29/12/2021

SMART CLINIC

DESIGN PATTERNS

Dr. Diana Tharwat

TEAM MEMBERS

محمد إبراهيم عبداللطيف حسين (SE) TL

حسن السيد علي حسن الشرقاوي (SE)

حازم محمد رفعت محمد سيف (SE)

عادل توفيق صفي الدين غانم (CS)

محمد عادل صلاح إبراهيم محمد (SE)

The Content

Overview

Smart Clinic system is a medical management system for managing some of important aspects of a medical clinic's operations such as medical, financial, administrative operations.

The clinic management allows easy access to patients and staff members data.

The system works on desktop with easy interface used by the doctors and staff members.

The rules of doctors or administrators include adding and deleting staff members, accessing their data and patients data, providing waiting list of patients in waiting room.

The rule of staff members is adding new patients to waiting list.

System Requirements

❖ Functional Requirements

1. The clinic system must provide two main interfaces one for the doctor and another for the staff members.

2. Admin interface must provide a panel that view the balance of the day and month also provides the pure profit in month. It should also give the admin the capability to add, view data, remove staff members. The admin or doctor should have a button to enter the next patient in the waitting list.

3. Staff member interface gives the user the capability to add patients to waiting list. And in the same moment the patient name must appear in the waiting list in admin interface.

4. Clinic system should send an email to new doctors and staff members with their usernames and passwords that they will use them to access the system.

