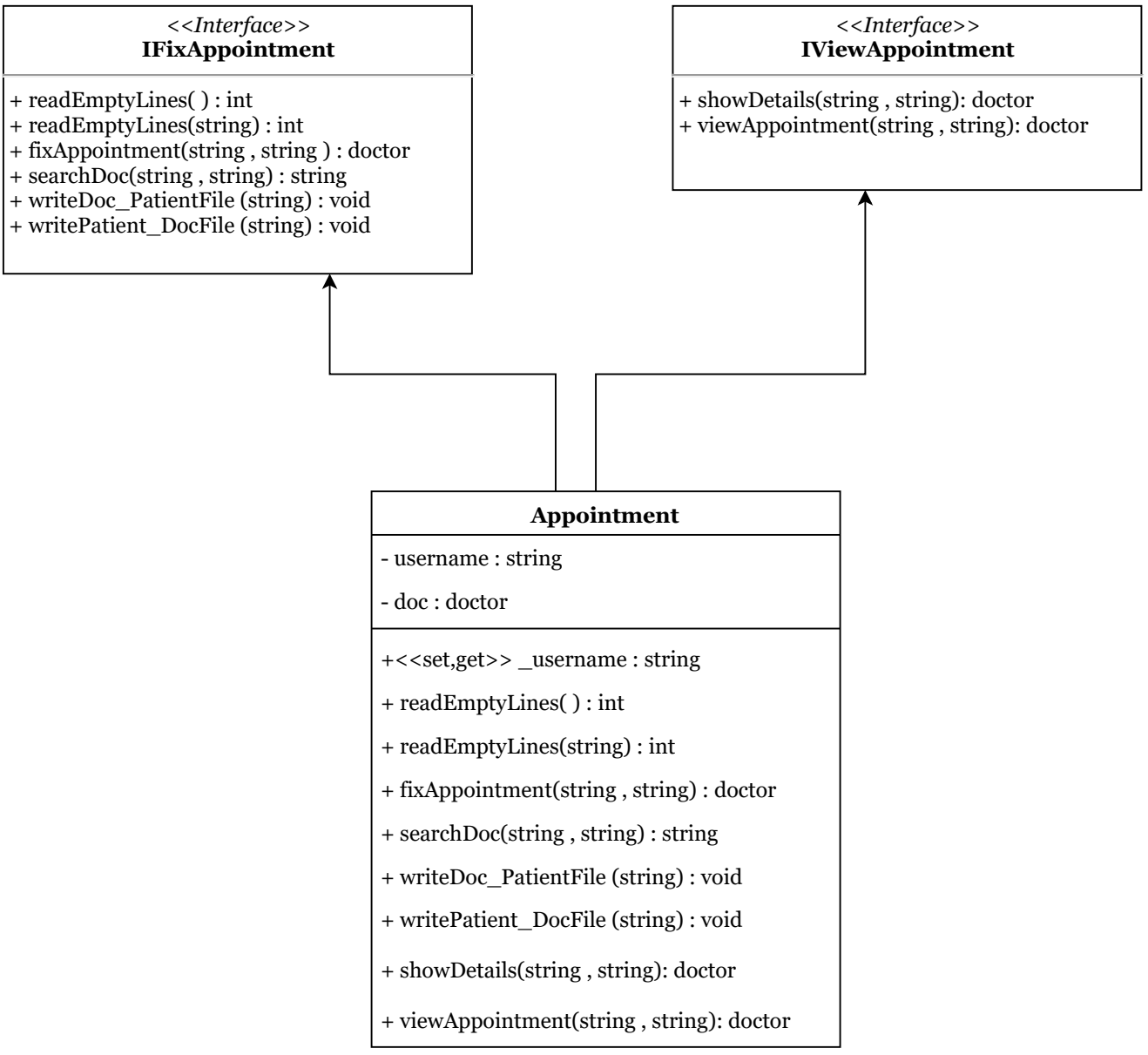


Parent Class : Drugs

This class contains the attributes needed to define a medicinal drug, drug type (tab, syrup, powder etc) drug name and drug strength (100mg, 250mg). All the data members are private and public properties have been made for accessing and mutating the data members. This class is "Serializable" i.e. we are writing and reading the complete object of drug class into the file. This class is important for maintaining a list of drugs and for performing CRUD operations.

Child Class: Prescription

Prescription class is an extension of the drug class. In addition to the data members of drug class it has three more data members (drug dose, drug duration, drug advice), so that attributes of a medical prescription can be easily depicted. There is a single level inheritance between the drug and prescription class, the inheritance is done because prescription class is modification of drug class through the addition three data members. All the data members are private and public properties have been made for accessing and mutating the data members. This class is "Serializable" i.e. we are writing and reading the complete object of prescription class into the file. This class is important for properly maintaining patient's prescription written by the doctor and later displaying the prescription in the tabular format.



