Smart City Controller Service Requirements

Brett Bloethner

Introduction

This document provides system architecture details for the Smart City Software System's Smart City Controller Service. In this document you will find use case diagrams, class diagrams, sequence diagrams and implementation details that will help outline and explain the Smart City Controller Service design.

Table of Contents

Overview

The Smart City Controller Service can be considered the "brain" of the Smart City. One of the core requirements of the Smart City is that it needs to respond events that happen in the city. These events include things that people initiate, like buying movie tickets as well as metrics from sensors on smart city IoT devices. The Smart City Controller Service is the component responsible for coordinating how the city reacts to each of these events as they occur. The Controller Services ability to be notified of and respond to events is what allows it to coordinate responses to events in the city.

Requirements

This section describes the features required of the Controller Service. In order to effectively support the other components of the Smart City System, the Smart City Controller Service must…

Monitor the City's IoT Devices by…

- Implementing the Observer Pattern, allowing the Controller service to "subscribe" to a stream of events sent by the Smart City Model Service
- memorizing the last three CO2 readings form unique CO2 sensors in each city order to get an accurate overall CO2 reading for monitoring of air quality
- Appropriately parsing out voice command events to be responded to accordingly

- logging out all of the event and event responses processed by the Controller Service

Response to Events from the City's IoT Devices by…

- implementing the Command Pattern to allow effective queueing and execution of responses to events
- evaluating a set of predefined rules which determine if and how the Controller Service should respond to different types of events
- creating and executing commands, in response to events, that include generating and sending control messages back to the device from which the event was generated

Trigger Transactions in the Ledger Service by…

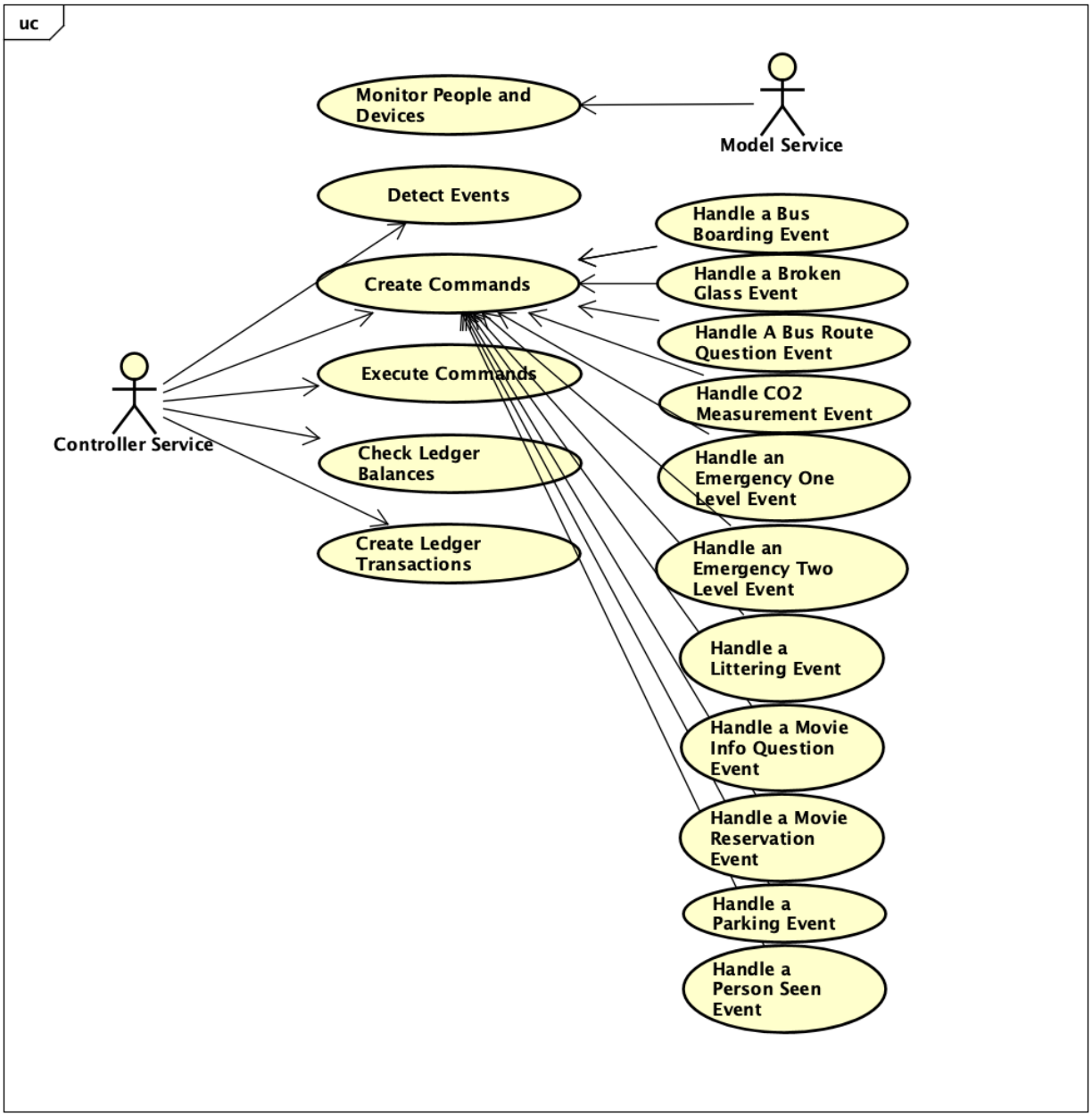
- sending transactions to the Smart City Ledger Service to be executed in response to events
- accurately handling an event in response to a successful transaction
- accurately handling an event in response to a failed transaction