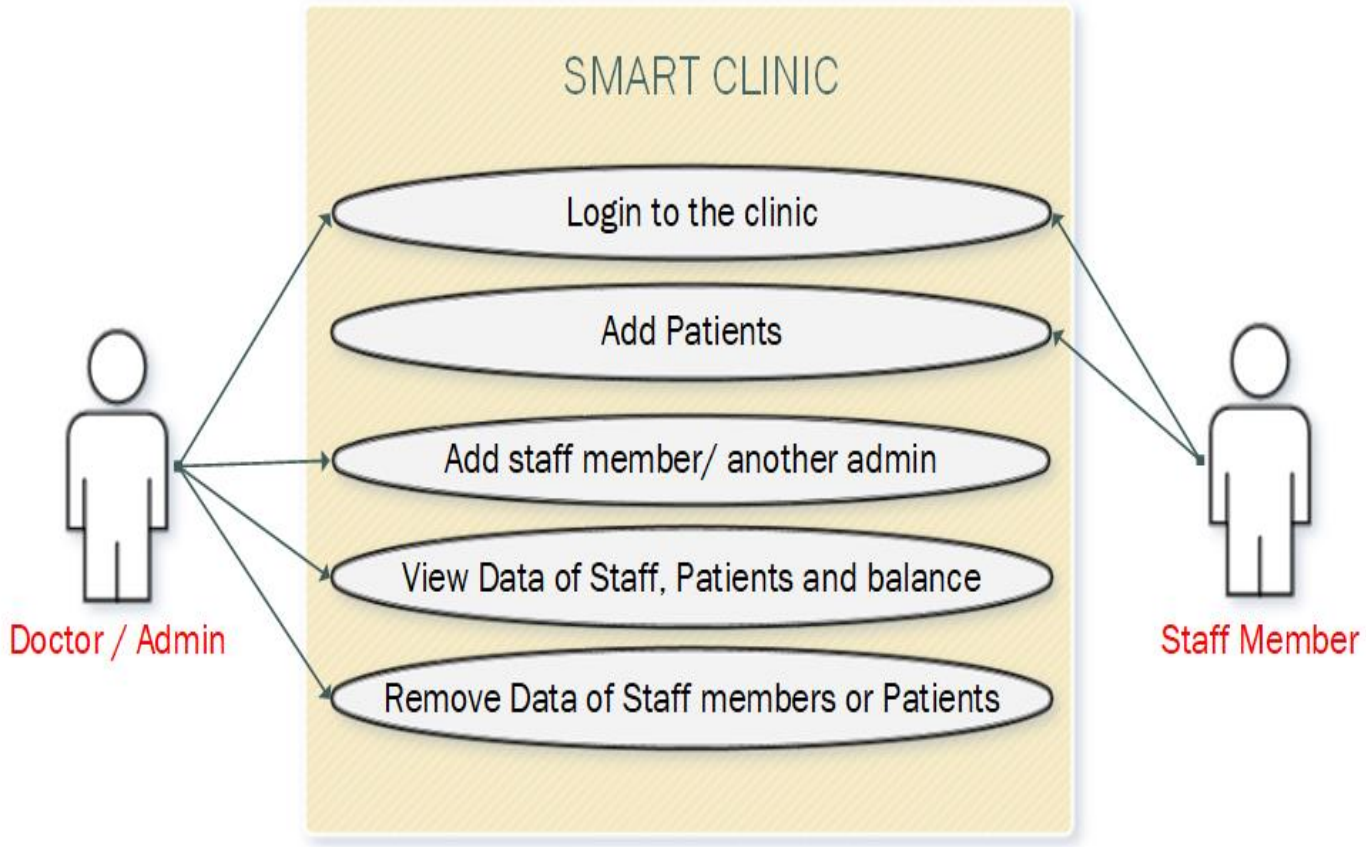
❖ Non-Functional Requirements

1. **Usability**:
   - Interfaces should be easy to use.
   - Using images and icons is necessary.
   - Buttons must contains tool-tips.
   - User training time must be within 2 days.

2. **Efficiency**:
   - Perform all the functions of the software.
   - Keep in mind:
     - Efficiency of processing.
     - Efficiency of storage.
     - Efficiency of communication.
     - Efficiency of power usage.

3. **Security**:
   - Only the admin and the staff members who can access the system using password and username.
   - The passwords and usernames must be generated randomly then sent to the user via email.
   - The user must be able to change his password and username.

Use Case Diagram



Use Case Diagram

Design Patterns

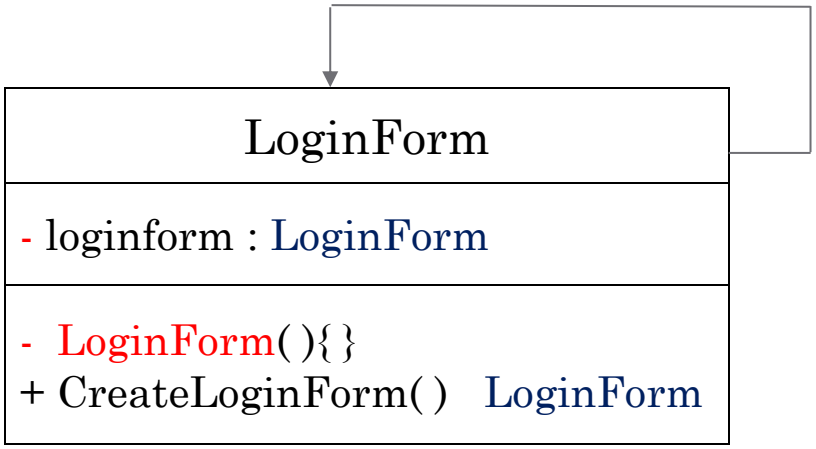
a. Singleton

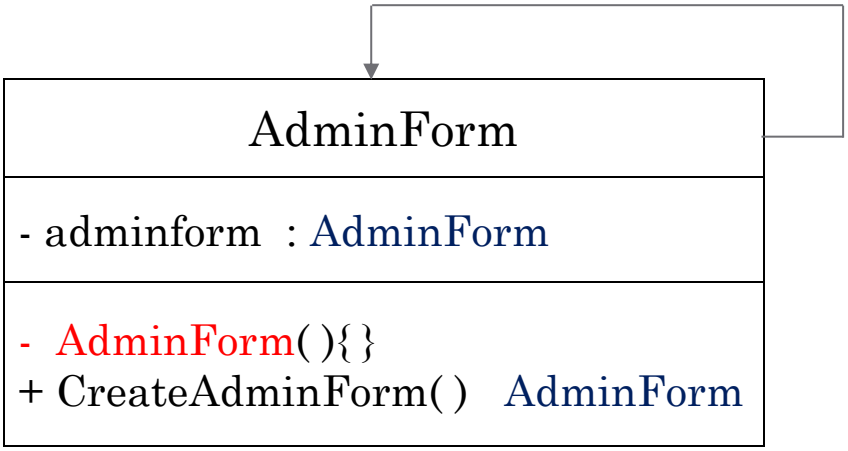
Problem:

- In windows forms the default way to move form form1 to form2 is to create a new object of form2 and show it and hide the form1. If you want to return back to the form1 is to create a new object from it. The problem is the new object of form1 not the same object that we started with. That means: foreach move from form to another we create a new object which consuming the memory.

