House Mate Controller Design Document

Date: 10/18/2015
Author: Yingtian Wang
Reviewers:

Introduction

This document provides the design for the House Mate Controller Service. House Mate Controller is an important component of House Mate System. It is the brain that adds intelligence to the House Mate System.

Overview

This document recaps the requirements first, then the use case diagram shows the idea of how to use this program. The implementation part of the document describe how the implementation meets the requirement, it presents the class diagram to how the control of the system implemented, following with class dictionary, it describe the property, association, operation of the class in detail. In addition, the sequence diagram shows how a command from the the command line interface be processed by this programs.

Requirements

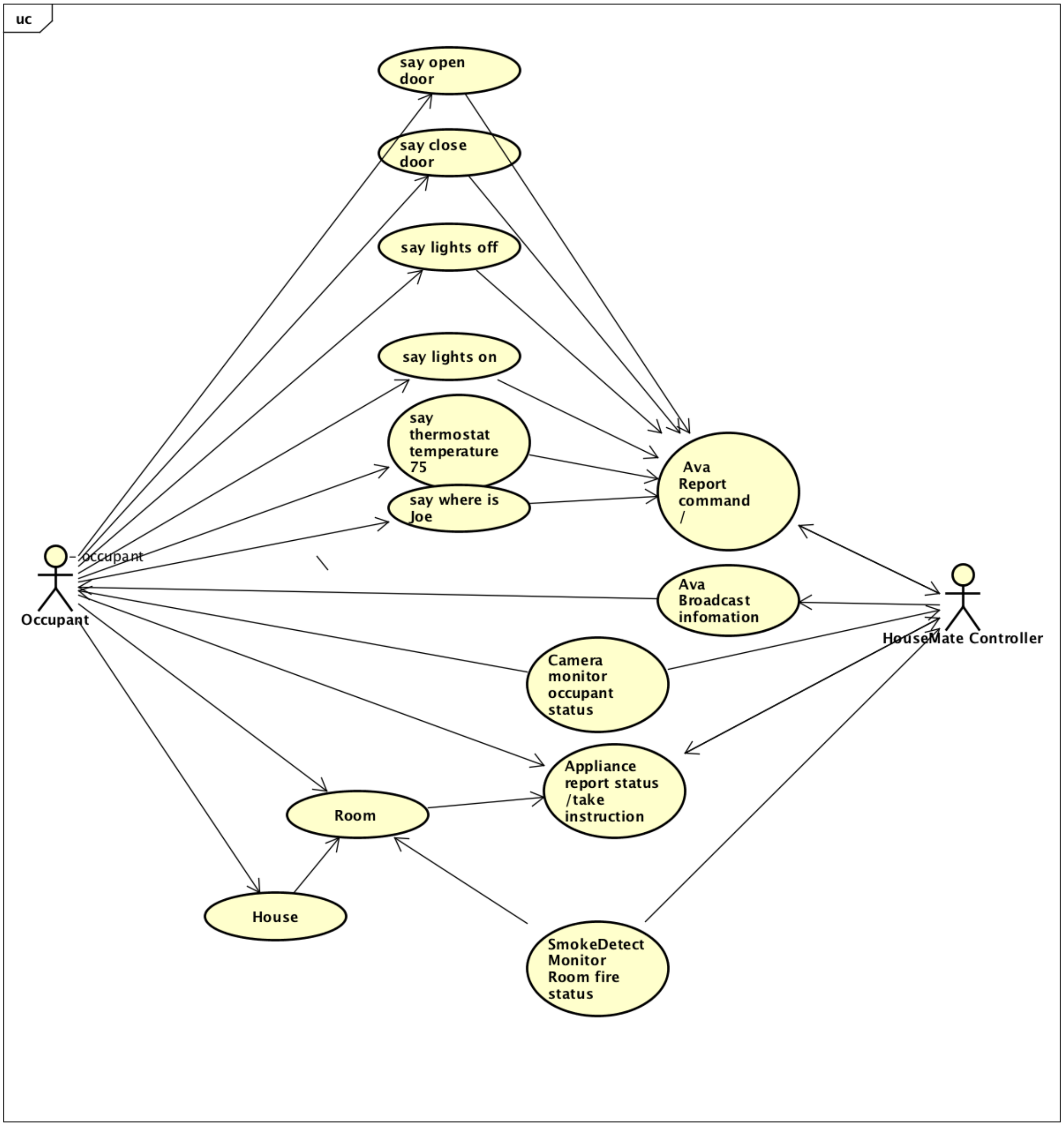
This section defines the requirements for the House Mate Controller.
The House Mate Controller Service use the interface of the House Mate Model Service to monitor the status the IOT devices.