Interface: IViewAppointment

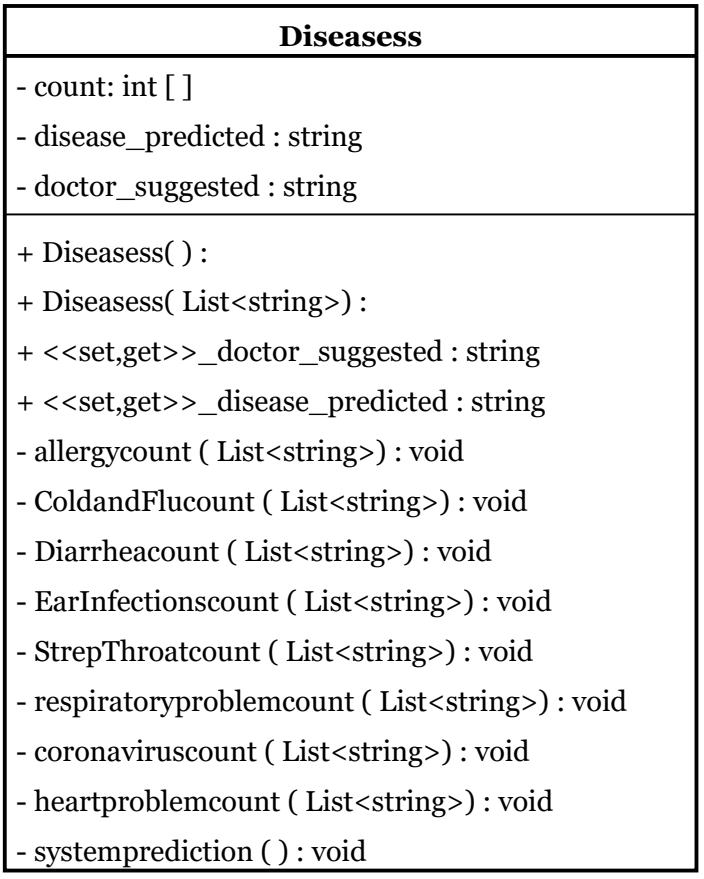
The interface IViewAppointment contains the declarations of functions that need to be implemented when the details regarding patient's appointment is required to be displayed. This interface have been implemented in the Appointment class.

Interface: IFixAppointment

The interface IFixAppointment contains the declarations of functions that are necessary to be implemented for fixing the appointment of patient. It is also implemented in the Appointment class.

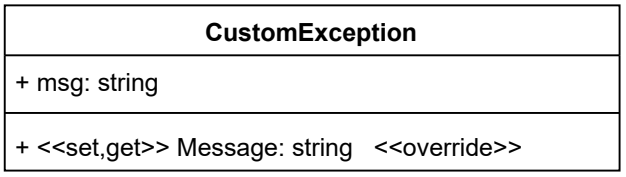
Class: Appointment

This class implements two interfaces that are IFixAppointment and IViewAppointment.By implementing these interfaces Appointment class accomplishes the task of fixing an appointment for the patient after searching appropriate specialist depending upon the symptoms and disease predicted. Secondly it also shows the details of appointments (name of the doctor, specialty, fees etc).



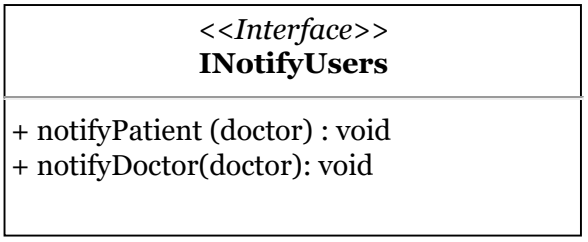
Description:

This class is the heart of our project, having hardcoded symptoms in lists of string type.The reason behind making it list of string type instead of arrays of strings is that we are unaware about the number of symptoms patient would be selecting at runtime from the symptoms check list. The main purpose of making this class was to represent a particular disease. Each disease has some name, symptoms and specialist doctor associated with it, that's why we made them the attributes of disease class. Each member function has list of symptoms related to a certain disease. The symptoms entered by the patient are checked through each function and finally a disease and Specialist is suggested by the systemPrediction function.



Class: CustomException

This class has been inherited from the Exception class, the Message property have been overridden for displaying custom messages when an exception isencountered.



Interface: INotifyUsers

This interface has declarations of functions that are necessary to be implemented for notifying the patient and doctor through email about appointment details that has been fixed by the system. It is implemented in the “SendingNotifications” form.