Use Cases

Actors

- **Subject Model Service:** The Smart City Subject Model Service interacts with the Smart City Controller Service through one use case which is "Monitoring People and Devices." Through this use case, the Smart City Model Service provides events to the Smart City Controller Service for continuous people, device and city monitoring.
- **Controller Service:** The Controller Service the main actor in the Smart City Controller Service. It is the user of most of its own functionality. This is by design since its job is primarily to listen to events and process them from start to finish.

Below is the use case diagram which outlines what the expected actors are and what abilities they'll have in the application.



Monitor People and Devices

- The notify() function on the Smart City Model Service gets called as a result of an event occurring
- The Smart City Model Service calls the update() function on each observer watching it

Detect Events

- The Smart City Controller service update() function is called by the Smart City Model Service, notifying the Controller Service that an even has just occurred

Create Commands

- Create concrete commands for each of the event types below. In Each concrete command an execute() method is available, allowing the command to be processed.

Handle a Bus Boarding Event

- Get the relevant City, Device and Subject/Person from the Model Service
- Welcome the subject onto the bus using the bus speaker
- If the subject is a resident then charge their ledger account according to the Bus fare/fee

Handle a Broken Glass Event

- Get the relevant City and Device
- Dispatch the robot nearest to the event location to clean up the broken glass

Handle A Bus Route Question Event

- Get the relevant City, Device and Subject/Person from the Model Service
- Respond to the subjects question using the Bus speaker

Handle CO2 Measurement Event

- Determine whether or not the CO2 level is above or below 1000 and whether the last 3 CO2 events we of the same condition
- If the event is the 3rd consecutive event in that City with a level above 1000, get the City and disabled all cars in the city
- If the event is the 3rd consecutive event in that City with a level below 1000, get the City and enable all cars in the City

Handle an Emergency One Level Event

- Get the relevant City and Device from the Model Service
- Announce City wide safety announcement across all device's speakers on devices In the city limits
- deploy half of the nearest robots to help with emergency
- deploy other half of the nearest robots to help people take shelter
    - emit message from the Device that reported the event
    - deploy the 2 nearest robots to the location of the event

Handle an Emergency Two Level Event

- Get the relevant City and Device from the Model Service
- Emit safety message from the Device that reported the event
- Dispatch the 2 nearest robots to the location of the event to help

Handle a Littering Event

- Get the relevant City, Device and Subject/Person from the Model Service
- Emit an anti-littering message from the device that sensed the event
- Dispatch the nearest robot to clean up the garbage
- Use the Ledger Service to check the subjects account balance if the person/subject is a Resident
- Use the Ledger Service to charge the subject/person 10 unit for the movie reservation if the person/subject is a Resident

Handle a Movie Info Question Event

- Get the relevant City, Device
- Confirm the sensing device is a Kiosk
- Respond with the current movie schedule
- Change the kiosks display to the movie title cover

