UNIVERSITÉ Concordia UNIVERSITY

Project: **Smart Home Simulator**

Project Report - Deliverable 3

STUDENT NAMES:

Gregory Tucker - 40092432
Aymen Metallaoui - 40057294
Tyler Znjog - 40005987
Adrien Kamran - 40095393
DeepKumar Patel - 40096716

Tutorial Section HC

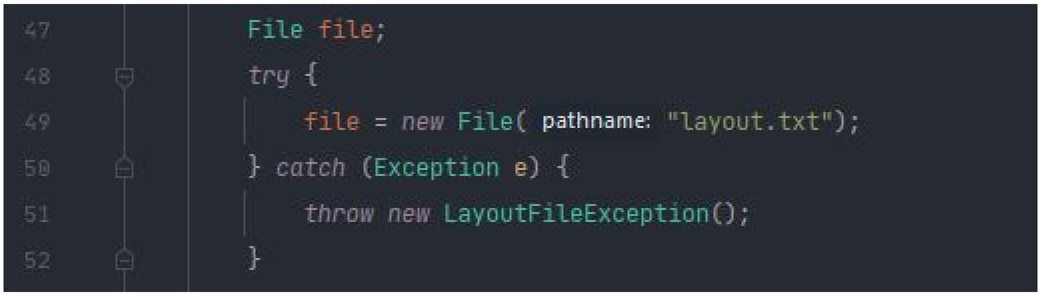
Github Repository: https://github.com/gregtuc/SmartHomeSimulator
Demo Link:https://drive.google.com/file/d/18uMdWtQ1XjAjwRYsniUPIvM90EuxAEa7/view?usp=sharing

Table of Contents

Functional Requirements **1**
Table of Contents 3
Exception Class 4
Refactoring 5
Layout Display Refactoring: 5
Code Optimization: 5
Summary: 6
Use Cases 7
UC1: Create a Zone 7
UC2: Manage Zone Temperature 8
UC3: Display a Room Temperature 9
UC4: Manage heating during summer 10
UC5: Manage Windows 11
UC6: Manage HVAC using User Location 12
UC7: Alert User to Temperature Issue 13
Class Diagram 14
Sequence Diagram (One for each Use Case) 15
State Machine Diagram 16
Activity Diagram 17

Exception Class

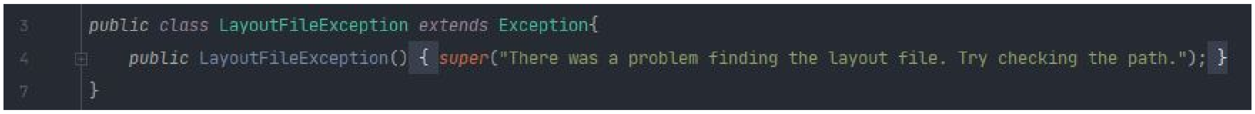
The custom exception class implemented for this delivery was class LayoutFileException. With commits coming from different developers and no primary package manager, the path's for text files were frequently changed (automatically) by developer IDE's. When the exception occurred, a generic error would appear with a large stack trace. For this reason, it became very difficult and time consuming to troubleshoot the error when it occurred. The solution to this was to create a custom exception and wrap the File declaration in a try catch, where errors occurring would result in a LayoutFileException being thrown. The code segment from LayoutParser.java is shown below:



```
File file;
try {
    file = new File( pathname: "layout.txt");
} catch (Exception e) {
    throw new LayoutFileException();
}
```

Figure 1: *Throwing a LayoutFileException.*

With the error thrown, a custom exception message would be printed, indicating that there was an issue finding the house layout file and to verify the path. An image of the exception class is shown below:



```
public class LayoutFileException extends Exception{
    public LayoutFileException() { super("There was a problem finding the layout file. Try checking the path."); }
}
```

Figure 2: *LayoutFileException Class.*

Refactoring

Layout Display Refactoring:

A primary target for refactoring was the code related to the layout display. Referencing the house-layout display was incredibly inefficient, as an array of FXML objects had to be created and passed in a complicated manner to whichever method or class need to reference it to make a change. Furthermore, room details were cluttered because all of the information was contained in a 50px by 50px FXML pane.

There were several issues with this implementation - the primary being code complexity. It was unclear to the entire team how exactly changes to the house-layout were made, as well as how information was stored and fetched. In order to clear up the clutter, a feature was implemented that opened a pop-up modal onclick of a room in the house display. This modal contained all of the house information in a clear and concise manner. This primary strength of this implementation was the change in coupling and cohesion. While both are good, it is best to strike a balance between the two. The issue was that modules were wildly dependent on one another with an extremely high amount of complexity and interdependence. This was reduced in github commits ***67ad92b, bac5b17, a9b144d*** for issue ***#43***.

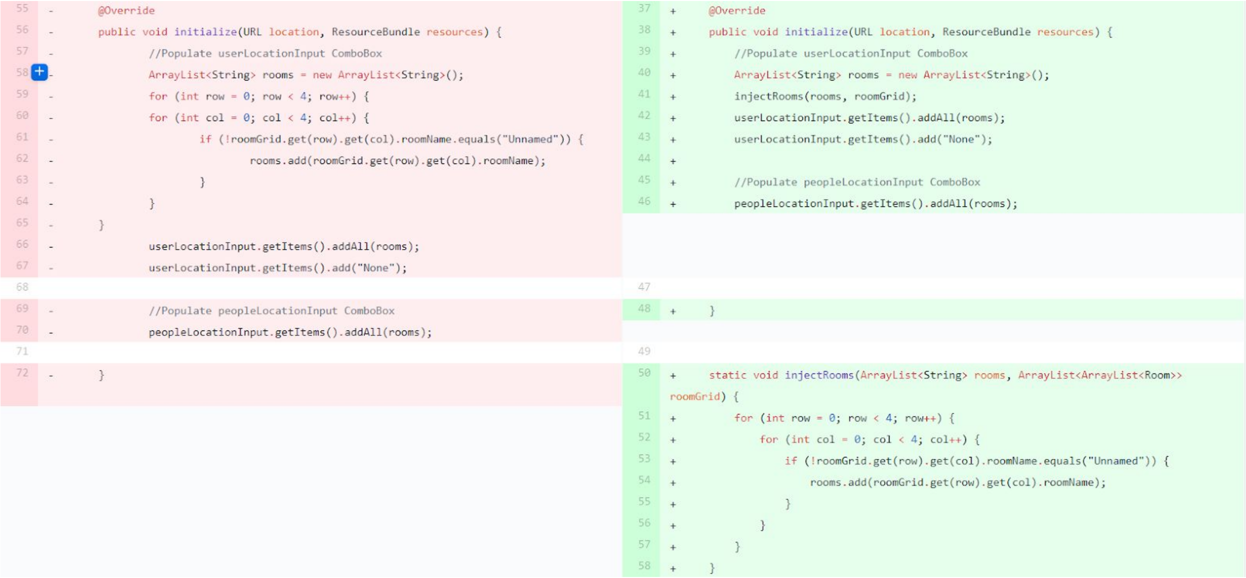
Code Optimization:

Through Delivery 1 and 2, a large amount of un-optimized and inefficient coding practices were used during periods of high-pressure with respect to deadlines. These inefficiencies came in the form of un-used imports and methods, large conditional blocks, complicated nested for-loops, syntax errors, redundant conditional booleans, and a large amount of duplicated code blocks.