1. This design use an observer pattern to monitor the IOT devices, the abstract class IOTDevices extends java.util.observable class, so every sensor and appliance can be observed.
2. The controller implements a ModelObserverInterface to be notified when devices status changes, and take actions based on status change.
3. Command pattern is used to implement the interaction between the Model Service and the Controller Service.
4. The knowledge graph is used to keep track of the location and status of the occupants.

Use Cases

The following use case diagram shows the use case supported by the House Mate Controller System.

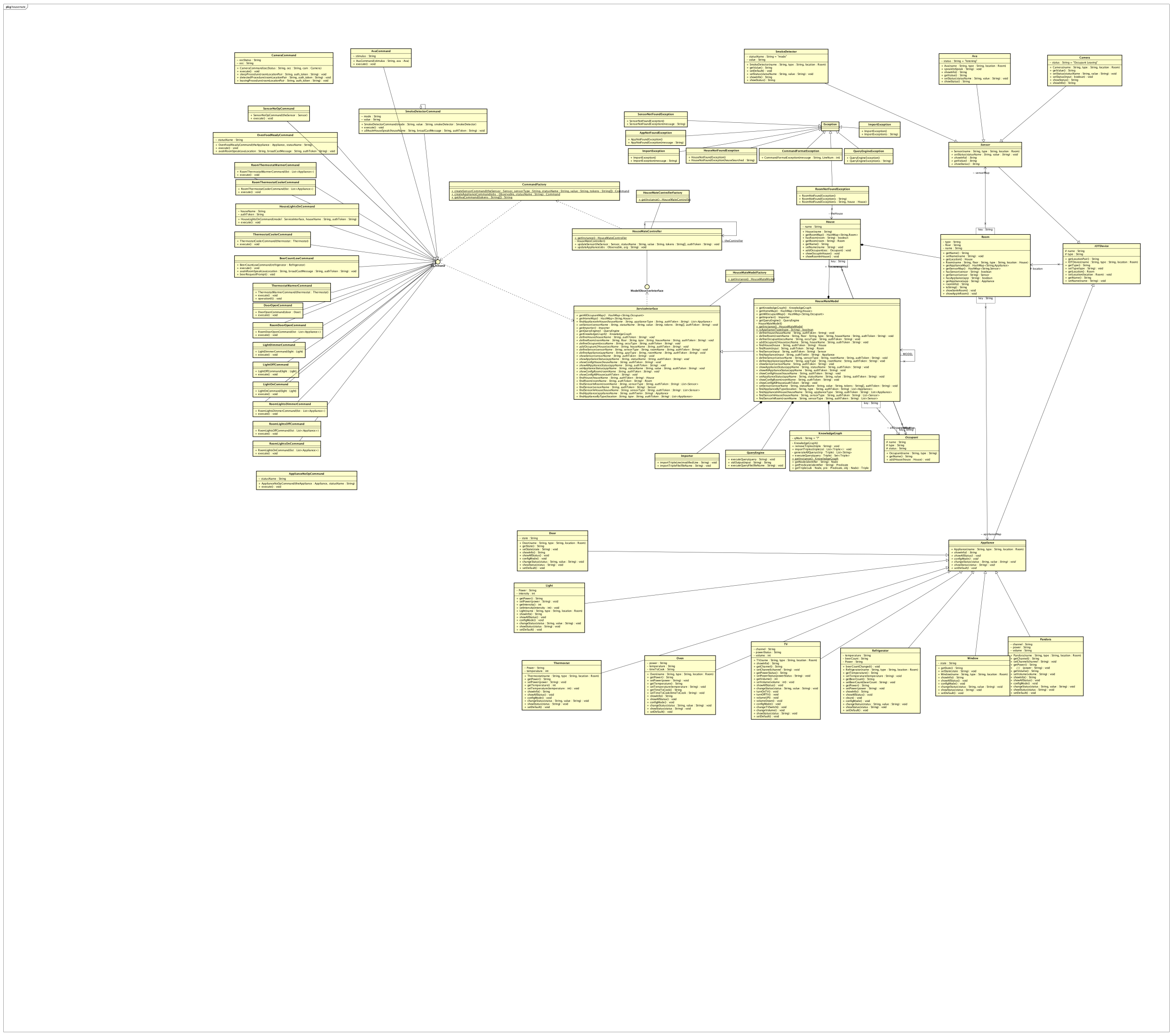
The Occupant can say commands to Ava device in the room. Ava device report the command to the controller, and then controller take actions. The controller also uses Ava to broadcast information to the occupant when necessary. The camera and smoke detector also monitor occupant behavior or room status and report to controller.