Handle a Movie Reservation Event

- Get the relevant City, Device and Subject/Person from the Model Service
- Confirm the sensing device is a Kiosk
- Use the Ledger Service to check the subjects account balance if the person/subject is a Resident
- Use the Ledger Service to charge the subject/person 10 unit for the movie reservation if the person/subject is a Resident
- Using the speaker on the sensing device, announce to the subject/person the confirmed reservation

Handle a Parking Event

- Get the relevant sensor City and sensor Device and Vehicle City and Vehicle Device from the Model Service
- Use the Ledger Service to confirm the Vehicle account has sufficient funds for the transaction
- Use the Ledger Service to charge the Vehicle account for the parking fee times the parking duration

Handle a Person Seen Event

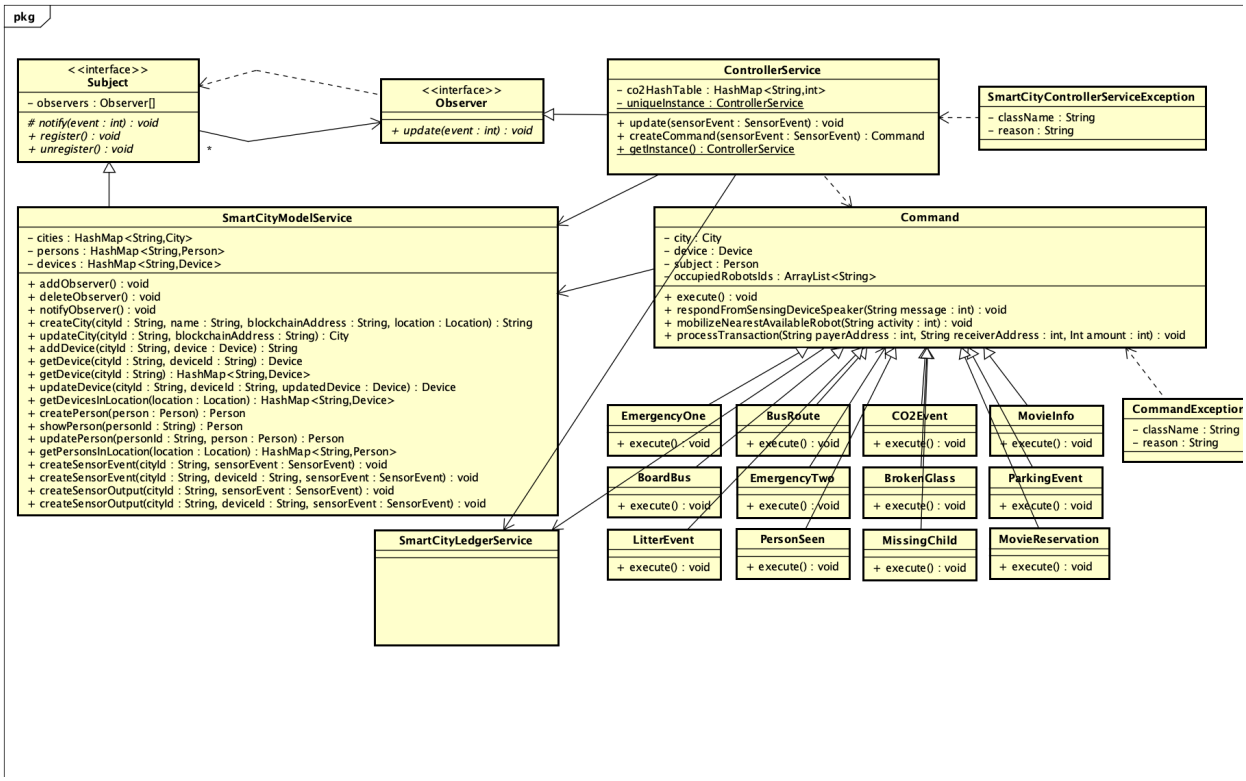
- Get the relevant City, Device and Subject/Person from the Model Service
- Update subject/person with a new location (the location of the event sensor)

Execute Commands

- Run the execute function on the concrete command
- Command will process the Event provided in accordance with the rule logic in the concrete command
- If there is a CommandException, then catch and report the error as a Controller Service Exception

Class Diagrams

The following class diagram defines the Smart City Controller implementation classes contained within the package “com.cscie97.smartcity.controller”. Below is the class diagram outlining the design and relation of all the classes in the Smart City Controller Service.



