CSCI 5448 - Object Oriented Analysis and Design
Project Part 3 - Progress Report 1
#31 Smart Home Management System

Team:

1. Raj Kumar Subramaniam
2. Saritha Senguttuvan
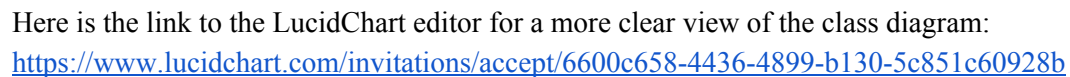3. Savitha Senguttuvan

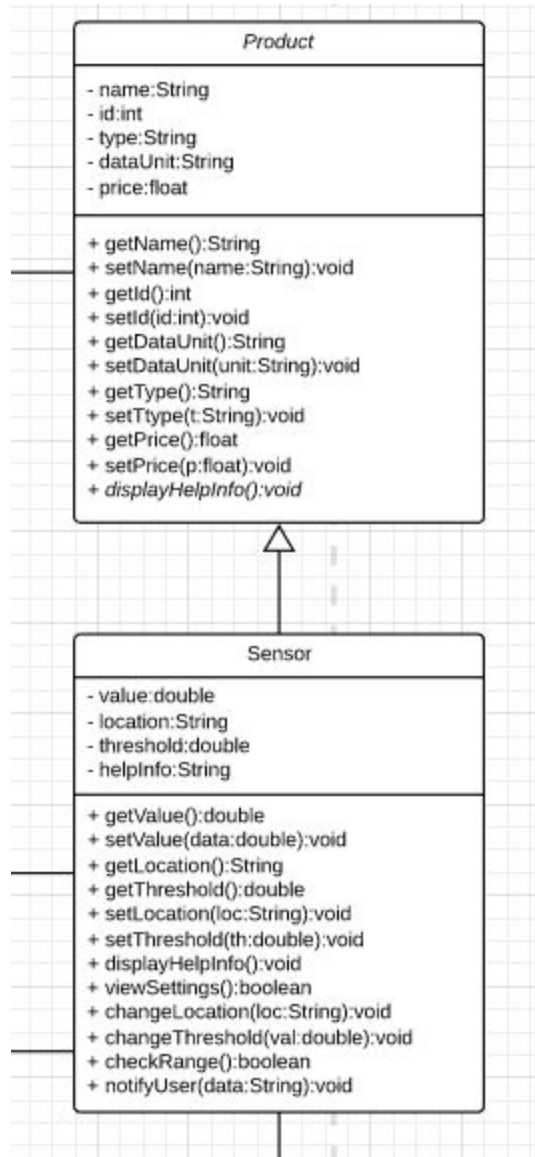**Title:**

Smart Home Management System

Description:

Smart Home Management System is a desktop app, which helps to monitor and update notifications from the smart sensors at Customer's house. The project is assumed to be for a client company, ABC Smart Home which manufactures smart home products, so the Admin will be the company's employee and the customers will be the one who purchases the products from the company. The app will help the customer and the admin to configure sensors in the home network and update any notifications to the user.
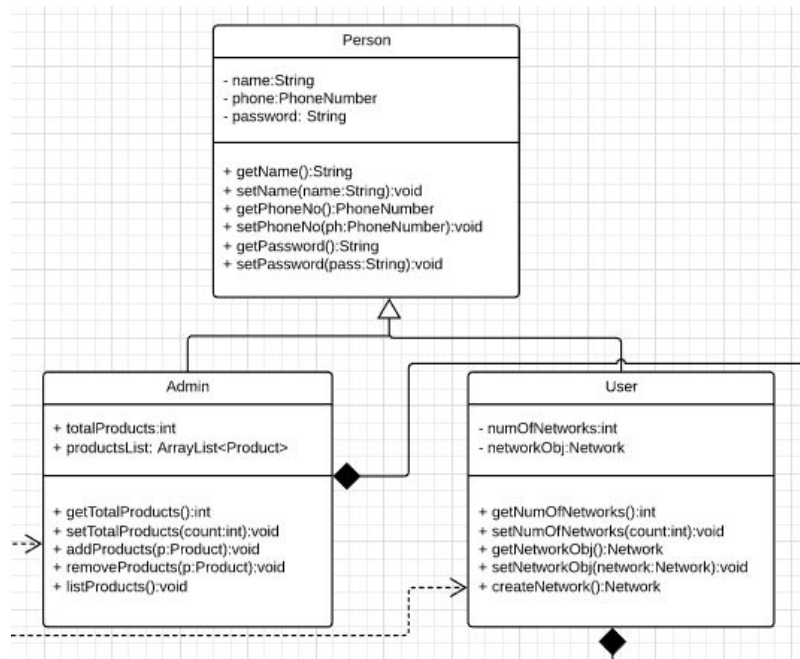
Updated Class Diagram:

We have fixed the problems we had in the class diagram based on the feedback and also optimized the class diagram to make a good use of Object Oriented Principles in the design. The fixed and detailed class diagram is given below.