In order to reduce these issues, the following changes were made:

1) Imports were optimized (unused imports were removed)
2) Methods were optimized (unused methods were removed)
3) Conditional blocks were optimized (enhanced switch statements used to replace if-else blocks when available)
4) Complicated nested for-loops were optimized (enhanced for-loops used when available)
5) Duplicated code fragments were removed (static methods used to extract and replace duplicated segments)
6) Redundant conditional booleans were removed
7) Syntax errors fixed
8) Unused commented code was removed

A sample duplicate code extraction is shown below for class LocationController.java:



```
-    @Override
-    public void initialize(URL location, ResourceBundle resources) {
-        //Populate userLocationInput ComboBox
-        ArrayList<String> rooms = new ArrayList<String>();
-        for (int row = 0; row < 4; row++) {
-        for (int col = 0; col < 4; col++) {
-            if (!roomGrid.get(row).get(col).roomName.equals("Unnamed")) {
-                rooms.add(roomGrid.get(row).get(col).roomName);
-            }
-        }
-    }
-        userLocationInput.getItems().addAll(rooms);
-        userLocationInput.getItems().add("None");

-        //Populate peopleLocationInput ComboBox
-        peopleLocationInput.getItems().addAll(rooms);

-    }
```

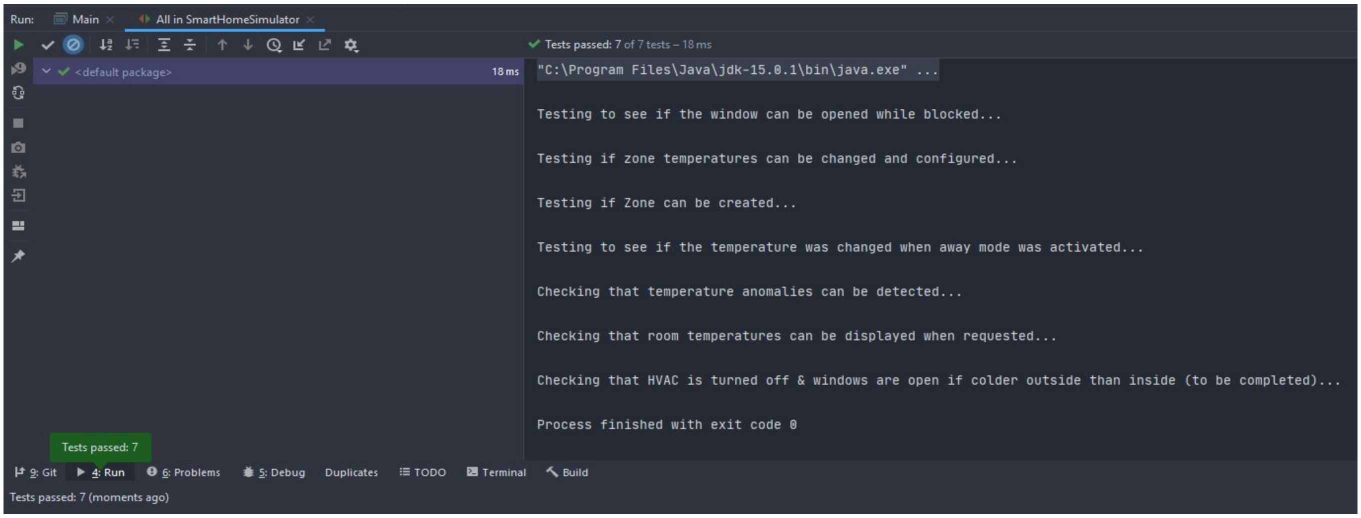
```
+    @Override
+    public void initialize(URL location, ResourceBundle resources) {
+        //Populate userLocationInput ComboBox
+        ArrayList<String> rooms = new ArrayList<String>();
+        injectRooms(rooms, roomGrid);
+        userLocationInput.getItems().addAll(rooms);
+        userLocationInput.getItems().add("None");
+
+        //Populate peopleLocationInput ComboBox
+        peopleLocationInput.getItems().addAll(rooms);

+    }

+    static void injectRooms(ArrayList<String> rooms, ArrayList<ArrayList<Room>> roomGrid) {
+        for (int row = 0; row < 4; row++) {
+            for (int col = 0; col < 4; col++) {
+                if (!roomGrid.get(row).get(col).roomName.equals("Unnamed")) {
+                    rooms.add(roomGrid.get(row).get(col).roomName);
+                }
+            }
+        }
+    }
```

Figure 3: *Refactoring duplicated code fragments for LocationController.java*

The commit id made for these changes is ***3bd8656*** for issue ***#43***.

Summary:

The above-mentioned refactoring changes resulted in a program that was more efficient, less cluttered, and easier to understand both for developers and for end-users. Unit Tests were run to ensure proper function of the program after refactoring changes were made. These tests are found in SmartHomeSimulator/343Proj/tests and will be demonstrated during the project demonstration. A photo showing the tests passing is shown below:



Tests passed: 7 of 7 tests – 18 ms

```
"C:\Program Files\Java\jdk-15.0.1\bin\java.exe" ...

Testing to see if the window can be opened while blocked...

Testing if zone temperatures can be changed and configured...

Testing if Zone can be created...

Testing to see if the temperature was changed when away mode was activated...

Checking that temperature anomalies can be detected...

Checking that room temperatures can be displayed when requested...

Checking that HVAC is turned off & windows are open if colder outside than inside (to be completed)...

Process finished with exit code 0
```

Tests passed: 7

Tests passed: 7 (moments ago)

Figure 4: *Unit Tests passing after refactoring.*

Use Cases

UC1: Create a Zone

Use case name	UC1: Create a Zone
Level	User-level
Brief description	With exceptions, rooms are often set to the same temperature. Zones are used to group rooms together. By adding rooms to a zone, they will all be set to the same zone temperature unless manually overridden.
Preconditions	There exists at least one created room in the house. The SSH tab is open inside of the system dashboard.
Triggering event	The user selects the Zone and presses the “Modify Zone” button.
Main flow	1. A “Modify Zone” window opens for the selected zone. 2.
Extensions	During step 2 of the main flow, the User enters a zone name that already exists: 1. An alert box appears informing the User of the error. 2. The User dismisses the alert box. 3. Continue from step 2 of the main flow. During step 3 of the main flow, the User doesn’t select any rooms: 1. An alert box appears informing the User of the error. 2. The user dismisses the alert box. 3. The “Create a Zone” window closes without creating a Zone.
Postconditions	The Zone is created and is shown in the SSH tab. The selected rooms are grouped inside of the SSH tab.

