Smart city controller service design document

Date: 10/17/20

Author: Yuri Machkasov

Reviewer:

Contents

Introduction:

The Smart City Controller Service is responsible for monitoring events generated by the IoT devices (or simulated) and directing commands to one or more devices in response..

The service utilizes the publish-subscribe model to be notified of the events by the Model Service, and abstracts the commands by delegating their execution to the objects in the hierarchy of the devices.

Requirements:

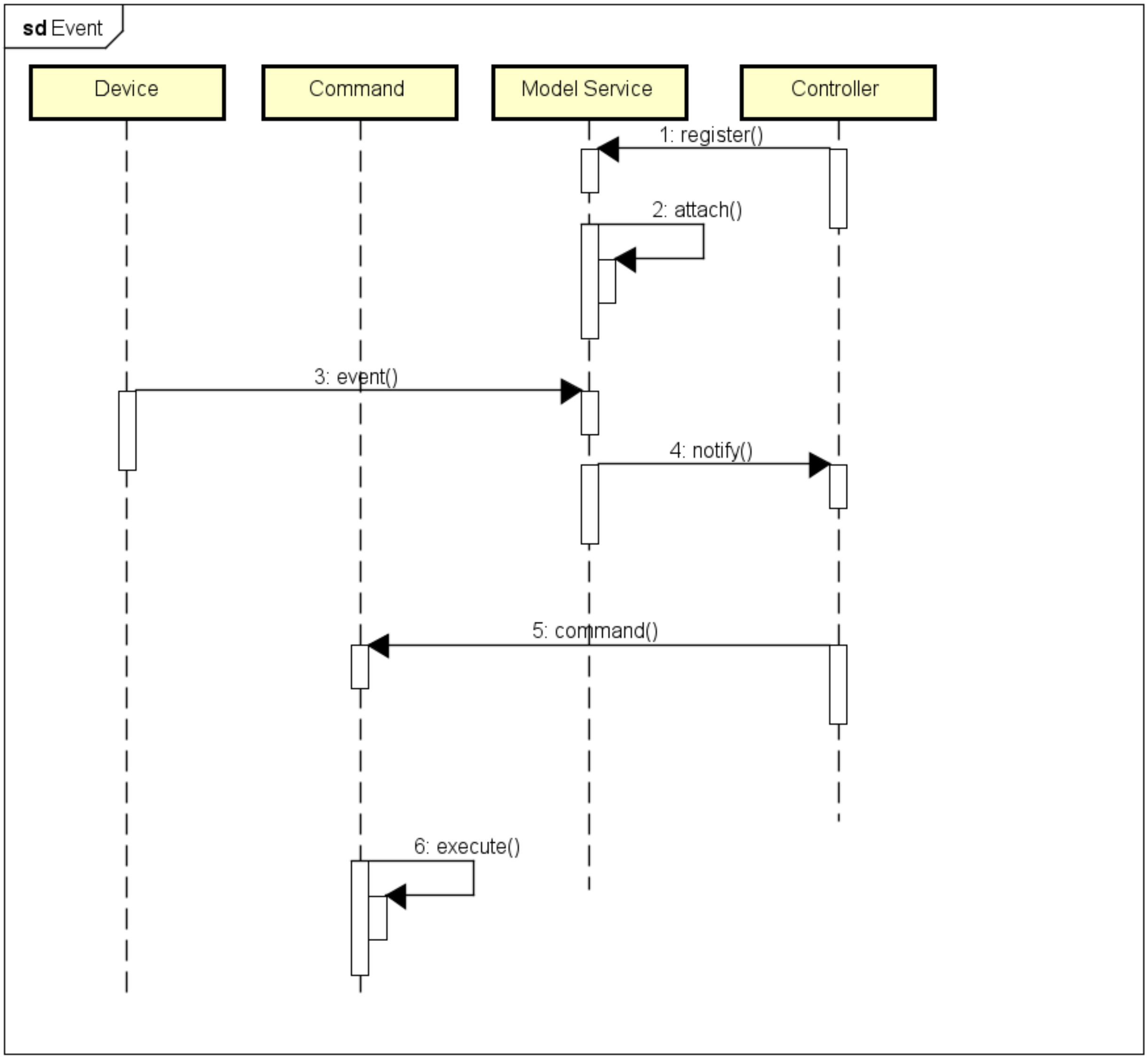
The service will provide the framework for consuming device events and forming responses in form of commands which will be routed through the model service to one or more devices for execution. The options will include a single device, all devices within a city, or devices closest to a certain location. Each device in response to a command can change its own state or location, cause a message to be broadcast through its speakers, or cause limited changes in the state of a person or a city object (for example, interact with the blockchain account through the Ledger service).

