House Mate Model Service Design Document .

Date: 2/22/2017
Author: Bry Power

Introduction

Overview

This document provides the design details for the House Mate Model Service, which is part of a larger system called The House Mate System. The HouseMate system consists of a collection of sensors that can be used to monitor and control appliances throughout the house.

The goal of this system is to enable control and modification over houses, occupants, rooms, appliances and sensors through voice commands. The House Mate Model Service acts as manager of the entities of the system. The Model API supports creating, querying and updating the entities.

Requirements

The House Mate Model service contains one or more Houses. A house contains Rooms and Occupants. An Occupant can be either an adult, child, or and animal and can be known or unknown. A Room contains multiple Devices. A device can either be a Sensor or an appliance. Sensors provide information about the state of other entities. An appliance is similar but can be controlled through commands.

To parse commands, a command line interface is supported. The commands can be listed in a file to provide a configuration script. The CLI uses the ModelGraph API for processing the commands to create, update, or query the service.

This section defines the requirements for the House Mate model service.

Use Cases

Enumerate the use cases supported by the design,
This design supports the following use cases:

Include a Use Case Diagram.

The House Mate model service supports use cases for three actors

1. People
2. Pets
3. The housemate controller

The possible use cases for pets are:

1. Wake up/go to sleep
2. Go to another room

The use cases for people are:

1. Wake up/go to sleep
2. Go to another room
3. Issue a command

The use cases for Ava are:

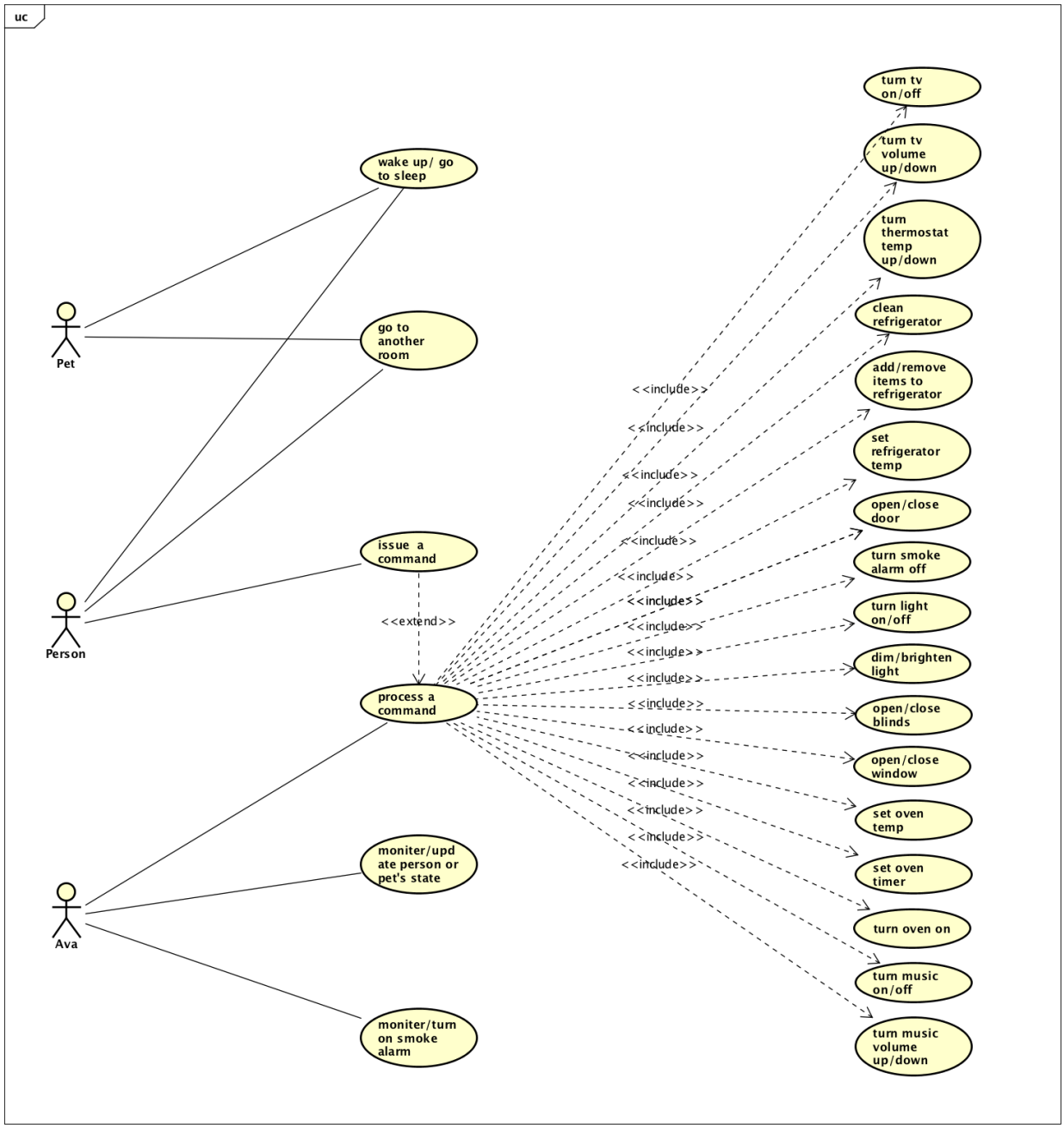
1. Process a command
2. Moniter/update person or pet's state
3. Moniter/turn on smoke alarm