uc
say open door
say close door
say lights off
say lights on
say thermostat temperature 75
say where is Joe
Ava Report command /
occupant
Occupant
Ava Broadcast infomation
HouseMate Controller
Camera monitor occupant status
Appliance report status /take instruction
Room
House
SmokeDetect Monitor Room fire status

Implementation

Class Diagram

Please find “HMCS_Class Diagram.png“   in /hw4/docs folder for more clear view of the class diagram.



Class Dictionary

This section specifies the class dictionary for the HouseMateController.

House Mate Controller

Method Name	**Signature**	**Description**
getInstance	HouseMateController : void	HouseMateController is a singleton, this method Return the instance of itself
updateSensor	(theSensor: Sensor, statusName : String, value : String, tokens : String[], authToken : String) : void	House Mate Controller is an observer implements ModelObserverInterface, this is to update a sensor settings when it observed changes.
updateAppliance	(obs : Observable, arg : String) : void	House Mate Controller is an observer implements ModelObserverInterface, this is to update a appliance settings and take actions based on the trigger when it observed changes.

Associations

Association Name	**Type**	**Description**
HouseMateControllerFactory	Class	A smiple factory to create a HouseMateController instance.
CommandFacotry	Class	A factory to create command based on input. It also parses the Ava voice input used as Ava status value

House Mate Controller Factory

Based on the Architecture Document, this class is to provide a factory for accessing the HouseMateController singleton instance.

Methods

Method Name	Signature	Description
getInstance	HouseMateController	Return the HouseMateController singleton instance

Command Factory

A factory to create command based on input. It also parses the Ava voice input used as Ava status value.

Method Name	Signature	Description
createSensorCommand	(theSensor : Sensor, sensorType : String, statusName : String, value : String, tokens : String[]) : Command	This method generate a type of sensor command and return the command
createApplianceCommand	(obs : Observable, statusName : String) : Command	This method generate a type of appliance command and return the command
+ getAvaCommand(tokens : String[]) : String	(tokens : String[]) : String	This method use regular expression to get the ava command input like 'open door'

ModelObserverInterface

This is an interface for HouseMateController to observe the modle service status change.

Method Name	Signature	Description
updateSensor	(theSensor: Sensor, statusName : String, value : String, tokens : String[], authToken : String) : void	this is to update a sensor settings when it observed changes.
updateAppliance	(obs : Observable, arg : String) : void	this is to update a appliance settings and take actions based on the trigger when it observed changes.

