House Mate Model Service Design Document

NOTE: Text in italics should be replaced with your own content.

Date: 10/7/2015
Author: Gerald Trotman
Reviewer(s): Takayuki Lido

Introduction

This document defines the design for the House Mate Model Service API, which is a central component of the House Mate Model Service.

Overview

The overarching issue that is being solved is managing the various appliances/sensors and their corresponding states. Both are able to collect and share data, but unlike sensors, appliances can also be controlled.

Specifically, the House Mate Model Service solves the problem of being able to have access to all sensors and appliances that other occupants such as children and pets may not for example or monitor and things like location and status that even Adults can not.

Consider adding a diagram that explains how this component fits into the overall System with some descriptive text explaining the diagram.

Requirements

The section defines the requirements for the House Mate Model API component. Each model entity specifies that it is to have a unique identifier where specifically the House, Room, and Occupant interact uniquely named (and as we will later on in future assignments implement Authentication Services and the use of a GUID access token which I added completely unimplemented at the moment.)

The Appliance and Sensors are instantiated by their name and type that monitor the houses surroundings. What we are tasked with is importing a state .text file into the House Mate API (supported by the auth token) that simulates speaking commands for the House Mate that holds the state of the various

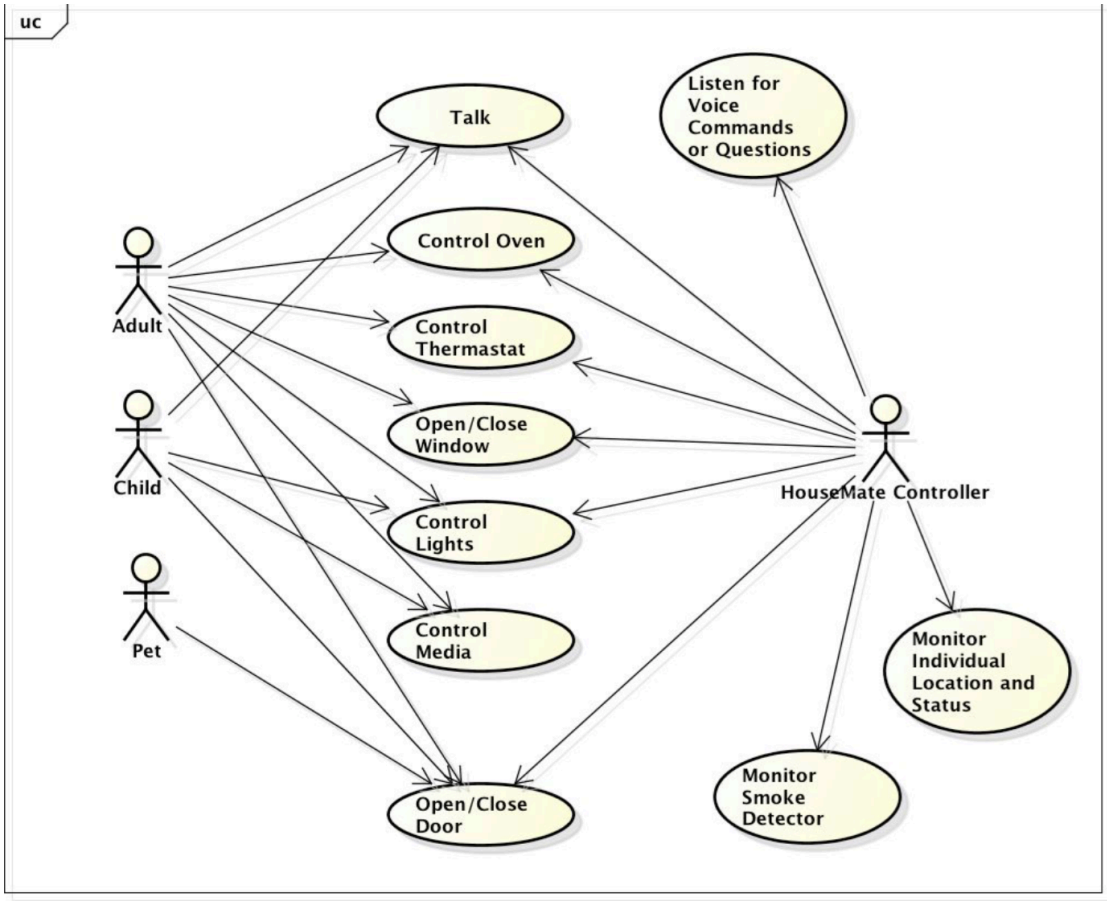
More specific details on the requirements are found in the House Mate Model Service Requirments.pdf.