UC2: Manage Zone Temperature

Use case name	UC2: Manage Zone Temperature
Level	User-level
Brief description	Zones contain groups of room that are all kept at the same temperature, known as the zone temperature. The User must be able to manage this zone temperature.
Preconditions	There exists a zone containing one or more rooms inside. The SSH tab is open inside of the system dashboard.
Triggering event	The user presses the “Manage Zone” button.
Main flow	1. A “Manage Zone” window appears for the selected zone. 2. The User enters temperature values for the third suggested periods during the day and submits the information. 3. The “Manage Zone” window closes. 4. The automated HVAC system adjusts the temperature appropriately.
Extensions	During step 2 of the main flow the user enters invalid temperature values: 1. An alert window appears informing the user that they entered illegal temperature values. 2. The alert window closes. 3. Continue from step 2 of the main flow.
Postconditions	The zone temperature values have been changed. The HVAC has adjusted (or begun adjusting) according to the new values.

UC3: Display a Room Temperature

Use case name	UC3: Display a Room Temperature
Level	User-level
Brief description	Temperatures of individual rooms must be displayed so that the User can decide if it should be decreased, increased, or maintained. Temperature values should be displayed graphically in the dashboard.
Preconditions	The User is on the dashboard. The house layout file has been processed.
Triggering event	The User presses on the room square in the house layout graphic.
Main flow	1. A window appears with information about the selected room, including the room temperature. 2. The window is closed.
Extensions	During step 1 of the main flow, the user presses on a square that doesn't contain a room: 1. An alert box appears informing the user that there does not exist a room in the selected grid square. 2. Continue from step 2 of the main flow.
Postconditions	N./A.

UC4: Manage heating during summer

Use case name	UC4: Manage heating during summer
Level	User-level
Brief description	The air conditioning must be deactivated when the exterior temperature is lower than the interior temperature and a window should be opened. This creates an efficient way to cool the house without using energy.
Preconditions	The current season is summer.
Triggering event	The exterior temperature drops to a temperature lower than the interior.
Main flow	1. The system deactivates the HVAC unit in the room. 2. The Core module opens a window in the specified room. 3. The system waits for the room to reach the same temperature as the exterior. 4. The window is closed.
Extensions	During step 3 of the main flow, the room never reaches the target temperature: 1. The system closes the window. 2. The system activates the air conditioning unit in the room. 3. The system waits for the room to reach the target temperature. 4. The air conditioning unit is deactivated.
Postconditions	The room is at the target temperature. The room window is closed.

UC5: Manage Windows

Use case name	Manage Windows
Level	Sub-Function
Brief description	The System shall not open or close any windows if there is something preventing the window from opening or closing and will send a notification to the User to let them know that the system cannot complete the action.
Preconditions	The system is in a state where the state of a window cannot be changed, such as away mode.
Triggering event	The User attempts to open or close a window that cannot be.
Main flow	1. The system prevents the user from opening or closing the window by not changing the state of the window 2. The system notifies the User through an alert in the dashboard
Extensions	1. N./A.
Postconditions	The User is alerted to the fact that the state of the window cannot be changed.

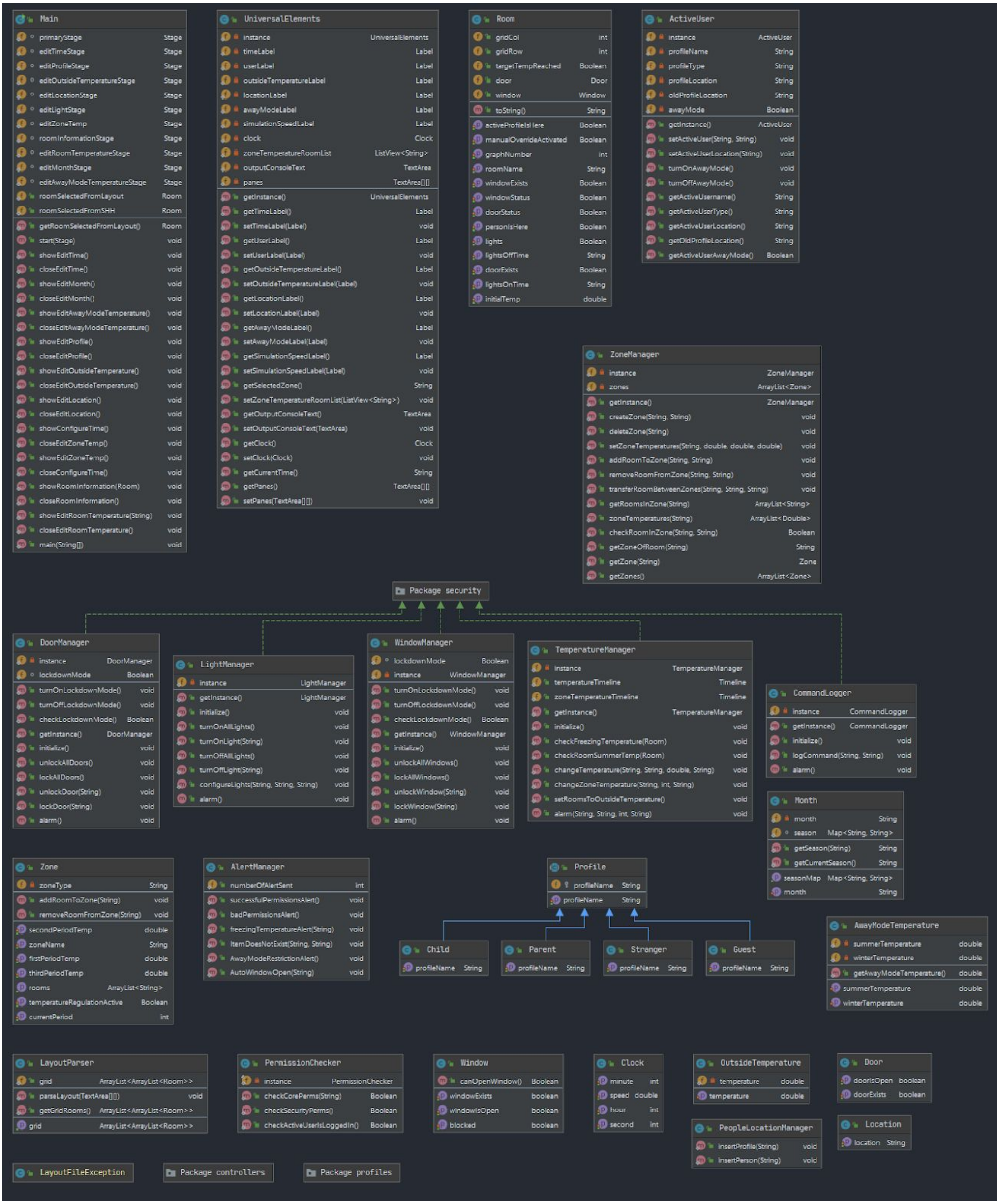
UC6: Automatic Away Mode based on User Location