Class Dictionary

This section specifies the class dictionary for the Smart City Controller Service. The classes should be defined within the package “com.cscie97.smartcity.controller”.

Controller Service

The Controller Service is responsible for coordinating responses to events that occur in the Smart City. To do this, the Controller Service becomes the observer in an implementation of the observer pattern while the Smart City Model Service is the subject being observed. Once an event occurs, the Smart City Model Service notifies the Controller Service of the event and the Controller Service creates a Command based on the events specs. Immediately after creation, the Command is executed by the Controller Service.

Methods

Method Name	Signature	Description
getInstance	SmartCityControllerService	Returns the singleton instance of the SmartCityControllerService, creates one if it doesn't yet exist
createCommand	(SensorEvent sensorEvent) void	Behaves as a Command factory to create the correct concrete command for the passed in SensorEvent. Executes the command after creating it.
update	(SensorEvent sensorEvent)	Triggers the createCommand method. Interfaces with the observer pattern and is used by subjects to notify the SmartCityControllerService of new events.

Properties

Property Name	Type	Description
uniqueInstance	SmartCityControllerService	Stores the singleton SmartCityControllerService

Associations

Association Name	Type	Description
smartCityModelService	SmartCityModelService	Allows access to features in the SmartCityModelService singleton
smartCityLedgerService	SmartCityLedgerService	Allows access to the features in the SmartCityLedgerService singleton

Command

The Command class is the parent class of all the concrete commands. It takes in a variety of String ids for objects relevant to the Sensor Event and then fetches those objects in the constructor. If the ID is present as an argument and not null, for the City, Device and Subject, the Command class will automatically fetch each object so it is available for the concrete command to work with. In addition to that, the Command includes a method to use the sensing devices speaker to respond to a Sensor Event, a method to validate and process a simple Ledger transaction and a method to mobilize the nearest Robot to whatever activity you specify. Mobilized Robot's IDs are stored in the Command that way the same Robot cannot be dispatched twice.

Methods

Method Name	Signature	Description

respondFromSensingDeviceSpeaker	(String message):void	Respond to an event with the passed in message using the speaker of the device that sensed the event
mobilizeNearestAvailableRobot	(String activity):void	Mobilize the Robot nearest to the event location with the activity passed in, as long as the Robot ID is not in the occupied list
processTransaction	(String payerAddress,, String receiverAddress, Integer amount):void	Process a simple Ledger transaction from a payer to a receiver for a specified amount. Confirm the payer has an account and sufficient funds first.
execute	void	throw an error to show the execute method has not been properly overridden in the child class

Properties

Property Name	Type	Description
rand	Random	Java random util used to create random IDs for ledger transactions
occupiedRobotsIds	ArrayList<String>	List of IDs of Robots who have been dispatched for a particular event and are considered busy and not again dispatchable

Associations

Association Name	Type	Description
smartCityModelService	SmartCityModelService	Allows access to features in the SmartCityModelService singleton
smartCityLedgerService	SmartCityLedgerService	Allows access to the features in the SmartCityLedgerService singleton
city	City	The City object related to where the event occurred
device	Device	The Device object related to where the event occurred
subject	Person	The subject/person involved in the event

BoardBus extends Command

BoardBus is a concrete command that extends Command and overrides the execute method. In BoardBus, the execute method will use the Bus speaker to welcome the subject into the bus and then it will execute a ledger transaction for the associated bus fare.

Methods

Method Name	Signature	Description
execute	void	Welcome the subject onto the bus using the bus speaker and charge them bus fare

BrokenGlass extends Command

BrokenGlass is a concrete command that extends Command and overrides the execute method. In BrokenGlass, the execute method will dispatch the nearest Robot to the sensing device location to clean up the mess.

Methods

Method Name	Signature	Description
execute	void	Dispatch the robot nearest to the sensing device location to clean up the broken glass

BusRoute extends Command

BusRoute is a concrete command that extends Command and overrides the execute method. In BusRoute, the execute method will use the Bus speaker to tell the subject that the bus does in fact go to the specified location.

Methods

Method Name	Signature	Description
execute	void	If the sensing device is a Bus, then respond via speaker with an answer that the bus goes to the specified location

Properties

Property Name	Type	Description
locationString	String	The dialog location of where the subject is asking the bus if it goes to

CO2Event extends Command

CO2Event is a concrete command that extends Command and overrides the execute method. In CO2Event, the execute method gets all of the Cars in the event's City and it will set each car to either disabled or enabled based on the value of the isCarsEnabled Boolean.