Solution:

- To solve this problem we need to find out a way to create single object from every form and dealing with this only one. Singleton pattern will help us in this problem because the singleton pattern used when you are writing a program and you have a need for one – *and only one* – instance of a class. And you need to enforce that "and only one" requirement.





-----: and so on

In this way we will create in each form a private instance and public method to return this single instance.

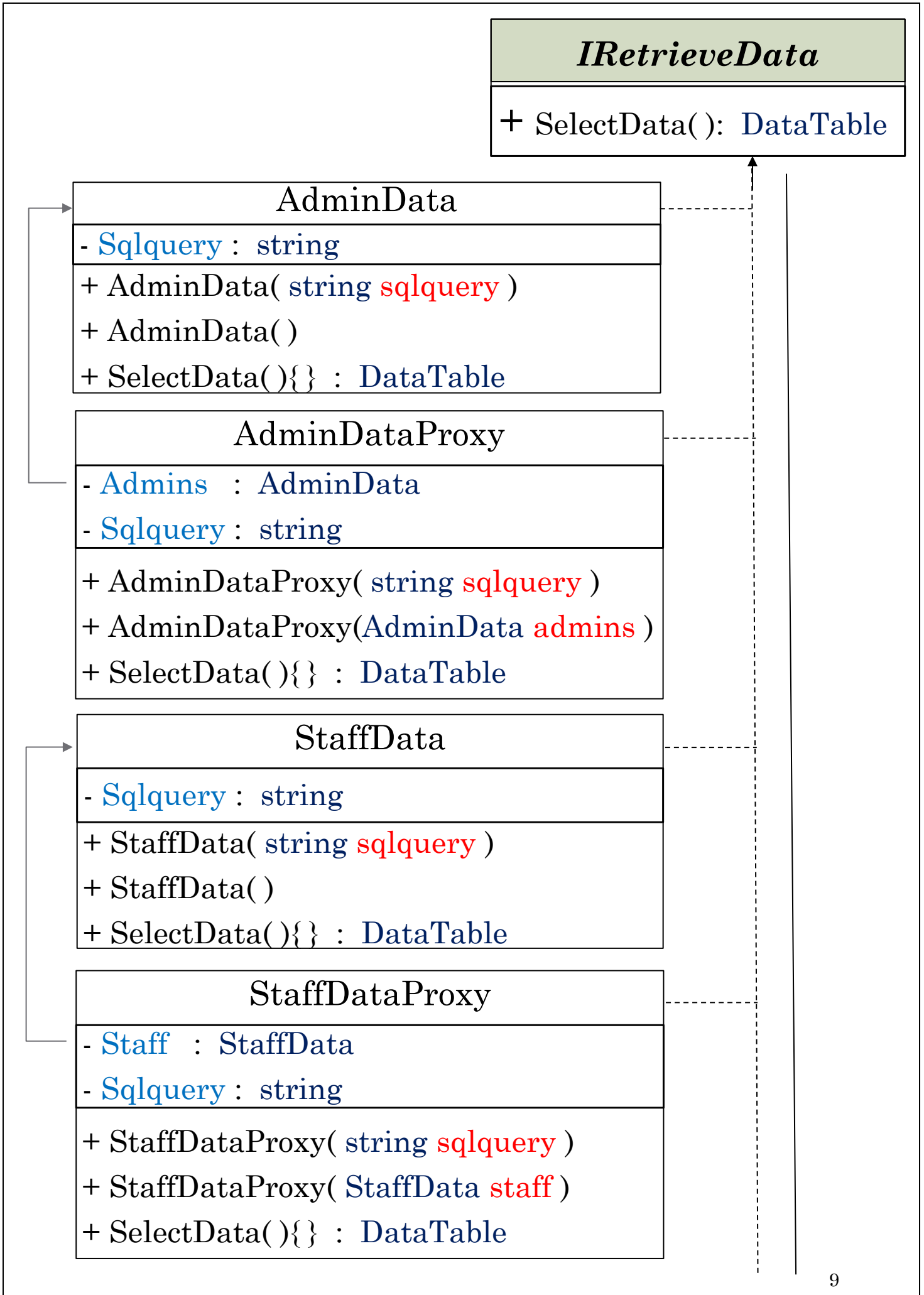
b. **Proxy**

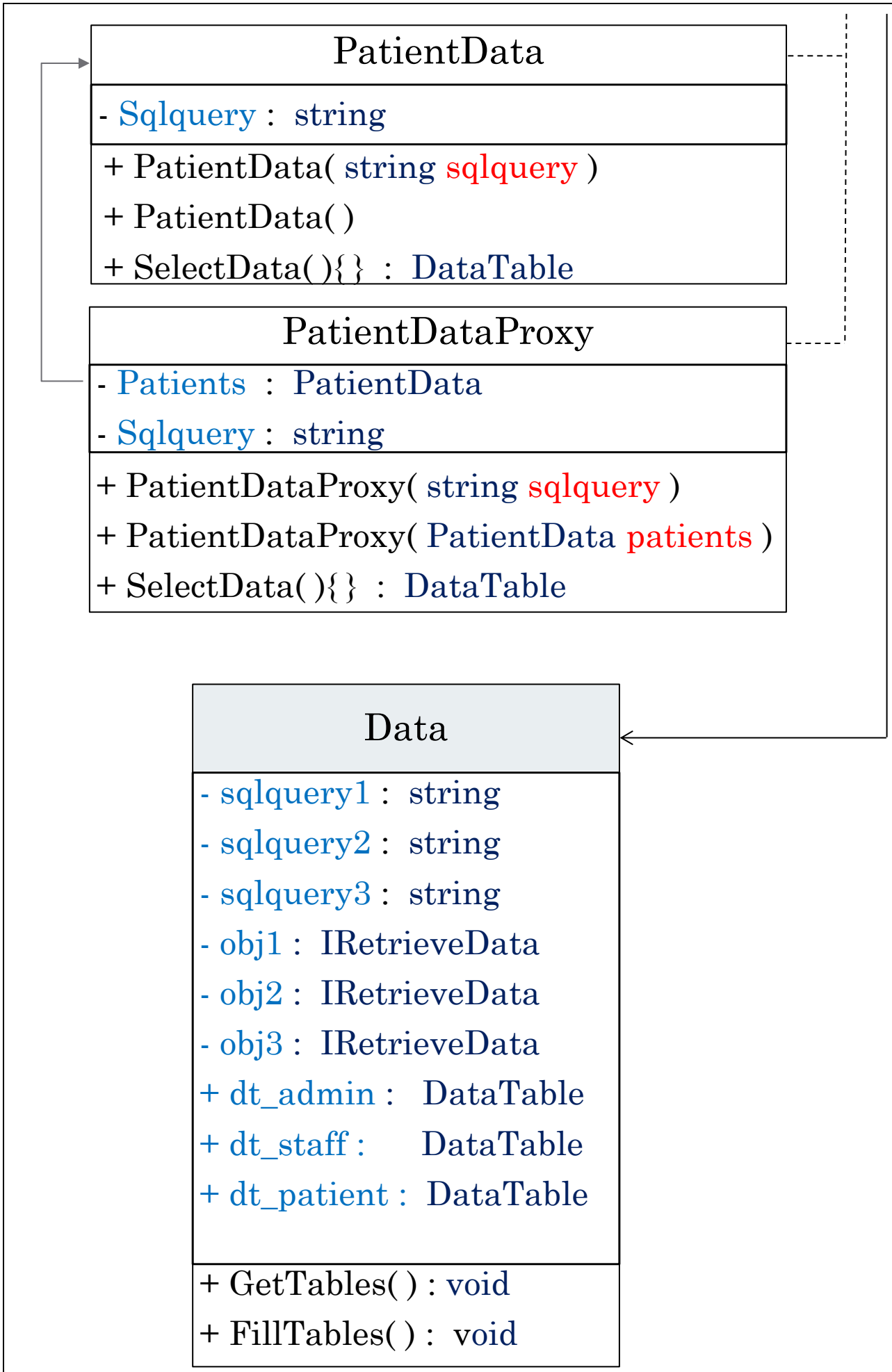
Problem:

- In clinic management system there is more than one case where dealing with database such as getting staff members data and patients data then use this data to check existing of staff member, admin and patient. So for every operation we need to connect with database and send the same queries then retrieve the same tables, this of course consumes a lot of time.

Solution:

- To solve this problem we need to find out a way to check if we retrieve data from database before or not. If not then connect to database and retrieve the data or if we already retrieve this data before then we should not connect to database but give us the same data that we retrieve before using something like the cache.
  The Proxy pattern is the suitable pattern for solving this problem.


IRetrieveData
+ SelectData( ): DataTable
AdminData
- Sqlquery : string
+ AdminData(string sqlquery)
+ AdminData( )
+ SelectData( ){ } : DataTable
AdminDataProxy
- Admins : AdminData
- Sqlquery : string
+ AdminDataProxy(string sqlquery)
+ AdminDataProxy(AdminData admins)
+ SelectData( ){ } : DataTable
StaffData
- Sqlquery : string
+ StaffData(string sqlquery)
+ StaffData( )
+ SelectData( ){ } : DataTable
StaffDataProxy
- Staff : StaffData
- Sqlquery : string
+ StaffDataProxy(string sqlquery)
+ StaffDataProxy(StaffData staff)
+ SelectData( ){ } : DataTable


PatientData
- Sqlquery : string
+ PatientData(string sqlquery)
+ PatientData( )
+ SelectData( ){ } : DataTable
PatientDataProxy
- Patients : PatientData
- Sqlquery : string
+ PatientDataProxy(string sqlquery)
+ PatientDataProxy(PatientData patients)
+ SelectData( ){ } : DataTable
Data
- sqlquery1 : string
- sqlquery2 : string
- sqlquery3 : string
- obj1 : IRetrieveData
- obj2 : IRetrieveData
- obj3 : IRetrieveData
+ dt_admin : DataTable
+ dt_staff : DataTable
+ dt_patient : DataTable
+ GetTables( ) : void
+ FillTables( ) : void