Use case name	Automatic Away Mode based on User Location
Level	Subfunction
Brief description	Away Mode is used to manage the temperature of the home when the User is not present or when manually activated. The entirety of the house will be cooled to a specific temperature set by the User depending on the season.
Preconditions	The simulator has been started. There exists a user object in the house. The User has set the temperatures for Away Mode
Triggering event	The User leaves the house or activates away mode.
Main flow	1. The system detects that the User has left the house or that the user has manually activated Away Mode 2. The system selects the Away Mode temperature based on what season it is 3. The system cools the entire house the house the specified temperature
Extensions	During step 3 of the main flow, it is possible that the house needs to be heated instead of cooled during Away Mode as the temperature for Away Mode, set by the User, may be higher than the zone temperature
Postconditions	Each room in the house is cooled or heated to the specified Away Mode temperature

UC7: Alert User to Temperature Issue

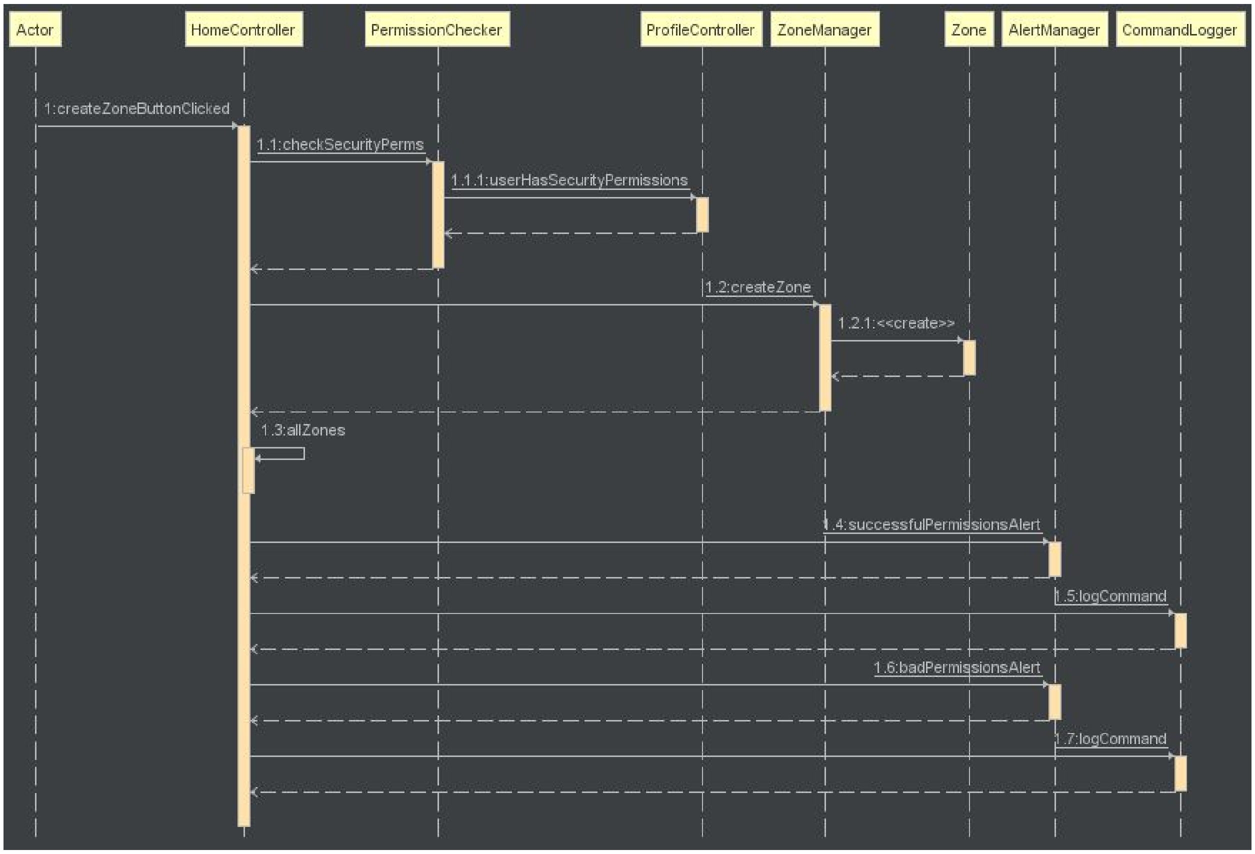
Use case name	Alert User to Temperature Issue
Level	Sub-Function
Brief description	The System (SHH) will send an alert to the user if there is something unusual with the temperature in the home. This will notify the User if the temperature inside is low enough that pipes could freeze and burst.
Preconditions	N./A.
Triggering event	The temperature reaches 0 degrees or lower.
Main flow	1. The system monitors the heat and notices a 0 or below temperature 2. The system notifies the User through an alert in the dashboard
Extensions	N./A.
Postconditions	The User is alerted to the unusual temperature and can decide on how to fix this issue if they need to.