Use Cases

Enumerate the use cases supported by the design,

1. In the case of the Adult Occupant for the House Mate to listen for and monitor:
    a. The Adult can talk to the HouseMate
    b. The Adult can control the oven
    c. The Adult can control the thermostat
    d. The Adult can open and close the window
    e. The Adult can control the lights
    f. The Adult can control the functions of Media (Pandora or TV, etc)
    g. The Adult can open and close the door
2. In the case of the Child Occupant for the House Mate to listen for and monitor:
    a. The Child can talk to the House Mate
    b. The Child can control the lights
    c. The Child can control the functions of Media
    d. The Child can open and close the door
3. In the case of the Pet Occupant for the House Mate to listen for and monitor:
    a. It can open and close the door
4. In the case of the House Mate Controller
    a. The House Mate can talk to the Occupants
    b. The House Mate can control the oven
    c. The House Mate can control the thermostat
    d. The House Mate can open and close the windows
    e. The House Mate can control the lights
    f. The House Mate can open and close the door
    g. The House Mate can listen for questions or commands of the Occupants
    h. The House Mate can monitor the location of the Occupants
    i. The House Mate can monitor the smoke detector

This design supports the following use cases:



Implementation

This section of the document will describe the implementation details for ...

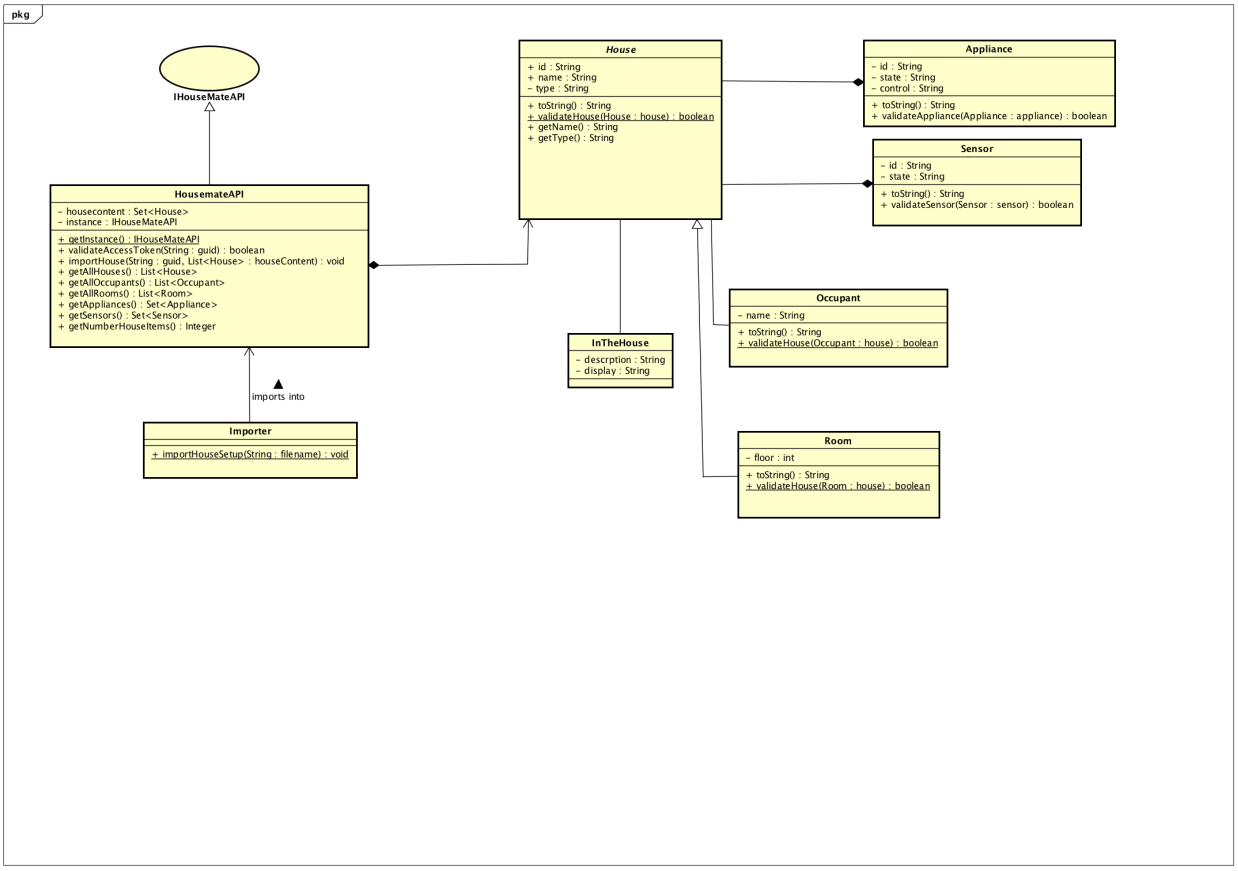
The implementation section should cover the following topics:

- *What are the classes, and their properties, associations and methods?*
- *What are the important interfaces and how they will be implemented?*
- *How are the requirements addressed?*

Class Diagram

The following class diagram defines the classes defined in this design.

House Mate Model Service Deign Diagram
CSCI E-97 Assignment #2



Class Dictionary

This section specifies the class dictionary for the class … defined within the package …

House (abstract)

This abstract class provides the basic properties and methods that all content items implementing this class must abide by. The point is that these attributes are common to all content types.

Methods

Method Name	Signature	Description
validateHouse <<static>>	(House house) : Boolean	Given a content item, validate that the attributes on that content item follow the required guidelines for all content items, regardless of type. This method is called by the sub-classes of Content to ensure that all types are valid.
toString <<override>>	() : String	Overrides generic toString() method and print out all the properties of a House item I used for debugging.

Property Name	Type	Description
id	String	Unique identifier for each house item. Required.
name	String	Name for the house. Required.
type	String	A description of what the content item is. Required.
inTheHouse	InTheHouse	Instance of the InTheHouse class that describes what kind of House item the current object is. Current values are InTheHouse.OCCUPANT or InTheHouse.Room

Room

Models a Room content item; extends House.

Methods

Method Name	Signature	Description
validateHouse <<static>>	(Room house) : Boolean	Given a Room content item, validate that the attributes follow the required guidelines for a Room content item. Calls House.validateHouse() to validate the basic properties common to all House items.
toString <<override>>	() : String	Overrides generic toString() method and print out all the properties of an Application content item. Useful for debugging.

Properties

Property Name	Type	Description
		Required.

Occupant

Models a Occupant content item; extends House.

Methods

Method Name	Signature	Description
validateHouse <<static>>	(Occupant house) : Boolean	Given an Occupant content item, validate that the attributes follow the required guidelines for a Occupant content item. Calls House.validateHouse() to validate the basic properties common to all content items.
toString <<override>>	() : String	Overrides generic toString() method and print out all the properties of an Occupant content item. Useful for debugging.

Properties

Property Name	Type	Description
		Required.

Appliance

Models an Appliance and the state of an Appliance within a House to be stored

Methods

Method Name	Signature	Description
validateAppliance <<static>>	(appliance : appliance) : Boolean	Given an Appliance, validate that the attributes follow the required guidelines for an Appliance. Called when importing new Appliances into the House to ensure state of Appliance in House.
toString <<override>>	() : String	Overrides generic toString() method and print out all the properties of an Appliance. Useful for debugging.

Properties

Property Name	Type	Description
id	String	The unique identifier for the appliance. Required.
name	String	Name of the appliance. Required.
state	String	Name of the given state of an appliance. Required.

control	String	The functionality of a given appliance. Required

Sensor

Models a Sensor and the state of a Sensor within a House to be stored

Methods

Method Name	Signature	Description
validateSensor <<static>>	(Sensor : sensor) : Boolean	Given a Sensor, validate that the attributes follow the required guidelines for a Sensor. Called when importing new Sensors into the House and recording their states
toString <<override>>	() : String	Overrides generic toString() method and print out all the properties of a Sensor. Useful for debugging.