The possible commands that people may issue and Ava may process are:

1. Turn tv on/off
2. Turn tv volume up/down
3. Turn thermostat temp up/down
4. Clean refrigerator
5. Add/remove items to refrigerator
6. Set refrigerator temp
7. Open/close door
8. Turn smoke alarm off
9. Turn light on/off
10. Dim/brighten light

11. open/close blinds
12. open/close window
13. set oven temp
14. set oven timer
15. turn oven on
16. turn music on/off
17. turn music volume on/off

The following use case diagram illustrates the use cases supported by the House Mate model service/



Implementation

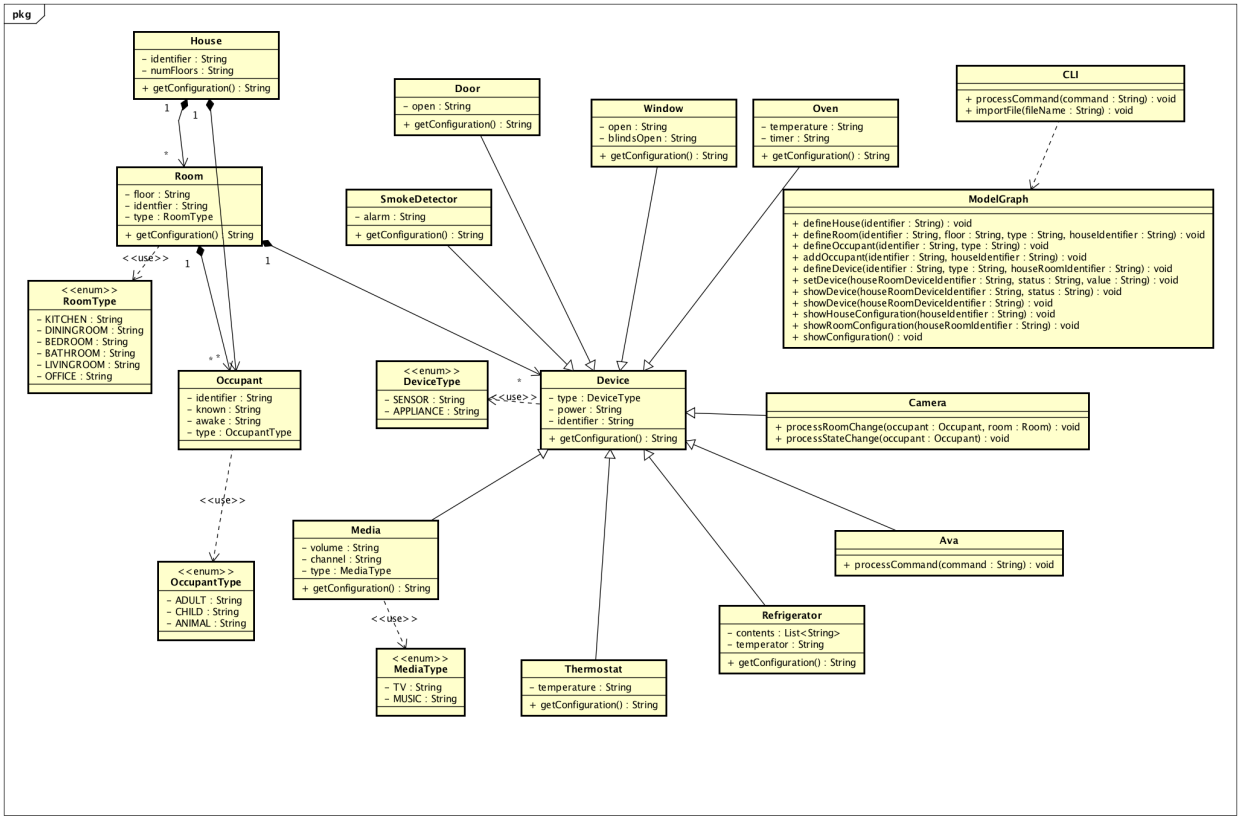
This section of the document will describe the implementation details for ...