Class Diagram

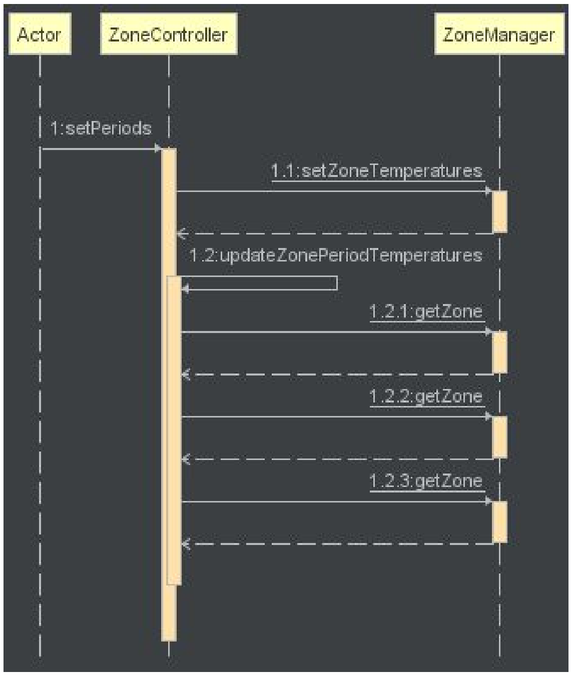

Main
primaryStage Stage
editTimeStage Stage
editProfileStage Stage
editOutsideTemperatureStage Stage
editLocationStage Stage
editLightStage Stage
editZoneTemp Stage
roomInformationStage Stage
editRoomTemperatureStage Stage
editMonthStage Stage
editAwayModeTemperatureStage Stage
roomSelectedFromLayout Room
roomSelectedFromSHH Room
getRoomSelectedFromLayout() Room
start(Stage) void
showEditTime() void
closeEditTime() void
showEditMonth() void
closeEditMonth() void
showEditAwayModeTemperature() void
closeEditAwayModeTemperature() void
showEditProfile() void
closeEditProfile() void
showEditOutsideTemperature() void
closeEditOutsideTemperature() void
showEditLocation() void
closeEditLocation() void
showConfigureTime() void
closeEditZoneTemp() void
showEditZoneTemp() void
closeConfigureTime() void
showRoomInformation(Room) void
closeRoomInformation() void
showEditRoomTemperature(String) void
closeEditRoomTemperature() void
main(String[]) void
UniversalElements
instance UniversalElements
timeLabel Label
userLabel Label
outsideTemperatureLabel Label
locationLabel Label
awayModeLabel Label
simulationSpeedLabel Label
clock Clock
zoneTemperatureRoomList ListView<String>
outputConsoleText TextArea
panes TextArea[][]
getInstance() UniversalElements
getTimeLabel() Label
setTimeLabel(Label) void
getUserLabel() Label
setUserLabel(Label) void
getOutsideTemperatureLabel() Label
setOutsideTemperatureLabel(Label) void
getLocationLabel() Label
setLocationLabel(Label) void
getAwayModeLabel() Label
setAwayModeLabel(Label) void
getSimulationSpeedLabel() Label
setSimulationSpeedLabel(Label) void
getSelectedZone() String
setZoneTemperatureRoomList(ListView<String>) void
getOutputConsoleText() TextArea
setOutputConsoleText(TextArea) void
getClock() Clock
setClock(Clock) void
getCurrentTime() String
getPanes() TextArea[][]
setPanes(TextArea[][]) void
Room
gridCol int
gridRow int
targetTempReached Boolean
door Door
window Window
toString() String
activeProfileIsHere Boolean
manualOverrideActivated Boolean
graphNumber int
roomName String
windowExists Boolean
windowStatus Boolean
doorStatus Boolean
personIsHere Boolean
lights Boolean
lightsOffTime String
doorExists Boolean
lightsOnTime String
initialTemp double
ActiveUser
instance ActiveUser
profileName String
profileType String
profileLocation String
oldProfileLocation String
awayMode Boolean
getInstance() ActiveUser
setActiveUser(String, String) void
setActiveUserLocation(String) void
turnOnAwayMode() void
turnOffAwayMode() void
getActiveUsername() String
getActiveUserType() String
getActiveUserLocation() String
getOldProfileLocation() String
getActiveUserAwayMode() Boolean
ZoneManager
instance ZoneManager
zones ArrayList<Zone>
getInstance() ZoneManager
createZone(String, String) void
deleteZone(String) void
setZoneTemperatures(String, double, double, double) void
addRoomToZone(String, String) void
removeRoomFromZone(String, String) void
transferRoomBetweenZones(String, String, String) void
getRoomsInZone(String) ArrayList<String>
zoneTemperatures(String) ArrayList<Double>
checkRoomInZone(String, String) Boolean
getZoneOfRoom(String) String
getZone(String) Zone
getZones() ArrayList<Zone>
Package security
DoorManager
instance DoorManager
lockdownMode Boolean
turnOnLockdownMode() void
turnOffLockdownMode() void
checkLockdownMode() Boolean
getInstance() DoorManager
initialize() void
unlockAllDoors() void
lockAllDoors() void
unlockDoor(String) void
lockDoor(String) void
alarm() void
LightManager
instance LightManager
getInstance() LightManager
initialize() void
turnOnAllLights() void
turnOnLight(String) void
turnOffAllLights() void
turnOffLight(String) void
configureLights(String, String, String) void
alarm() void
WindowManager
lockdownMode Boolean
instance WindowManager
turnOnLockdownMode() void
turnOffLockdownMode() void
checkLockdownMode() Boolean
getInstance() WindowManager
initialize() void
unlockAllWindows() void
lockAllWindows() void
unlockWindow(String) void
lockWindow(String) void
alarm() void
TemperatureManager
instance TemperatureManager
temperatureTimeline Timeline
zoneTemperatureTimeline Timeline
getInstance() TemperatureManager
initialize() void
checkFreezingTemperature(Room) void
checkRoomSummerTemp(Room) void
changeTemperature(String, String, double, String) void
changeZoneTemperature(String, int, String) void
setRoomsToOutsideTemperature() void
alarm(String, String, int, String) void
CommandLogger
instance CommandLogger
getInstance() CommandLogger
initialize() void
logCommand(String, String) void
alarm() void
Month
month String
season Map<String, String>
getSeason(String) String
getCurrentSeason() String
seasonMap Map<String, String>
month String
Zone
zoneType String
addRoomToZone(String) void
removeRoomFromZone(String) void
secondPeriodTemp double
zoneName String
firstPeriodTemp double
thirdPeriodTemp double
rooms ArrayList<String>
temperatureRegulationActive Boolean
currentPeriod int
AlertManager
numberOfAlertSent int
successfulPermissionsAlert() void
badPermissionsAlert() void
freezingTemperatureAlert(String) void
ItemDoesNotExist(String, String) void
AwayModeRestrictionAlert() void
AutoWindowOpen(String) void
Profile
profileName String
profileName String
Child
profileName String
Parent
profileName String
Stranger
profileName String
Guest
profileName String
AwayModeTemperature
summerTemperature double
winterTemperature double
getAwayModeTemperature() double
summerTemperature double
winterTemperature double
LayoutParser
grid ArrayList<ArrayList<Room>>
parseLayout(TextArea[][]) void
getGridRooms() ArrayList<ArrayList<Room>>
grid ArrayList<ArrayList<Room>>
PermissionChecker
instance PermissionChecker
checkCorePerms(String) Boolean
checkSecurityPerms() Boolean
checkActiveUserIsLoggedIn() Boolean
Window
canOpenWindow() Boolean
windowExists boolean
windowIsOpen boolean
blocked boolean
Clock
minute int
speed double
hour int
second int
OutsideTemperature
temperature double
temperature double
Door
doorIsOpen boolean
doorExists boolean
PeopleLocationManager
insertProfile(String) void
insertPerson(String) void
Location
location String
LayoutFileException
Package controllers
Package profiles