The interface will be implemented through the Observer design pattern: the service will subscribe to the messages emitted by the Model Service as the subject. The response will utilize the Command design pattern: the messages emitted by the service will take the form of instances of a command interface containing all necessary information for the device objects to effect the necessary changes.

For more details on supported interactions please refer to the requirements document.

Use Cases:

All use cases will follow the same sequence, represented on this diagram:



The first two steps are preliminary and not repeated; they represent the Controller service registering with the Model Service as a subscriber.

When a device generates an event (or one is imitated by the Model Service itself), the Model Service publishes it, notifying all subscribers, including the Controller. In response to the transmitted event, the Controller forms an instance of a command and causes it to be executed.

The service will recognize the following broad categories of events and responses:

Events:

1. A sensor event regardless of which device emitted it. These include events generated by CO2 sensors, microphones and cameras on every device.

2. A sensor event which is independent of the device but relies on its location. These also include events that require a response at a particular location.

3. A sensor event that is specific to a device type. These include interactions with devices capable of accepting payments.

Responses:

The response commands are capable of being directed at

1. a device (either change the state of the device or invoke a broadcast on its internal speaker)

2. a city object, or

3. a person object

A device command can also propagate itself to a group of devices (either explicitly defined or assembled according to specified criteria)

A command can also be a collection of commands; in this case it simply invokes all of them in a particular order.

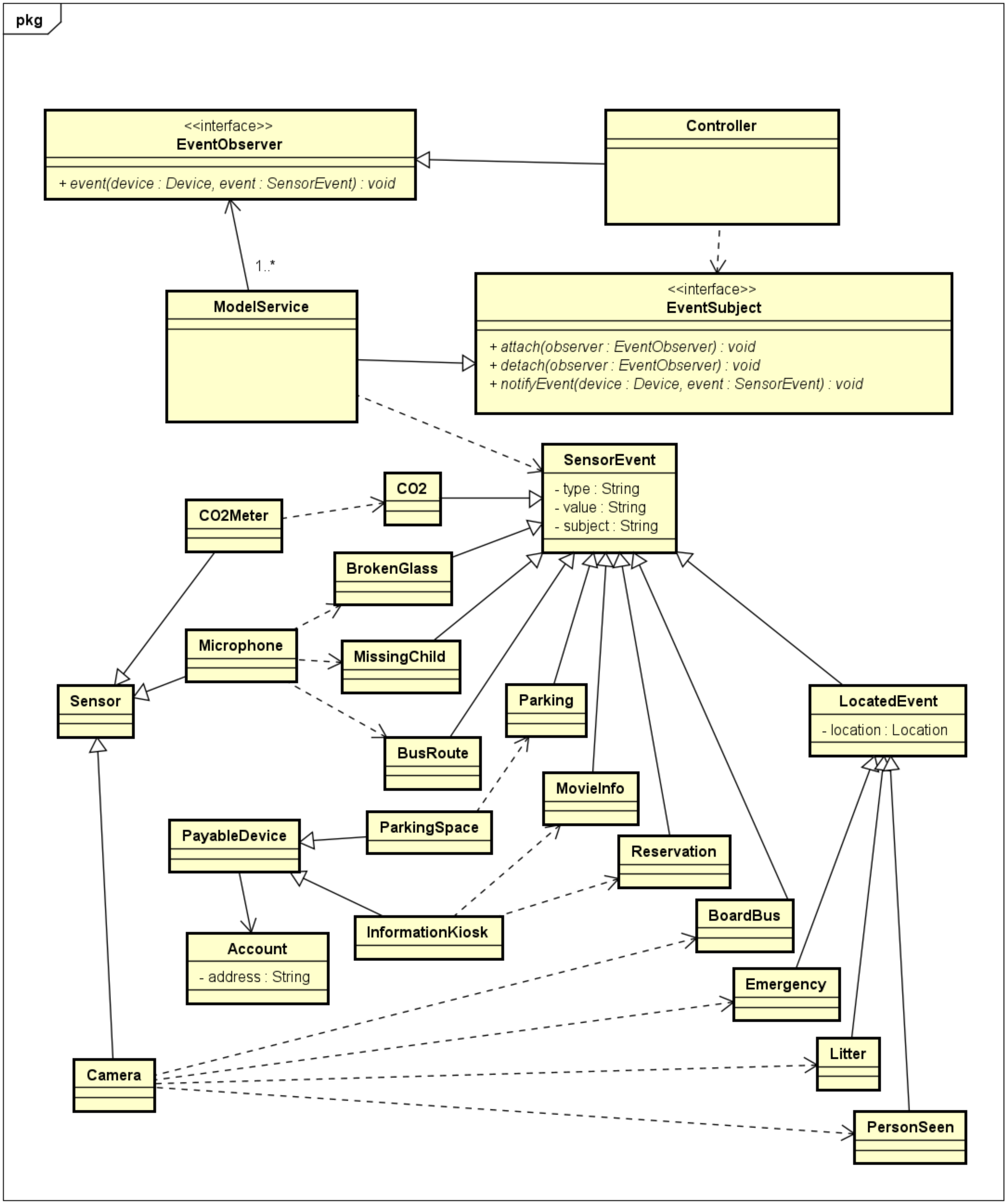
Within this framework, the current implementation handles the following cases, fulfilling the requirements outlined in the requirement document:

event type	**event parameters**	**response object(s)**	**response command(s)**	**command parameters**
emergency	location	city	announce and change state of robot devices	announcement text, device template
accident	location	reporting device	announce and change state of 2 closest robots	announcement text, device template
CO2	level	city	change state of car devices	device template
litter	person, location	reporting device	announce and change state of closest robot	announcement text, device template
		person	charge account	fee
broken glass	location	reporting device	change state of closest robot	device template
person seen	location	person	change location	location
missing child	person	reporting device	announce and change state of the robot closest to missing person’s location	announcement text, missing person’s location, device template
parking	vehicle	reporting device	charge account	fee
bus route	person	reporting device	announce	announcement text
board bus	person	reporting device	announce	announcement text
		person	charge account	fee
movie info	person	reporting device	announce and change state	announcement text, device template
movie reservation	person	reporting device	announce	announcement text
		person	charge account	fee

Implementation

Class diagram: event model

This diagram represents the hierarchy of supported events and the interaction of the Controller with the Model Service



Class description: event model

The design of this service utilizes classes defined and implemented as part of the Model Service. These classes are ModelService itself, Sensor (and its subclasses CO2Meter, Microphone and Camera), PayableDevice (and its subclasses ParkingSpace and InformationKiosk), Location and Account. The internal structure and functionality of them is not relevant to this component, and they are included in the diagram as placeholders. On the other hand, while the SensorEvent class has been defined earlier, it is in this component that it is extended and used, so we transfer its structure into the current diagram intact.

EventObserver

The interface that represents one part of the Observer pattern. Its only method is guaranteed to be invoked whenever an event fires or is being simulated. The Controller class implements this interface and routes the notifications for processing.

Methods

Method name	Signature	Description
event	device: Device event: SensorEvent	Is invoked for each sensor event; supplies the event itself and the device to which the reporting sensor belongs

EventSubject

The other half of the Observer pattern. Provides API for registering and unregistering observer instances, and sends notification of the events to all subscribers. Model Service is now implementing this interface.

Methods

Method name	Signature	Description
attach	observer: EventObserver	Registers the provided observer and starts publishing events to it
detach	observer: EventObserver	Unregisters the provided observer and stops publishing events to it

The following events are recognized by the Controller and produce a response in form of a Command. All of them set the corresponding fields in the generic SensorEvent structure.