ClientController

+main(args:String):void

ProfileFactory

+ createProfile(typeName:String):Person

Person

- name:String
- phone:PhoneNumber
- password: String

+ getName():String
+ setName(name:String):void
+ getPhoneNo():PhoneNumber
+ setPhoneNo(ph:PhoneNumber):void
+ getPassword():String
+ setPassword(pass:String):void

Product

- name:String
- id:int
- type:String
- dataUnit:String
- price:float

+ getName():String
+ setName(name:String):void
+ getId():int
+ setId(id:int):void
+ getDataUnit():String
+ setDataUnit(unit:String):void
+ getType():String
+ setType(t:String):void
+ getPrice():float
+ setPrice(p:float):void
+ displayHelpInfo():void

Admin

+ totalProducts:int
+ productsList: ArrayList<Product>

+ getTotalProducts():int
+ setTotalProducts(count:int):void
+ addProducts(p:Product):void
+ removeProducts(p:Product):void
+ listProducts():void

User

- numOfNetworks:int
- networkObj:Network

+ getNumOfNetworks():int
+ setNumOfNetworks(count:int):void
+ getNetworkObj():Network
+ setNetworkObj(network:Network):void
+ createNetwork():Network

Statistics

+ type:String

+ getTimeStamps: Time
+ getData(s:Sensor): double
+ applyFilter():void
+ displayGraph(type:String):void
+ viewStatistics():void

Sensor

- value:double
- location:String
- threshold:double
- helpInfo:String

+ getValue():double
+ setValue(data:double):void
+ getLocation():String
+ getThreshold():double
+ setLocation(loc:String):void
+ setThreshold(th:double):void
+ displayHelpInfo():void
+ viewSettings():boolean
+ changeLocation(loc:String):void
+ changeThreshold(val:double):void
+ checkRange():boolean
+ notifyUser(data:String):void

<<interface>>
DisplayList

+ listProducts():void

Network

+ networkName:String
+ sensorList:ArrayList<Sensor>
+ groupList:ArrayList<Group>

+ showSensorDetails(p:Sensor):void
+ addSensorToNetwork():Sensor
+ removeSensorFromNetwork(p:Sensor):boolean
+ createGroup():Group
+ deleteGroup(grp:Group):boolean
+ listProducts():void

Group

- grpName: String
- grpSpecificPower:boolean
- gsensorList:ArrayList<Sensor>

+ getGrpName():String
+ setGrpName(name:String):void
+ getGrpSpecificPwr():boolean
+ setGrpSpecificPwr(status:boolean):void
+ addSensorToGrp(s:Sensor):void
+ removeSensorFromGrp(s:Sensor):void
+ viewGroupSettings():void
+ changeGroupSettings(configPwr:boolean):void
+ listProducts():void

Here is the link to the LucidChart editor for a more clear view of the class diagram:
https://www.lucidchart.com/invitations/accept/6600c658-4436-4899-b130-5c851c60928b

Completed Class Diagram:
We have worked on stub creations and few implementations of few classes. We have already covered the detailed implementation of the class attributes and functions in the previous class diagram. Here, we are highlighting the implementations of few classes which are Product, Sensor, Person, Admin and User. The relationship with other classes can be found in the earlier class diagram. We have made all the association relationship to be more specific by choosing one of composition or aggregation relationships. We have also added factory design pattern to accomplish the tasks related to adding design patterns in the design.

Product

- name:String
- id:int
- type:String
- dataUnit:String
- price:float

+ getName():String
+ setName(name:String):void
+ getId():int
+ setId(id:int):void
+ getDataUnit():String
+ setDataUnit(unit:String):void
+ getType():String
+ setTtype(t:String):void
+ getPrice():float
+ setPrice(p:float):void
+ *displayHelpInfo():void*

Sensor

- value:double
- location:String
- threshold:double
- helpInfo:String

+ getValue():double
+ setValue(data:double):void
+ getLocation():String
+ getThreshold():double
+ setLocation(loc:String):void
+ setThreshold(th:double):void
+ displayHelpInfo():void
+ viewSettings():boolean
+ changeLocation(loc:String):void
+ changeThreshold(val:double):void
+ checkRange():boolean
+ notifyUser(data:String):void

We have partially implemented the above classes and have been creating stubs for the other classes as well.

Summary:

We have been working on creating stubs for all the classes and have restructured few class diagrams so that we should be able to achieve the required number of functionalities implemented before the due date of the project. We have implemented the Person, Admin and User, Product and Sensor classes. We have also thought of restructuring based on design patterns and added factory design pattern to create profile and also have plans of adding the iterator and observer patterns in the next iteration.