Sequence Diagram (One for each Use Case)

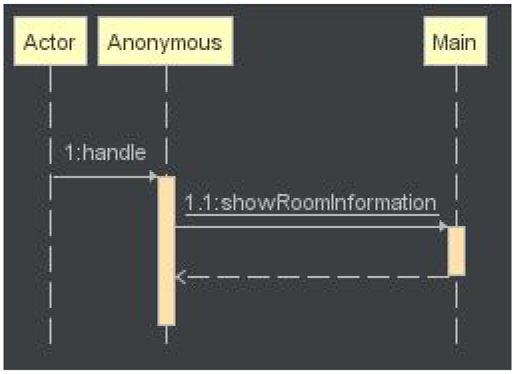
UC1 - Create a zone



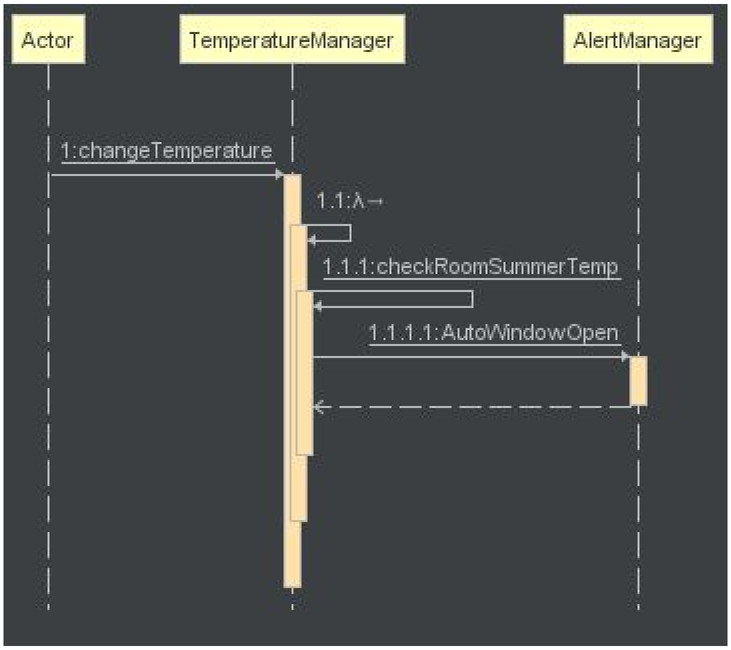
UC2 - Manage zone temperature



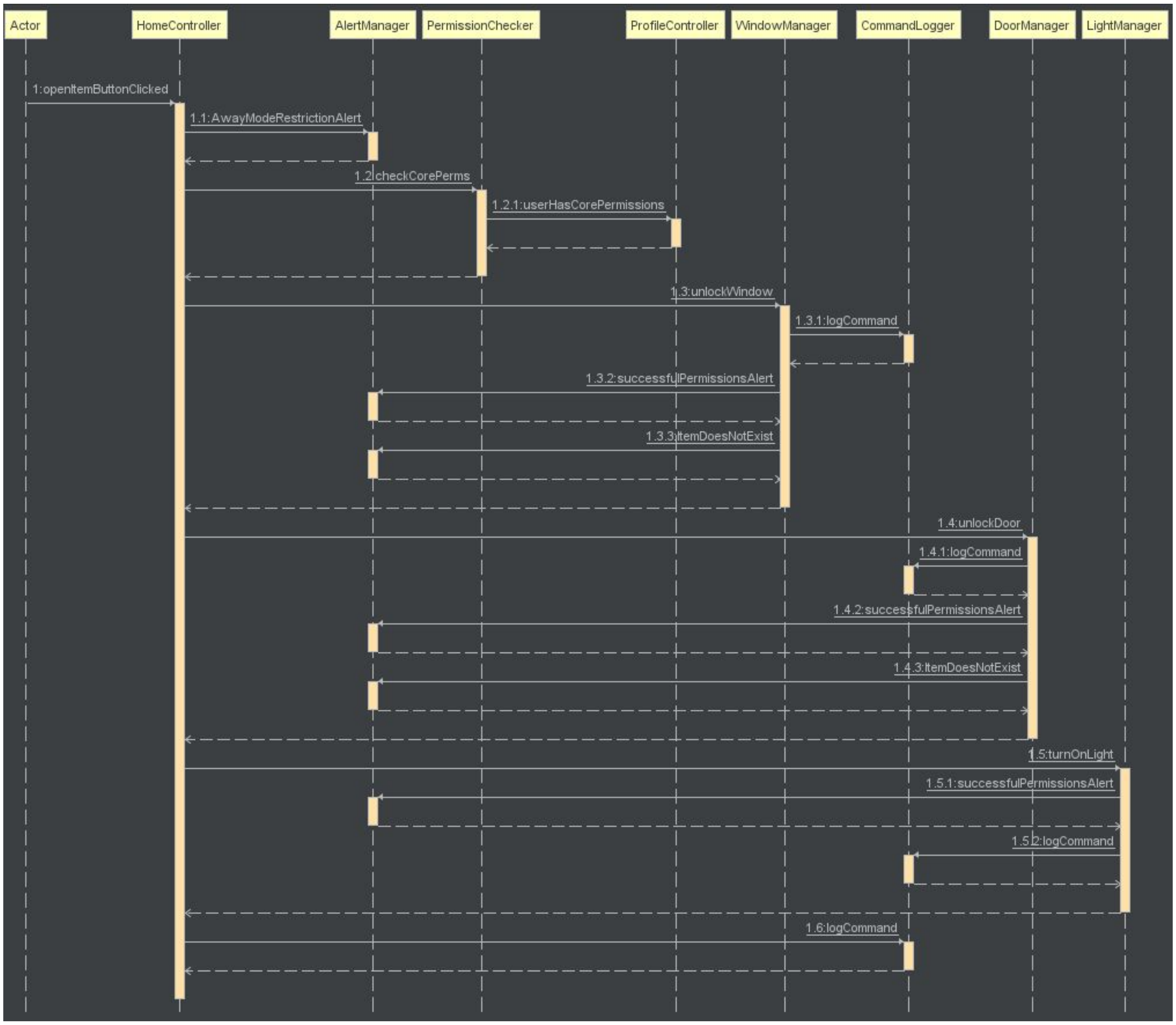
UC3 - Display a room’s temperature

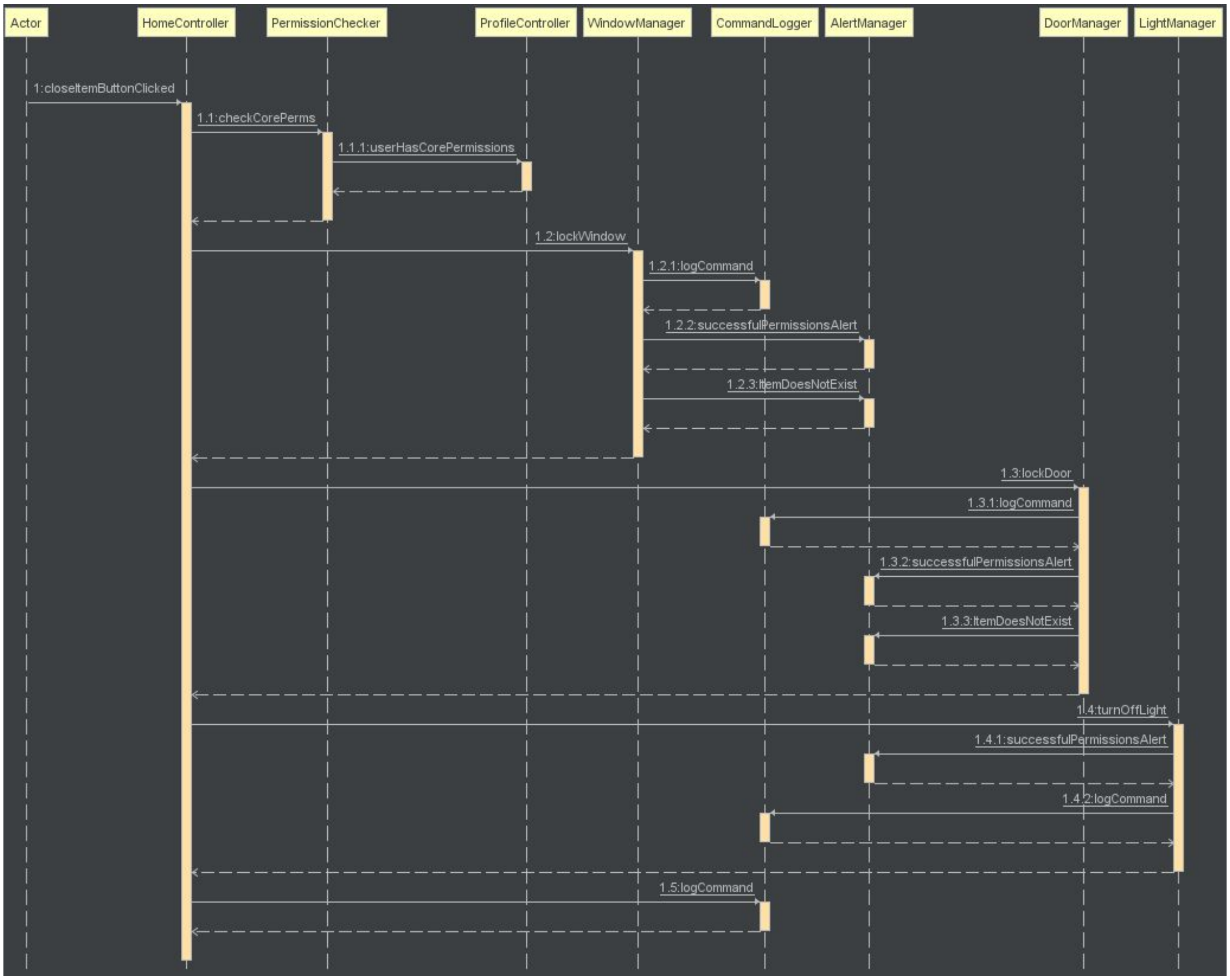