House Mate Model

The HouseMateModel class is the implementation of Service Interface. It provides methods for House Mate Controller take action on the sensors and appliance. HouseMateModel class is a singleton class which means there is only one HouseMateModel instance. It also has two hash map to track all the houses and occupants directly.

Note: this Class dictionary for HouseMateModel Class only shows the method added for assignment 3, for methods used in assignments 2 please refer back to House Mate Model Design Document.

Methods

Method Name	**Signature**	**Description**
setApplianceStatus	(appName : String, statusName : String, value : String, authToken : String) : void	Set appliance Status based on appliance name
setSensor	(sensorName : String, statusName : String, value : String, tokens : String[], authToken : String) : void	Set Sensor Status based on sensor name
findSensorInRoom	(roomName : String, sensorType : String, authToken : String) : List<Sensor>	This method find a type of sensor in a room based on the room name and the type of sensor
findSensorInHouse	(houseName : String, sensorType : String, authToken : String) : List<Sensor>	This method find a type of sensor in a house based on the house name and the type of sensor
findApplianceByType	(location : String, type : String, authToken : String) : List<Appliance>	This method find a type of appliance based on the room name and the type of appliance
getKnowledgeGraph	void : KnowledgeGraph	This method return the KG inside the HouseMateModel
findApplianceInHouse	(houseName : String, applianceType : String, authToken : String) : List<Appliance>	This method find a type of appliance based on the house name location and the type of appliance
getImporter	void : Importer	This method return the importer for KG inside the HouseMateModel
getQueryEngine	Void : QueryEngine	This method return the QueryEngine for KG inside the HouseMateModel

Command

Command is an interface that has an execute method.
There are different types of command classes implements this interface. Those instances of command classes are created by command factory.

AvaCommand

AvaCommand is executed when ava status changes.

Methods

Method Name	Signature	Description
execute	void: void	Execute ava command input

CameraCommand

CameraCommand is executed when camera status changes.

Methods

Method Name	Signature	Description
execute	void: void	Execute camera command input
sleepProcedure	void: String roomLocationPair, String authToken	Actions when occupant is asleep
detectedProcedure	void: String roomLocationPair, String authToken	Actions when occupant enters the room
leavingProcedure	void: String roomLocationPair, String authToken	Actions when occupant leaves the room

SmokeDetectorCommand

SmokeDetectorCommand is executed when smokeDetector is status is fire, execute sensorNoOpCommand if the status is ok.

Methods

Method Name	Signature	Description
execute	void: void	Execute smokeDetector command input
allAvaInHouseSpeak	Void: String houseName String broadCastMessage String authToken	Ava broadcast fire message in all rooms

BeerCountLowCommand

BeerCountLowCommand is executed when fridge status beer count is low, execute AppNoOpCommand if the status is not low.

Methods

Method Name	Signature	Description
execute	void: void	Execute BeerCountLowCommand
avaInroomSpeak	Void: String avaLocation String broadCastMessage String authToken	use the ava in the room that same as the fridge to speak
beerRequestPrompt	Void:void	Ask user if he want more beer

OvenFoodReadyCommand

OvenFoodReadyCommand is executed when oven status time to cook is zero, execute AppNoOpCommand if the status is not zero.

Methods

Method Name	Signature	Description
execute	void: void	Execute OvenFoodReadyCommand
avaInroomSpeak	Void: String avaLocation String broadCastMessage String authToken	use the ava in the room to notify the user

SensorNoOpCommand

SensorNoOpCommand is executed when sensor status changes but no actions should be taken.