CO2

Attributes

Attribute name	Attribute type	Description
type	String	the string “CO2”
value	String	the string representation of the level of carbon dioxide
subject	String	not set

BrokenGlass

Attributes

Attribute name	Attribute type	Description
type	String	the string “BrokenGlass”
value	String	not set
subject	String	not set

MissingChild

Attributes

Attribute name	Attribute type	Description
type	String	the string “MissingChild”
value	String	not set
subject	String	the id of the missing person

BusRoute

Attributes

Attribute name	Attribute type	Description
type	String	the string “BusRoute”
value	String	the string representation of the query text as recognized by speech-to-text device
subject	String	not set

Parking

Attributes

Attribute name	Attribute type	Description
type	String	the string “Parking”
value	String	the length of time the vehicle has been parked
subject	String	the id of the vehicle

MovieInfo

Attributes

Attribute name	Attribute type	Description
type	String	the string “MovieInfo”
value	String	the string representation of the query text as recognized by speech-to-text device
subject	String	not set

Reservation

Attributes

Attribute name	Attribute type	Description
type	String	the string “Reservation”
value	String	the string representation of the query text as recognized by speech-to-text device
subject	String	id of the person making the reservation

BoardBus

Attributes

Attribute name	Attribute type	Description
type	String	the string “BoardBus”
value	String	not set
subject	String	id of the person boarding

The following events are specific to the location of the device reporting them. As such, they extend LocatedEvent, a subclass of SensorEvent that contains location information.

Emergency

Attributes

Attribute name	Attribute type	Description
type	String	the string “Emergency”
value	String	the string representation of the emergency type
subject	String	not set

Litter

Attributes

Attribute name	Attribute type	Description
type	String	the string “Litter”
value	String	not set
subject	String	the id of the person littering