Methods

Method Name	Signature	Description
execute	void	Get all of the Cars in the event's City, set the Cars' enabled boolean according to the provided isCarsEnabled Boolean provided.

Properties

Property Name	Type	Description
isCarsEnabled	Booelan	Dictates wether the Citys cars will be enabled or disabled

EmergencyOne extends Command

EmergencyOne is a concrete command that extends Command and overrides the execute method. In EmergencyOne, the execute method will announce a city wide emergency message,

then it will dispatch half of the nearest Robots to assist with the relief effort and send the remaining Robots to help people take shelter.

Methods

Method Name	Signature	Description
execute	void	Announce the emergency message in the city then dispatch half the robots to relief activity and dispatch the remaining robots to help people take shelter

Properties

Property Name	Type	Description
emergencyType	EmergencyOneTypeEnum	Enum selection of the type of emergency

EmergencyTwo extends Command

EmergencyTwo is a concrete command that extends Command and overrides the execute method. In EmergencyTwo, the execute method will output a stay calm announcement from the sensing devices speaker before it dispatches the two nearest Robots to assist with emergency. Right now, traffic accident is only available type of EmergecnyTwo.

Methods

Method Name	Signature	Description
execute	void	On the original sensing device, output a speaker announcement to stay calm, then dispatch the nearest two Robots to respond to the emergency

Properties

Property Name	Type	Description
emergencyType	EmergencyTwoTypeEnum	Enum selection of the type of emergency

LitterEvent extends Command

LitterEvent is a concrete command that extends Command and overrides the execute method. In LitterEvent, the execute method outputs a littering message from the original sensing device then it will dispatch the nearest Robot to clean the garbage before charging the offender 50 units in a ledger transaction.

Methods

Method Name	Signature	Description
execute	void	On the original sensing device respond with a do not litter message, then dispatch the nearest Robot to clean up the mess, then charge the offending person 50 units fine

MissingChild extends Command

MissingChild is a concrete command that extends Command and overrides the execute method. In MissingChild, the execute method will dispatch the nearest Robot to the sensing device location to next retrieve the child and bring them back to the original location. A calming message is also output from the speaker of the original sensing device.

Methods

Method Name	Signature	Description
execute	void	On the original sensing device, announce a message with the child's location, then dispatch the Robot nearest to the original sensing device to go a retrieve the child

MovieInfo extends Command

MovieInfo is a concrete command that extends Command and overrides the execute method. In MovieInfo, the execute method will use the speaker of the original sensing device to announce the current movie schedule if that device is a Kiosk.

Methods

Method Name	Signature	Description
execute	void	If the original sensing device is a Kiosk, then announce that Casablanca is showing and display Casablancas cover image

MovieReservation extends Command

MovieReservation is a concrete command that extends Command and overrides the execute method. In MovieReservation, the execute method will verify and process a transaction against the Subject for the price of the movie reservation. If the transaction goes through, the speaker of the original sensing device is used to announce the residents reservation.

Methods

Method Name	Signature	Description
execute	void	If the original sensing device is a Kiosk and the subject is a Resident then charge the resident for the tickets and announce via speaker the residents reservation

ParkingEvent extends Command

ParkingEvent is a concrete command that extends Command and overrides the execute method. In ParkingEvent, the execute method will get the City the car is registered to as well as the Vehicle object. It will then validate and process a Ledger transaction from the Vehicle to the Parking Space for the hourly fee times the duration of time parked.

Methods

Method Name	Signature	Description
execute	void	Get the Vehicle object and charge to the Vehicle account the fee on the device (parking space) times the provided duration

Properties

Property Name	Type	Description
cityOfVehicle	String	cityId of the City where the Vehicle is registered at
idOfVehicle	String	deviceID of the Vehicle
duration	Integer	How long, in hours, the Vehicle was parked in the parking space

PersonSeen extends Command

PersonSeen is a concrete command that extends Command and overrides the execute method. In PersonSeen, the execute method will get the person and update their location to the same location as the device that sensed the event.

Methods

Method Name	Signature	Description
execute	void	Update the subject (Person) location to the location of the original sensing device

CommandException extends Throwable

Exception used when errors occur in the Command class.