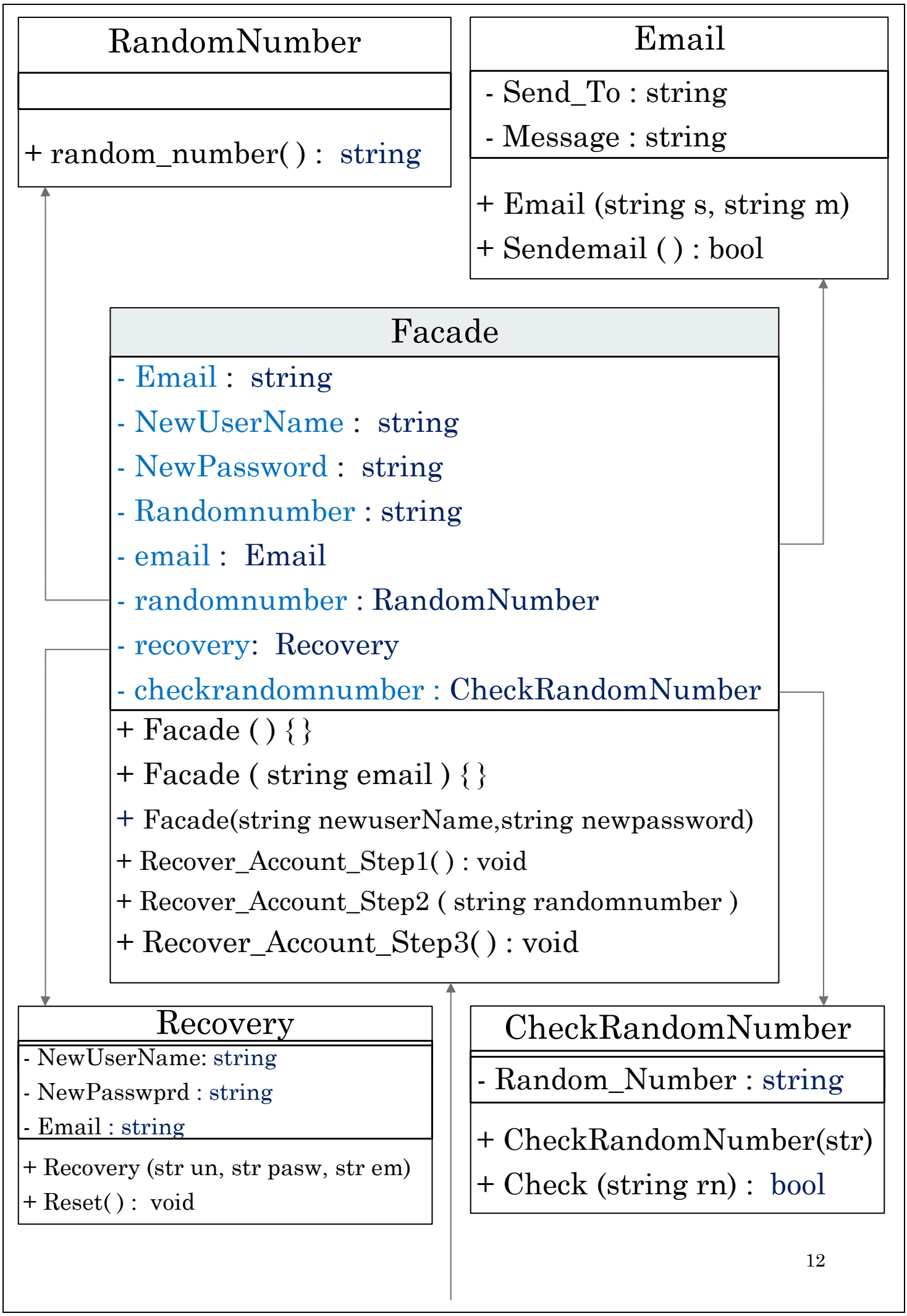
c. Façade

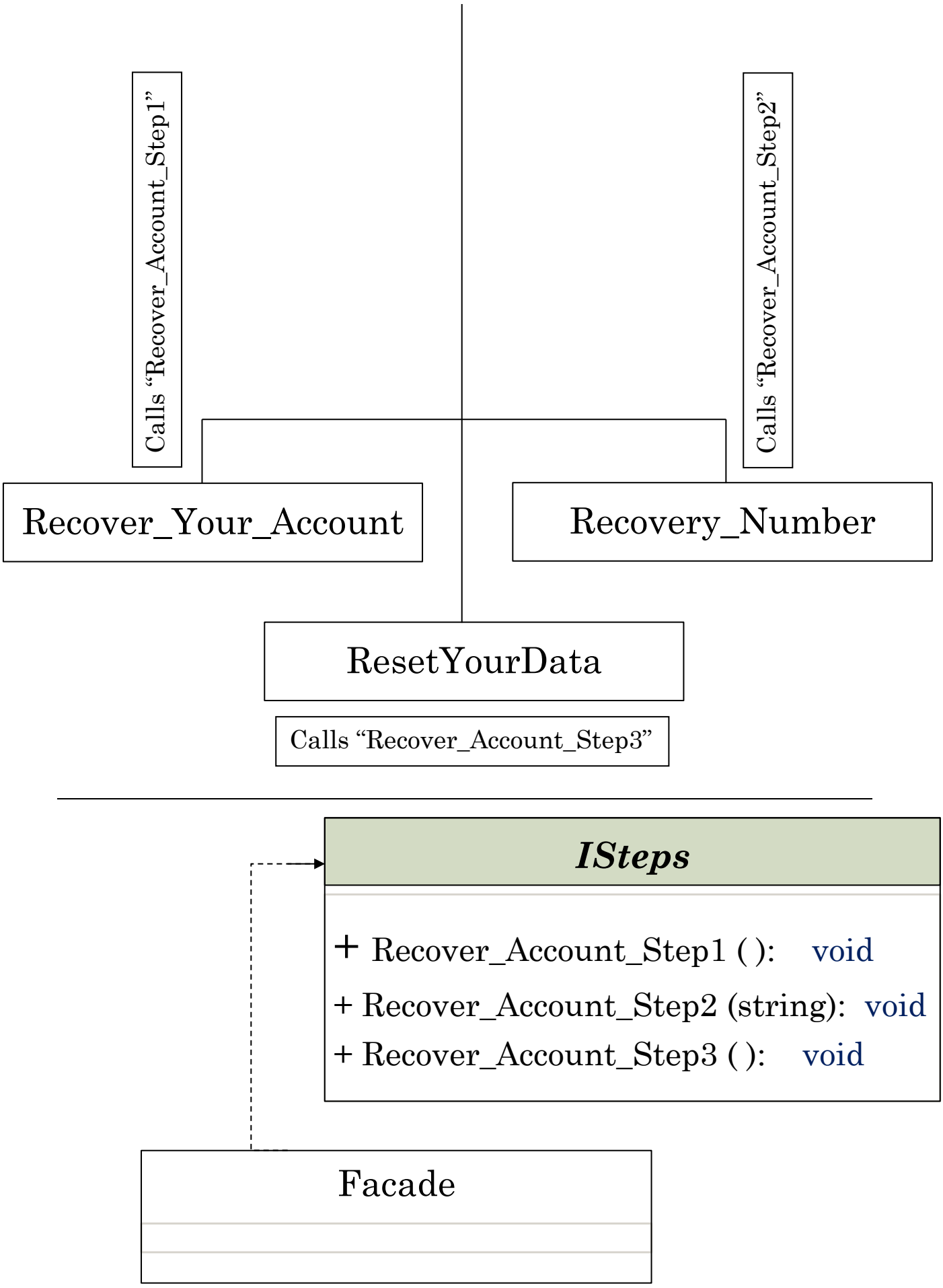
Problem:

- In clinic management system when the administrator adds a new staff member or another new admin, the new user should receive an e-mail with the password and the username, or if we forgot the password or the username and we want to recover it, there are some steps should be followed to send this email such as check email validation and create a random password and username or a random verification code and so on, the problem is we want to deal with one class, and this class manage all the steps of recovering the password.

Solution:

- The best solution for this problem is using Façade pattern.


RandomNumber
+ random_number( ) : string
Email
- Send_To : string
- Message : string
+ Email (string s, string m)
+ Sendemail ( ) : bool
Facade
- Email : string
- NewUserName : string
- NewPassword : string
- Randomnumber : string
- email : Email
- randomnumber : RandomNumber
- recovery: Recovery
- checkrandomnumber : CheckRandomNumber
+ Facade ( ) { }
+ Facade (string email) { }
+ Facade(string newuserName,string newpassword)
+ Recover_Account_Step1( ) : void
+ Recover_Account_Step2 (string randomnumber)
+ Recover_Account_Step3( ) : void
Recovery
- NewUserName: string
- NewPasswprd : string
- Email : string
+ Recovery (str un, str pasw, str em)
+ Reset( ) : void
CheckRandomNumber
- Random_Number : string
+ CheckRandomNumber(str)
+ Check (string rn) : bool


Calls "Recover_Account_Step1"
Calls "Recover_Account_Step2"
Recover_Your_Account
Recovery_Number
ResetYourData
Calls "Recover_Account_Step3"
ISteps
+ Recover_Account_Step1 ( ): void
+ Recover_Account_Step2 (string): void
+ Recover_Account_Step3 ( ): void
Facade