Methods

Method Name	Signature	Description
execute	void: void	Show the current status of the sensor

ApplianceNoOpCommand

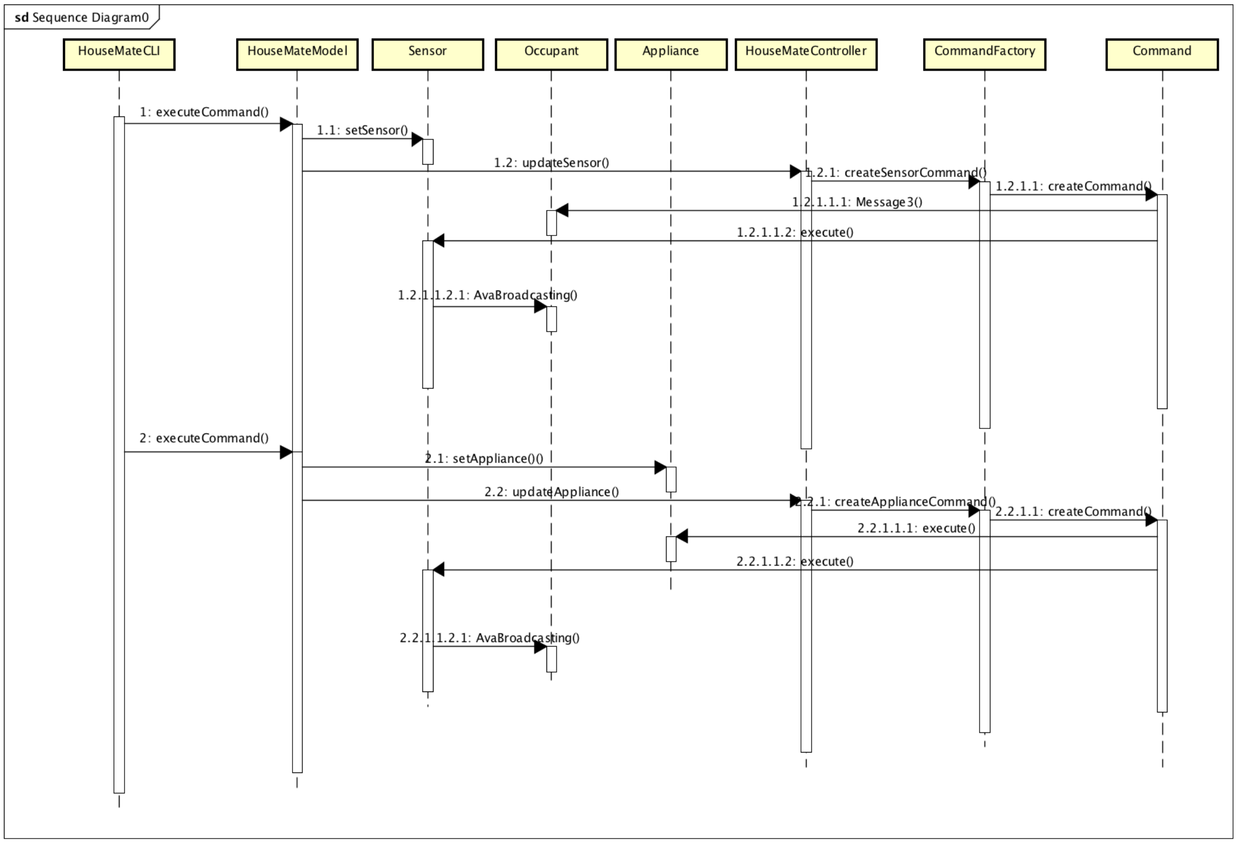
ApplianceNoOpCommand is executed when appliance status changes but no actions should be taken.

Methods

Method Name	Signature	Description
execute	void: void	Show the current status of the appliance

Sequence Diagram

When the command imported to the HouseMate Command interface, it calls executeCommand method, if it is a setSensor or setAppliance method, House Mate Model change the status of the sensor or appliance, if the changes is successful, it calls the updateSensor or updateAppliance method to invoke the rules by the HouseMateController, the controller use the commandFacotry to generate the right rule command and execute it. Please see the sequence diagram below.