Properties

Property Name	Type	Description
className	String	Name of the class where the exception is thrown
reason	String	Reason for the exception

SmartCityControllerServiceException extends Throwable

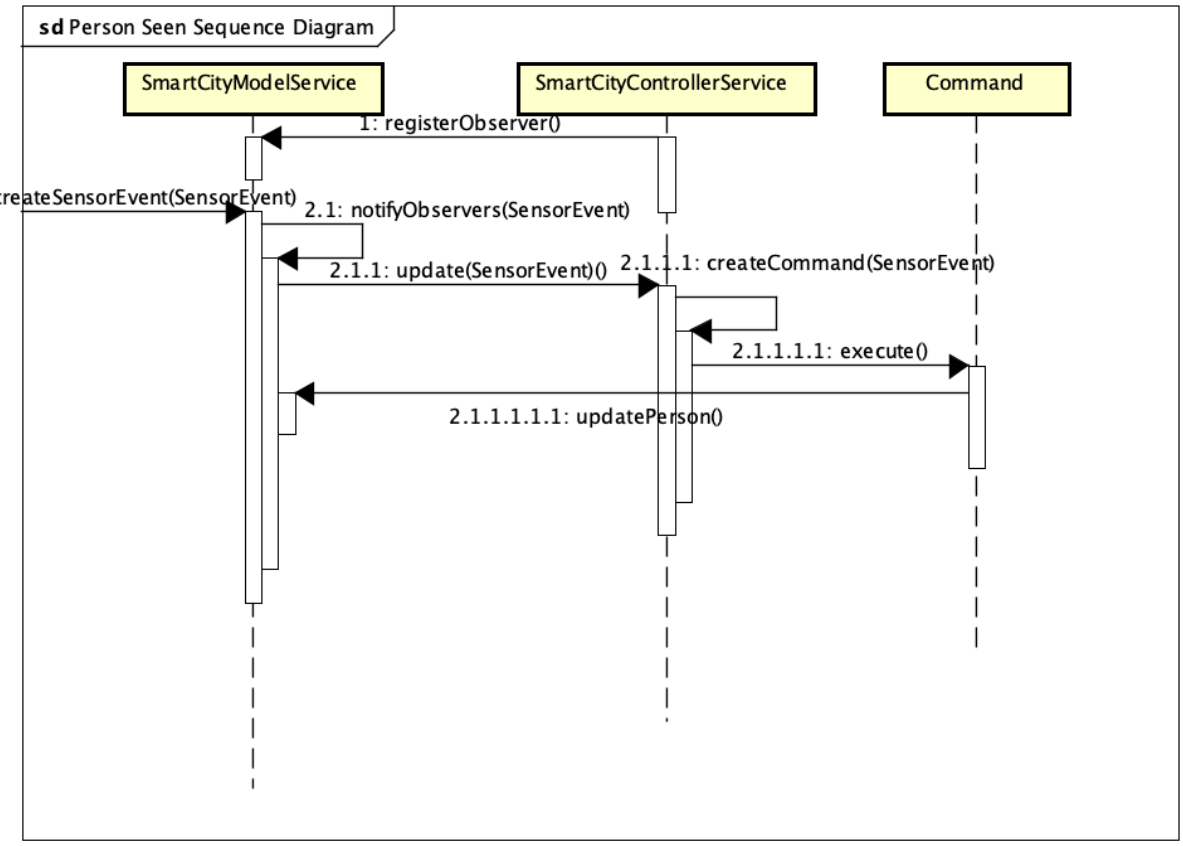
Exception used when errors occur in the SmartCityControllerService class.