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.



Class Dictionary

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

House

Class used to model a house. A house can contain any number of rooms and of occupants in a map from the identifier to the object for querying.

Properties

Property Name	Type	Description
identifier	String	Private unique non-mutable identifier of the house.

numFloors	String	Private string representation number of floors in the house.

Associations

Association Name	Type	Description
roomMap	Map<String, Room>	Private association for maintaining the active set of rooms in the house. The map key is the room identifier and value is the associated Room. Room identifiers are case insensitive.
occupantMap	Map<String, Occupant>	Private association for maintaining the active set of occupants in the house. The map key is the occupant identifier (the name) and value is the associated Occupant. Occupant identifiers are case insensitive.

Methods

Method Name	Signature	Description
getConfiguration	():String	Public method that returns a string representing the current configuration of the house. The number of floors, rooms and occupants currently associated with the house.

Room

Class used to model a room. A room can contain any number of occupants and devices.

Properties

Property Name	Type	Description
identifier	String	Private unique non-mutable identifier of the room.
floor	String	Private string representing the floor of the house that the room is on.

RoomType	enum	A public enumeration of the possible types of rooms. They are KITCHEN, DININGROOM, BEDROOM, BATHROOM, CLOSET, LIVINGROOM, and OFFICE
type	RoomType	Private type of the room.

Associations

Association Name	**Type**	**Description**
occupants	Set<Occupant>	Private set of occupants that are in the room.
deviceMap	Map<String, Device>	Private association for maintaining the active set of devices in the room. The map key is the device identifier and value is the associated Device. Device identifiers are case insensitive.

Methods

Method Name	**Signature**	**Description**
getConfiguration	():String	Public method that returns a string representing the current configuration of the room. The floor, devices and occupants currently associated with the room.

Occupant

Class representing a living entity if the house – can be either an adult, child, or animal. Can be known or unknown, awake or asleep.

Properties

Property Name	**Type**	**Description**
known	String	Private string indicated whether the occupant is known or unknown.
identifier	String	Private unique non-mutable identifier or

		name of the occupant.
awake	String	Private string indicated whether the occupant is awake or sleeping.
OccupantType	enum	A public enumeration of the possible types of rooms. Can be ADULT, CHILD, or ANIMAL
type	OccupantType	Private type of occupant

Device

Parent class for all sensors and appliances.

Methods

Method Name	**Signature**	**Description**
getConfiguration	():void	Default public method that returns a string representing the current configuration of the device.

Properties

Property Name	**Type**	**Description**
identifier	String	Private unique non-mutable identifier or name of the device.
power	String	Private string representing whether the device is powered on or off
DeviceType	enum	A public enumeration of the possible types of devices. Can be a SENSOR or APPLIANCE.
type	DeviceType	Private type of the device.

SmokeDetector

Represents a smoke detector in the room. A smoke detector is DeviceType SENSOR and has

an alarm which is either on or off.

Methods

Method Name	Signature	Description
getConfiguration	():void	Public method that returns a string representing the current configuration of the Smoke detector. If it is on and if the alarm is on or off.

Properties

Property Name	Type	Description
alarm	String	Private indicates if the alarm is ON or OFF.

Camera

Represents a camera in the room. Cameras detect if there is a state or room change with an occupant. A camera is DeviceType SENSOR.

Methods

Method Name	Signature	Description
processRoomChange	(occupant:Occupant, room:Room):void	Private method to process a change in room of an occupant
processStateChange	(occupant:Occupant):void	Private method to process a change in the state (sleeping or awake) in one of the occupants

Ava

Represents an Ava in the room. Avas listens for voice commands from an occupant and processes the command by passing it to the ModelGraph API processCommand method.. An Ava is DeviceType SENSOR.

Methods

Method Name	Signature	Description

processCommand	(command:String):void	Process a detected command

Media

Represents a media playing device. The type of media can be either a tv or a music player. Both have a volume level and a current channel setting. A media player is DeviceType APPLIANCE.

Methods

Method Name	Signature	Description
getConfiguration	():void	Public method that returns a string representing the current configuration of the media player.

Properties

Property Name	Type	Description
volume	String	Private - the current volume setting
channel	String	Private - the current channel media is turned to.
MediaType	enum	A public enumeration of the possible types of media player. Can be TV or MUSIC
type	MediaType	Private type of media player.