Implementation Details

This House Mate Model Controller use a Command factory to generate the right rule command for different type of status change and execute it. Thus the rule commands are well exculpated and can be easily extended if a new rule need to be applied.

Testing

Testing needs to be thorough because of the complexity of the system. Many objects need to be initiated in order to test the how it works. Test case should be implement to simulate many scenarios. There are a few example scenarios of test case to implment.

Scenario 1: camera detect joe_smith, but it is not defined before(Unknow person detected)

Output should be :   Unknown person joe_smith detected

Scenario 2: camera in kitchen1 detect joe_smith, then Rover in dining_room1 ask Ava where joe is

input command: set sensor house1:kitchen1:camera1 status OCCUPANT_DETECTED value joe_smith

input command: set sensor house1:dining_room1:ava2 status LISTENING value "rover says: 'where is joe_smith'"

Output should be:

input query:

joe_smith is_in ?.

output:

Joe_smith is_in House1:kitchen1.

Scenario 3: camera in kitchen1 detect joe_smith, then it detect joe left kitchen1 entered dining_room1,

then Rover in dining_room1 ask Ava where joe is.

(This shows the KG can updated the loacation of joe)

input command: set sensor house1:kitchen1:camera1 status OCCUPANT_DETECTED value joe_smith

input command: set sensor house1:kitchen1:camera1 status OCCUPANT_LEAVING value joe_smith

input command: set sensor house1:dining_room1:camera1 status OCCUPANT_DETECTED value joe_smith

input command: set sensor house1:dining_room1:ava2 status LISTENING value "rover says: 'where is joe_smith'"

```
Output:
input query:
joe_smith is_in ?.
output:
Joe_smith is_in House1:dining_room1.
```

Scenario 4: joe in dining_room1 ask ava2 to do things.

```
input command: set sensor house1:dining_room1:ava2 status LISTENING value "joe_smith
says: 'lights on'"
*********************************************
light1 in house1:dining_room1 is on
light2 in house1:dining_room1 is on
******************************************

input command: set sensor house1:dining_room1:ava2 status LISTENING value "joe_smith
says: 'open door'"
*********************************************
door2 in house1:dining_room1 is open
door1 in house1:dining_room1 is open
******************************************

input command: set sensor house1:dining_room1:ava2 status LISTENING value "joe_smith
says: 'thermostat temperature 70'"
*********************************************
The temperature of the thermostat is now 70
```

Scenario 5: rover ask ava joe's status when joe is active

```
input command: set sensor house1:dining_room1:ava2 status LISTENING value "rover says:
'joe_smith is ?'"

input query:
joe_smith is ?.
output:
Joe_smith is Active.
```

Scenario 6: Light and thermostat status change depends on occupant status
(default temperature of thermostat is 75,
change by 5 when call "cooler" or "warmer"
intensity of light is 50,
change by 5 when call "dimmer" or "brighter")

when there are no thermostat or light in the room (not defined before), report the info.

```
input command: set sensor house1:kitchen1:camera1 status OCCUPANT_LEAVING value
joe_smith
*********************************************
joe_smith left house1:kitchen1
no lights in this room
no thermostat in this room
```

when there are light or thermostat defined before

```
input command: set sensor house1:kitchen1:camera1 status OCCUPANT_DETECTED value
joe_smith
*********************************************
joe_smith entered house1:kitchen1
ligth3 in house1:kitchen1 is ON
light4 in house1:kitchen1 is ON
The temperature of the thermostat3 is now 80
******************************************
```

29.

```
input command: set sensor house1:kitchen1:camera1 status OCCUPANT_LEAVING value
joe_smith
*********************************************
joe_smith left house1:kitchen1
ligth3 in house1:kitchen1 is OFF
light4 in house1:kitchen1 is OFF
The temperature of the thermostat3 is now 75
******************************************
```