Properties

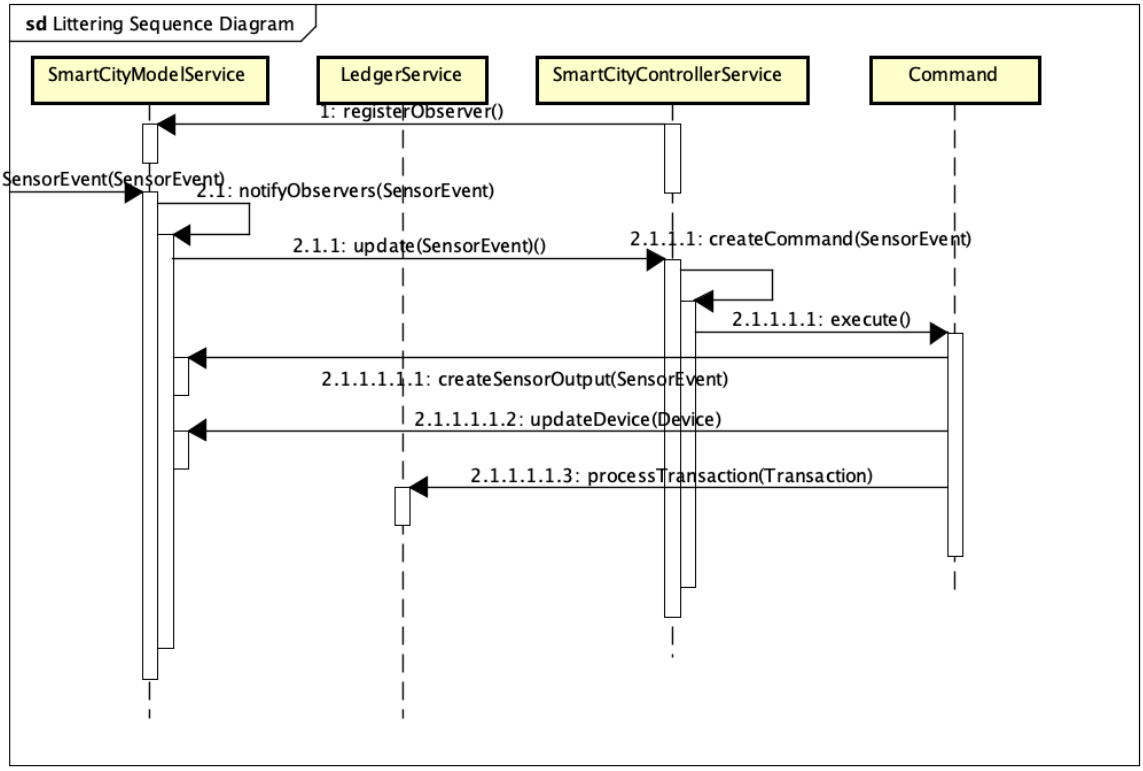
Property Name	Type	Description
className	String	Name of the class where the exception is thrown
reason	String	Reason for the exception

Implementation Details

The diagram below shows the sequence of events for what happens when a person is seen in the Smart City, starting from the moment the Smart City Controller Service is registered as an observer.



The diagram below shows the sequence of events for what happens when a littering event is detected in the Smart City, starting from the moment the Smart City Controller Service is registered as an observer.



All events start with the notify function. This is a core component of the observer patterns and allows the Smart City Model Service to tell the Smart City Controller Service when and event has occurred on one of the models. Next the Model Service used the event passed into Notify to call update on all of the objects observing the Model Service. This triggers the update function on the Smart City Controller Service. After the Controller Service receives the event, a command is created by the Controller Service by using the createCommand function. The createCommand function is basically a Command factory that will create a concrete command class based on whatever data is inside the Event object. Next, the execute function is called on the concrete command. This allows the rules in the concrete command to be processed and the event to responded to appropriately. Often times, a component of responding to an event will include calling back to the Smart City Model Service in order to update a device or relay a message. This is expressed in the last few steps where often times updateDevice will be called. However, it's important to note that the specific function called on the Device might not be updateDevice and it's even possible the Device might not need to be updated at all depending on the rules for the specific event being handled.

Exception Handling

Exception handling will be done primarily by way of making sure the Event objects include all the data needed to respond to an event. Exceptions will halt the command execution of a response to an event and will result in a error being logged rather than the event being responded to. Exceptions will also be raised if any other functions involved in the event

handling fail to process properly including but not limited to updating devices, creating sensor outputs and processing ledger transactions.

Testing

Testing was done using the provided test script and the extended custom test script which is titled Assignment_3_custom_sample_script.script.

Risks

I believe the main risks in this system come in the value parameter of the events being processed. Parsing out the important data for events like MovieReservation is a very fragile process where the user must speak to the system with the perfect pattern in order to trigger the right event. Ideally there would be some sort of natural language processing that would allow the system to still understand the users even with different patterns of words.

System Changes

Not very many changes were made compared to the original assignment documents. A few lines had to be changed in the sample script document to correct a few small errors in syntax and wording.