Thermostat

Represents the thermostat of the room. Can set the temperature in degrees Fahrenheit. A thermostat is DeviceType APPLIANCE.

Methods

Method Name	Signature	Description
getConfiguration	():void	Public method that returns a string representing the current configuration of the thermostat.

Properties

Property Name	Type	Description
temperature	String	Private string representing the degrees Fahrenheit

Refrigerator

Represents the refrigerator. Has a list of items in the refrigerator which can either be added or removed, as well as a temperature. A refrigerator is DeviceType APPLIANCE.

Methods

Method Name	Signature	Description
getConfiguration	():void	Public method that returns a string representing the current configuration of the refrigerator.

Properties

Property Name	Type	Description
contents	List<String>	Private list of the current contents of the refrigerator
temperature	String	Private string representing the temperature of the refrigerator

Door

Class represents a door that can be open or closed. A door is DeviceType APPLIANCE.

Methods

Method Name	Signature	Description
getConfiguration	():void	Public method that returns a string representing the current configuration of the door.

Properties

Property Name	Type	Description
open	String	Private String representing if the door is open

Window

Class represents a door that can be open or closed. The window has blinds that can be opened or closed. A window is DeviceType APPLIANCE.

Methods

Method Name	Signature	Description
getConfiguration	():void	Public method that returns a string representing the current configuration of the window

Properties

Property Name	Type	Description
open	String	Private string representing if the window is open
blindsOpen	String	Private string representing if the blinds in the window are open

Oven

Class representing an oven. Has a temperature and a timer that can be set.

Methods

Method Name	Signature	Description
getConfiguration	():void	Public method that returns a string representing the current configuration of the oven.

Properties

Property Name	Type	Description
temperature	String	Private string representing the temperature

		of the oven in degrees Fahrenheit
timer	String	Private string representing the number of minutes the timer is currently set to

CLI

The CLI class imports a file of commands, parses and processes each line as one command.

Methods

Method Name	**Signature**	**Description**
processCommand	(command:String):void	Takes the command and processes it through the API.
importFile	(filename:String):void	Opens the given file of commands and passes each one to the processCommand method.

ModelGraph

The API class is used to configure and update the state of House Mate model system. Can define, update, and show entities.

Methods

Method Name	**Signature**	**Description**
defineHouse	(String identifier):void	Creates a house with the given unique identifier
defineRoom	(String identifier, String floor, String type, String houseIdentifier):void	Creates a room in the house with the unique to the house identifier on the specified floor in the specified house.
defineOccupant	(String identifier, String type):void	Creates an occupant of the specified type. The type can be adult, child, or animal.
defineDevice	(String identifier, String type, String houseRoomIdentifier):void	Creates a device – either a sensor or appliance in the indicated room.

addOccupant	(String identifier, String houseIdentifier):void	Adds a already defined occupant with the identifier to the a house with the house identifier
setDevice	(String houseRoomDeviceIdentifier, String status, String value):void	Sets the property indicated by the status parameter to the value of the specified device.
showDevice	(String houseRoomDeviceIdentifier, String status):void	Prints to the console the current specified status of the device.
showDevice	(String houseRoomDeviceIdentifier):void	Prints all of the configuration of the specified device.
showHouseConfiguration	(String houseIdentifier):void	Prints to the console the configuration of the specified house
showRoomConfiguration	(String houseRoomIdentifier):void	Prints to the console the configuration of the specified room
showConfiguration	():void	Prints to the console all of the current houses and occupants.

Implementation Details

The file of queries is read in by the CLI class. The importFile method iterates over the lines in the file, checks to make sure they have content, and passes each one to the processCommand method. This method parses the command and calls the appropriate method in the modelGraph API class. The methods in this class create, print, and update entities in the system. Houses, rooms, occupants and devices can be created. When they are created, they are added to a map with the string identifier being the key for later querying. Rooms and occupants can be added to a house. Devices can be added to a room. The various states of appliances can be set. The API is able to dynamically call the set method depending on what is passed in as the status parameter. The configuration of the system, a house, a room, or a device can also be printed to the console.

Exception Handling

Problems with the file name or with badly formatted commands will result in a ImporterException being thrown. Other possible exceptions in the importing or processing of commands are: NoSuchMethodException, SecurityException, IllegalAccessException, InvocationTargetException, and ImportException

Testing

The a TestDriver class implements a static main() method. The main() method as a parameter the file name of configuration commands. This is passed to the CLI class inportFile method. The TestDriver class should be is defined inthe package "cscie97.asn2.test".

Risks

Because of the in-memory implementation, the number of entities is limited by the memory allocated to the JVM.