Properties

Property Name	Type	Description
id	String	The unique identifier for the sensor. Required.
name	String	Name of the Sensor. Required.
state	String	Name of the state of a given sensor. Required.

InTheHouse

Its like a Marker class for the unique types of discrete content supported by the Mobile Application Store. Current valid types for InTheHouse objects are ROOM and OCCUPANT. When creating an Occupant or Room instance, an InTheHouse matching the object's type is created and set on the House item object.

Properties

Property Name	Type	Description
description	String	1 or 2 sentence description of what the enum item is.
display	String	Short, 1-word description of what the enum item is.

inTheHouse	Set<InTheHouse>	List of InTheHouse to match content items on.

Importer

Used to load Devices, Countries, and Content items (Applications, Wallpapers, and Ringtones) into the Product catalog.

Methods

Method Name	Signature	Description
importHouseSetup <<static>>	(String : filename) : void : throws ImportException, ParseException	Imports a .txt file and parses each line, attempting to add a new House item (Occupant, Room, Sensor, and/or Appliance with corresponding state) to the House Mate. Throws ImportException or ParseException for file handling issues or formatting problems with the .txt.

IHouseMateAPI <<interface>>

Interface class defining the public methods of the implementing HouseMateAPI. This simulates Occupants to talk to the House Mate. Allowed authenticated and authorized occupants to have clearance to the various features of the House by means of a GUID string token passed to restricted methods.

Methods

Method Name	Signature	Description
validateAccessToken	(String : guid) : Boolean	Given the supplied access token, determine if it is valid for accessing the methods of restricted interfaces (e.g., adding content to the product catalog). *Note: currently this method is mocked because the implementation of the Authentication / Authorization service is for future assignments; this method will return true for any string passed as of now.*
importHouse	(String : guid, List<House> housecontent) : void	Adds new House items to the product catalog (can be Occupant or Room items). *Restricted Interface: requires a GUID token to proceed.*

getAllOccupants	() : List<Occupant>	Returns all the matching Occupant content.
getAllRooms	() : List<Rooms>	Returns all the matching Room items.
getAllHouses	() : List<House>	Returns all the House items in the product catalog, including all Occupants and Rooms
getSensors	() : Set<House>	Returns all the Sensors in the House
getAppliances	() : Set<House>	Returns all the Appliances in the House
getNumberHouseItems	() : integer	Returns the total number of house items

HouseMateAPI

Concrete class implementing the IHouseMateAPI interface. Stores all the occupants, rooms, appliances, and sensors in-memory. Singleton instance.

Methods

Method Name	Signature	Description
getInstance <<static>>	() : HouseMateAPI	Static getter. HouseMateAPI is a singleton, so the only way to get an instance of one is to call this method.

Properties

Property Name	Type	Description
housecontent	Set<House>	Holds all the House items in-memory.
instance <<private>>	IHouseMateAPI	Reference to the singleton IHouseMateI instance.

Implementation Details

The House Mate API is implemented as a Singleton which is explaind by the static getInstance() method. When you load the commands into the House Mate Controller, The House Mate maintains the states of each entity a single instance each.
The management interface holds each component of the House Mate but the way I chose to model it was from the perspective of the House. The house gives you a snapshot of everything.

Testing

Provide a testing strategy for testing the component.

- *Functional*
- *Performance*
- *Regression*
- *Exception Handling*

The TestDriver class will exercise both importing the commands into the House Mate Controller via public methods on the HouseMateAPI for now. The static main() method “listens” for the user’s commands and calls the import method on the HouseMateAPI to load them.

Risks

Document any risks identified during the design process.
Are there parts of the design that may not work or need to be implemented with special care or additional testing?

I didn’t get around to finishing the actual import functionality. I blame that partly on late design indecisiveness in wanting to include an equals object comparison method and hash code unique identifier. I also wanted to add a search feature that would simulate a “smarter home”.