SOFT252

The design of the program began with breaking apart the specification into two separate systems, the buildings with associated classes and functions and people/swipe cards and their associated classes/functions. This allowed the tasks to be broken down even further, which then lead to the design process.

Design Process

The design process for the project began with an initial, basic UML diagram to highlight needed classes. This then formed into the accessPeople package, which contains classes for each type of user and Swipecards, the Buildings package which contains all classes for every building type, aswell as room access methods.The last package to be populated was Buildings.Logging which contains classes to print text to documents.

Once the three packages had the majority of their classes and functions populated, the GUI was then designed.

accessPeople Package

To solve the problem of populating Roles to users, (as some users may have multiple roles) the PeopleFactory class was implemented. This class takes advantage of the factory pattern by using the role parameter for the current user by means of a switch statement, populating the current user swipecard with the particular role and other data. All users are a type of SwipeCard, allowing role(s) to be checked when entering a room.

Buildings Package

University, Campus and Building classes all implement IBuilding interface, this ensures each class has getName and SetName functions.

The Floor class provides a function to 'MakeRoom', as the floor object is required to associate any rooms to that floor object, this method is repeated within Building and Campus, this method is not repeated within University, as for testing purposes only one University has been populated, further development of this program would include this.

The ISubject interface is implemented by Campus, Building, Floor and Room, this provides the registerObservers and notifyObservers methods, these are used to register a new object to the and to notify any observers tiered below the current object (Which is why room does not implement ISubject).

IObservers Interface, which is implemented by Campus, Building, Floor and Room provides the update method, which is used to update the current object mode for every object registered as an Observer, for the current object type and those tiered below the current object .

The overall design of the library, utilises the following patterns:

- Singleton Pattern
    - Found within the Logger class, only allowing a single instance of that Logger to exist at a time.

- Factory Pattern

    - Used within the PeopleFactory class, to populate specific user roles.
- Observer Pattern
    - Used within Campus, Building, Floor and Room classes to change relevant mode.

JUnit Testing

The classes which include JUnit testing which has a host of tests to confirm that the classes are running correctly.

- AccessLoggerTest contains two tests to ensure that the accessLogger is populated and that the LocalTime is the current LocalTime.
- RoomTest contains 18 tests to confirm that users or specific roles can/cannot access specific rooms, some based off the current Time.
- PeopleFactory contains 10 tests to confirm that user Roles are returned correctly.

Review

- The program uses three of the specified appropriate software design patterns.
- JUnit testing is implemented.
- Follows the UML datamodel provided.
- Populates GUI log with simulated user entry.
- Populates Building mode.
- Displays user list.
- The log file is written to automatically, after every access attempt.

Current Bugs

- The GUI elements crash when another element is being used (e.g, If a user is simulated, the dropdown lists stop populating)
- Rooms are not associated correctly to the floor/building

Future development would involve quashing the current bugs, populating and polishing the GUI to include every required feature. The University class would be included within the observers, as to allow more than one university. Junit testing would be expanded to include many more variations of tests.