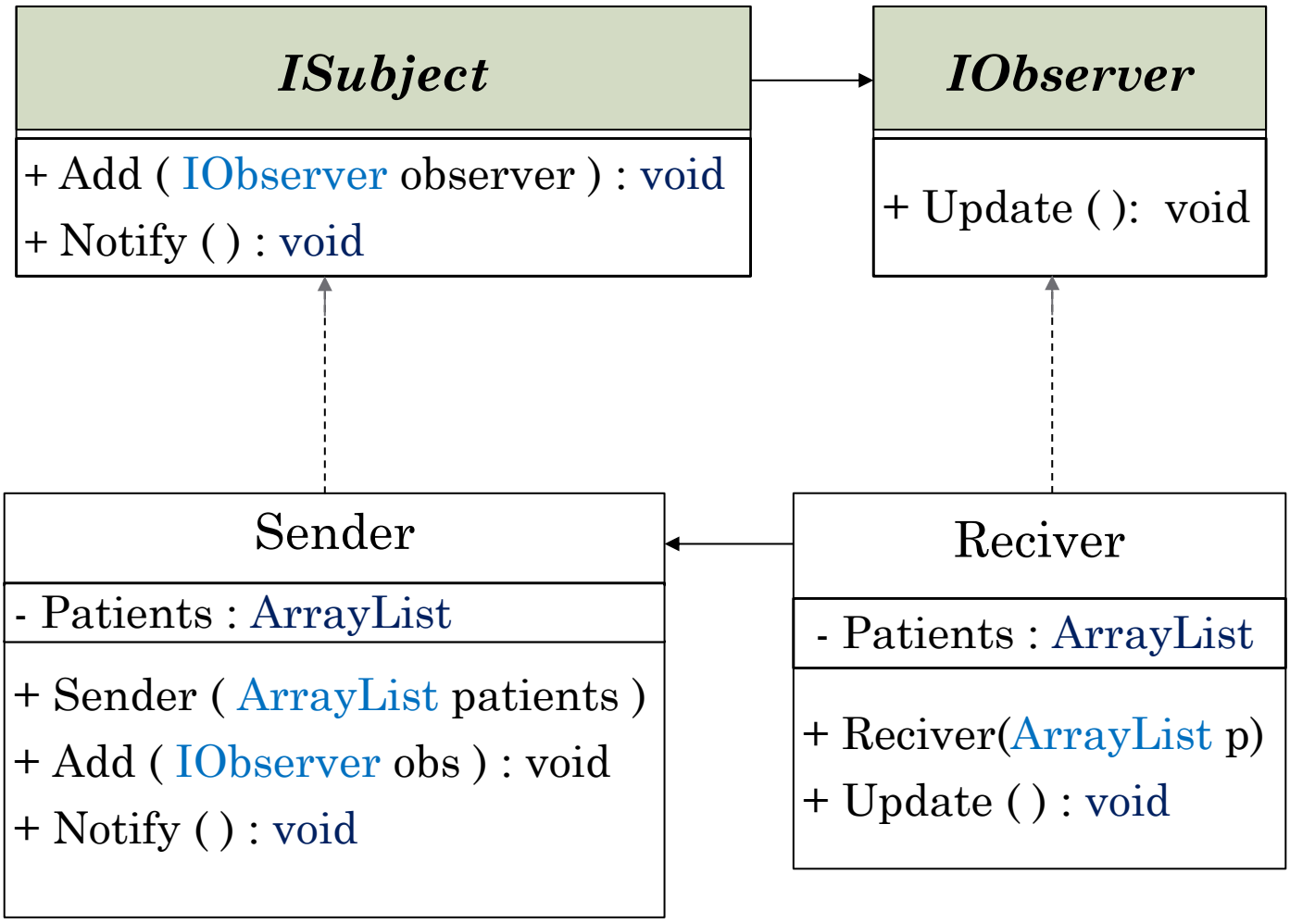
d. **Observer**

Problem:

- One of the most importance of Clinic System operations is adding a new patient to waiting list, this is the rule of stuff member, once the patient added, the waiting list in admin panel must be updated.
  So, the main problem is following changes in the clinic.

Solution:

- The best solution for this problem is using Observer pattern.


ISubject
+ Add (IObserver observer) : void
+ Notify ( ) : void
IObserver
+ Update ( ): void
Sender
- Patients : ArrayList
+ Sender (ArrayList patients)
+ Add (IObserver obs) : void
+ Notify ( ) : void
Reciver
- Patients : ArrayList
+ Reciver(ArrayList p)
+ Update ( ) : void

THANKS