30.

```
input command: set sensor house1:dining_room1:camera1 status OCCUPANT_DETECTED
value joe_smith
*********************************************
joe_smith entered house1:dining_room1
light1 in house1:dining_room1 is ON
light2 in house1:dining_room1 is ON
The temperature of the thermostat1 is now 80
******************************************
```

31.

```
input command: set sensor house1:dining_room1:camera1 status OCCUPANT_INACTIVE
value joe_smith
*********************************************
joe_smith is sleeping
The intensity of the light is now 40
The intensity of the light is now 40
```

```
******************************************
```

Scenario 7 : KG updates when occupant status change

```
input command: set sensor house1:dining_room1:camera1 status OCCUPANT_INACTIVE
value joe_smith
*********************************************
joe_smith is sleeping
The intensity of the light is now 40
The intensity of the light is now 40
******************************************
37.
input command: set sensor house1:dining_room1:ava2 status LISTENING value "rover says:
'joe_smith is ?'"
*********************************************
************************
input query:
joe_smith is ?.
output:
Joe_smith is Sleeping.
******************************************
38.
input command: set sensor house1:dining_room1:camera1 status OCCUPANT_ACTIVE value
joe_smith
*********************************************
joe_smith is active
******************************************
39.
input command: set sensor house1:dining_room1:ava2 status LISTENING value "rover says:
'joe_smith is ?'"
*********************************************
************************
input query:
joe_smith is ?.
output:
Joe_smith is Active.
```

Scenario 8: Smoke detector is set to Fire

To test, we define 2 lights in dining_room1, 2lights in kitchen1
and ava1 in dining room1, ava 2 in kitchen1;

```
input command: set sensor house1:kitchen1:smoke_detector1 status mode value FIRE
********************************************
FIRE!! Starting evacuation procedure
light1 in house1:dining_room1 is on
light2 in house1:dining_room1 is on
light4 in house1:kitchen1 is on
light3 in house1:kitchen1 is on
(ava2 in house1:dining_room1 is broadcasting)--Fire in kitchen1 , Please leave house1
immediately
(ava1 in house1:kitchen1 is broadcasting)--Fire in kitchen1 , Please leave house1 immediately
Calling 911
```

Scenario 9: 1. turn oven on, use a show oven power commmand to show it is on
2. Time to coock goes 0, ava2 in that room report the message.
3. use show method again to show the oven is turned off automaticlly.

```
1. input command: show appliance house1:dining_room1:oven1 status power
********************************************
The power status of the oven is now on
*****************************************
2. input command: set appliance house1:dining_room1:oven1 status timetocook value 0
********************************************
Time to cook of the oven is now 0
timetocook has changed
(ava2 in house1:dining_room1 is broadcasting)--Food is Ready
*****************************************
3.
input command: show appliance house1:dining_room1:oven1 status power
********************************************
The power status of the oven is now off
```

Scenario 10:   beer count is set 3, Ava in that room ask a question

result:

```
input command: set appliance house1:dining_room1:refrigerator1 status beercount value 3
*********************************************
Beer count is now 3
beercount has changed
(ava2 in house1:dining_room1 is broadcasting)--Would you like more beer?
Enter yes or no
yes
Order email has been sent
```

```
input command: set appliance house1:dining_room1:refrigerator1 status beercount value 3
*********************************************
Beer count is now 3
beercount has changed
(ava2 in house1:dining_room1 is broadcasting)--Would you like more beer?
Enter yes or no
no
Ok, no beer for you :(
```

Risks

There are many small command classes. This is one of the disadvantage of command pattern. For example, there are LightDimmerCommand LightBrighterCommand LIghtOnCommand LightOffCommand just for to control different actions of light. Also,Some command uses another command to finish a task. For example, CameraCommand execute lightOffCommand in the command. So a good way to organize and package different commands class and understand the relationships is important for the scalability of the system.