Breakdown of work:

All of us were working on the class diagram restructuring with frequent meetings and updated the class diagram for the fixes and new changes.

Raj Kumar Subramaniam - was working on implementing the Sensor and Product class and is in the process of adding more functions related to the Sensor and Product classes

Saritha Senguttuvan - was working on adding functionalities related to Person, Admin and User classes.

Savitha Senguttuvan - was working on implementing the Network and Group classes to implement the functionalities related to adding sensors to the network and adding new groups of sensors.

Github Graph:

The github graph had an issue that it wasn't loading all the commits though all the team members have contributed for the project. So, we have added the recent commits as well as a proof of the issue for grading purposes. The github graph* for all the users are shown below along with the commits made,

Testing changes
Saritha Senguttuvan committed 41 seconds ago

New changes
SarithaSenguttuvan committed 9 minutes ago

Dummy change for graph issue
SarithaSenguttuvan committed 16 minutes ago

Adding of Profile creation related classes
SarithaSenguttuvan committed 35 minutes ago

Adding Product and Sensor classes
rajkumarsubramaniam committed 43 minutes ago

ClientController, Group, Network Code
twinsavitha committed an hour ago

twinsavitha #1
3 commits 107 ++ 0 --

rajkumarsubramaniam #2
2 commits 125 ++ 0 --

SarithaSenguttuvan #3
1 commit 2 ++ 0 --
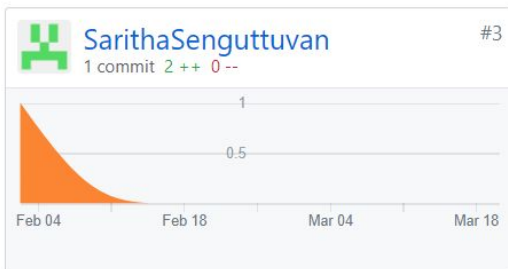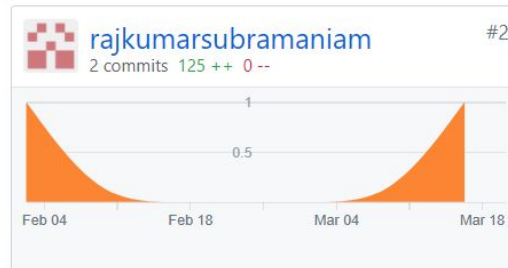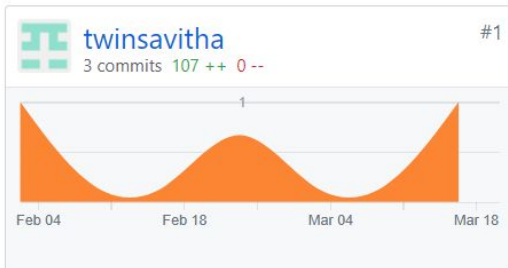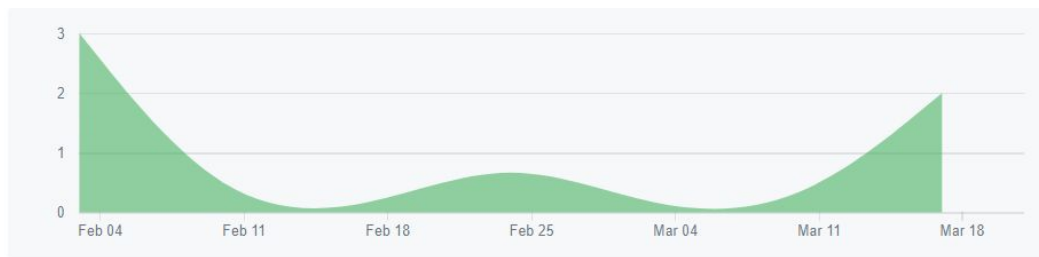
Saritha Senguttuvan has committed but there is some issue which doesn't update the Github graph.

Estimated Remaining Effort:

Now, the class diagram looks like a better version of the first iteration. As we progress we were able to add more specific relationship between the classes such as Aggregation and Composition. As we progress in the code, we have few ideas of design patterns that we can implement. We have implemented Factory design pattern to create profiles for admin and customer, we also have options of adding iterator and observer pattern to iterate through the sensor list and observer patterns for implementing notifications when the sensor value changes relative to threshold value settings. To deliver the project before the due date, we have to work around 10 hours per week per person and that is our plan to deliver the software in the coming weeks.

Next Iteration:

In the next iteration we are planning to implement the Statistics class and also one or more design patterns. We will be simultaneously working on other class implementations and will be planning for a good UI design. So, in the next iteration we will be adding design patterns and more functionalities on existing classes in the class diagram.