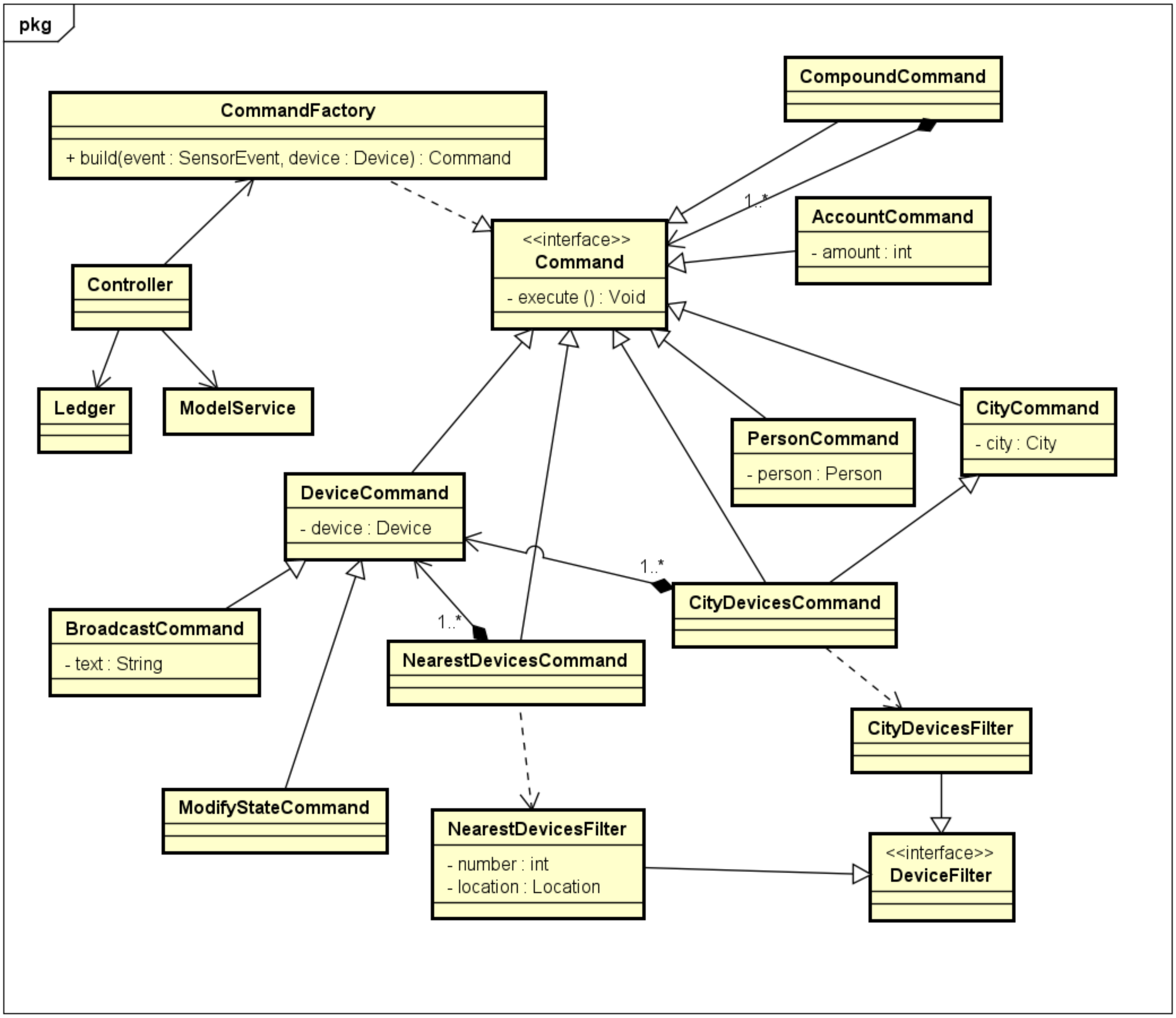
PersonSeen

Attributes

Attribute name	Attribute type	Description
type	String	the string “PersonSeen”
value	String	
subject	String	the id of the person

Class diagram:

This diagram represents the hierarchy of commands constructed by the Controller. Each instance of the Command interface encapsulates all information that is needed to execute it through the ModelService APIs.



Class description: commands

CommandFactory

A singleton implementation of a Factory pattern that constructs an appropriate instance of the Command interface using information contained in the event and device objects.

Methods

Method name	Signature	Description
build	device: Device event: SensorEvent	Static method constructing the Command instance

Command

The interface representing a command. Implementers need to override its method to effect the actual API calls.

Methods

Method name	Signature	Description
execute	void	Uses the information contained in the command instance to call the appropriate APIs of the model and ledger services

CompoundCommand

The implementation of the Command interface that contains a collection of commands in an ordered list. Its execution consists of calling the execute() method on them.

Attributes

Attribute name	Attribute type	Description
commands	List<Command>	the command collection

AccountCommand

The implementation of the Command interface that directs the Ledger service to add or subtract the specified sum from the specified account.

Attributes

Attribute name	Attribute type	Description
account	Account	the account address
amount	int	the amount (in units) to be added to the blockchain account; negative number to subtract

CityCommand

The implementation of the Command interface that is the parent of all commands operating on the City object.

Attributes

Attribute name	Attribute type	Description
city	City	the city object template

PersonCommand

The implementation of the Command interface that is the parent of all commands operating on the Person object.

Attributes

Attribute name	Attribute type	Description
person	Person	the person object template

DeviceCommand

The implementation of the Command interface that is the parent of all commands operating on the Device object.

Attributes

Attribute name	Attribute type	Description
device	Device	the device object

BroadcastCommand

Execution of this command consists of broadcasting the specified text through text-to-speech transformation on the device’s speaker

Attributes

Attribute name	Attribute type	Description
text	String	the text to be broadcast

ModifyStateCommand

Execution of this uses the polymorphic updateDevice () API of ModelService to transfer the set attributes in the template to the actual Device object.

DeviceFilter

This interface produces a list of devices depending on conditions specified in its instance.

Methods

Method name	Signature	Description
filter	void	Uses the information contained in the filter instance to produce a list of devices corresponding to its conditions

CityDevicesFilter

Implements the DeviceFilter interface to enable forming lists of devices belonging to the specified city.

Attributes

Attribute name	Attribute type	Description
city	City	the city object

NearestDevicesFilter

Implements the DeviceFilter interface to enable forming lists of the specified number of devices currently closest to the specified location.

Attributes

Attribute name	Attribute type	Description
number	int	the number of closest devices to collect
location	Location	the location for search

MultiDeviceCommand

Uses the supplied filter to produce a list of commands; its execution consists of calling the execute() method on them.

Attributes

Attribute name	Attribute type	Description
filter	DeviceFilter	the instance of filter used to form the device list

NearestDevicesCommand

The instance of MultiDeviceCommand that uses the NearestDevicesFilter.

CityDevicesCommand

The instance of MultiDeviceCommand that uses the CityDevicesFilter.

Command syntax

Implementation details

Exception handling

Testing

Risks