UC4 - Manage heat during Summer

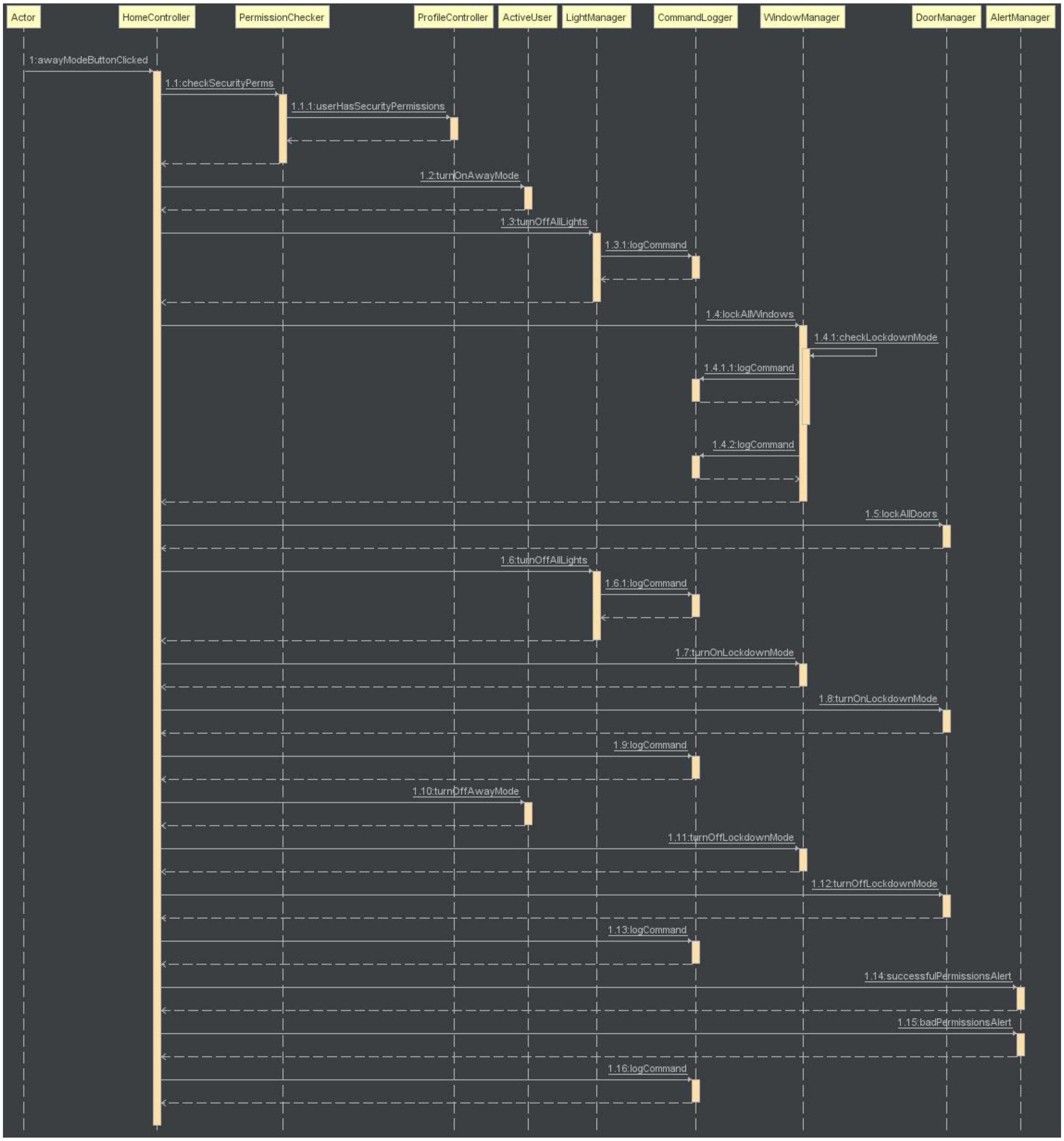


UC5 - Manage windows

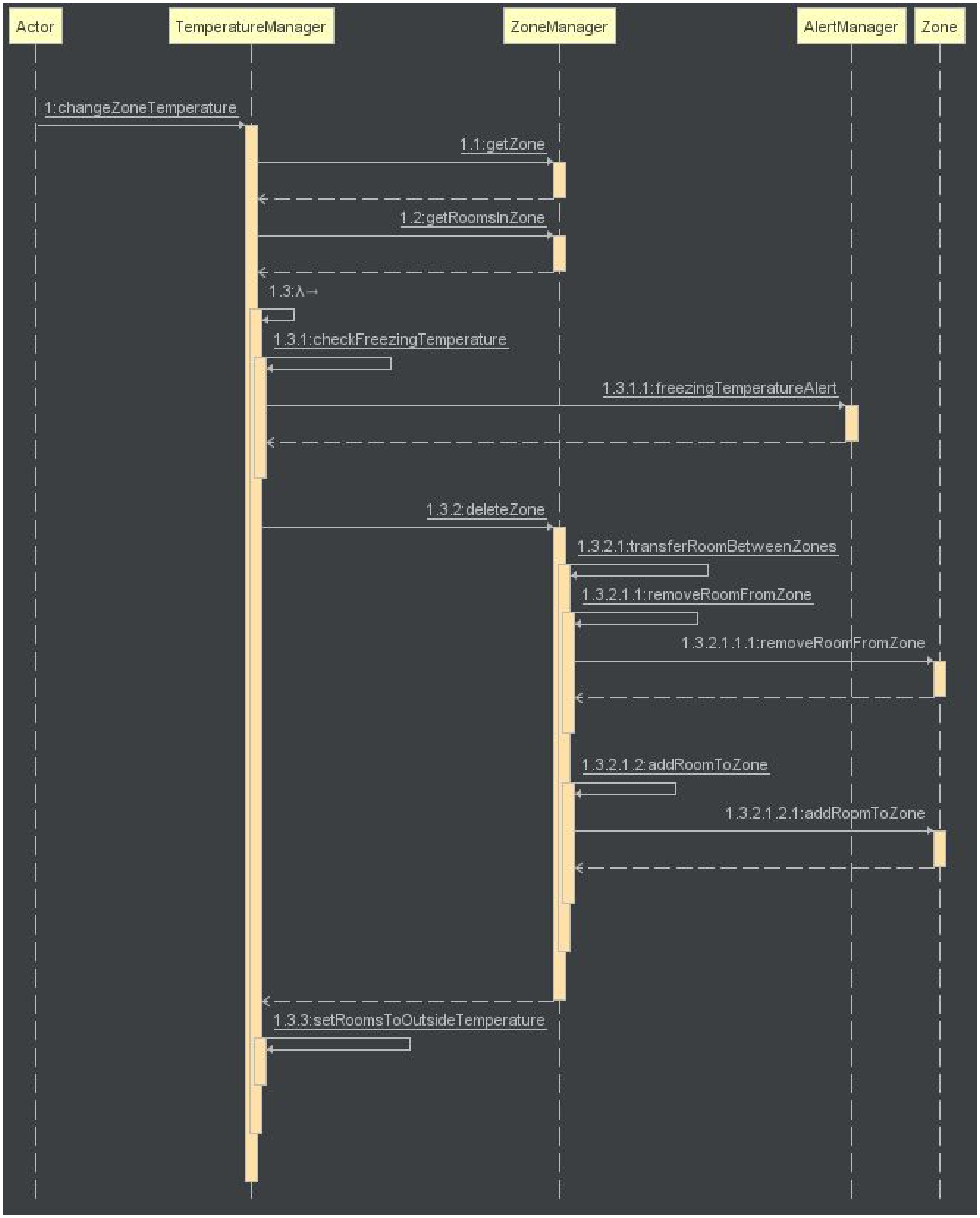



Actor
HomeController
PermissionChecker
ProfileController
WindowManager
CommandLogger
AlertManager
DoorManager
LightManager
1:closeItemButtonClicked
1.1:checkCorePerms
1.1.1:userHasCorePermissions
1.2:lockWindow
1.2.1:logCommand
1.2.2:successfulPermissionsAlert
1.2.3:ItemDoesNotExist
1.3:lockDoor
1.3.1:logCommand
1.3.2:successfulPermissionsAlert
1.3.3:ItemDoesNotExist
1.4:turnOffLight
1.4.1:successfulPermissionsAlert
1.4.2:logCommand
1.5:logCommand

UC6 - Away Mode activation

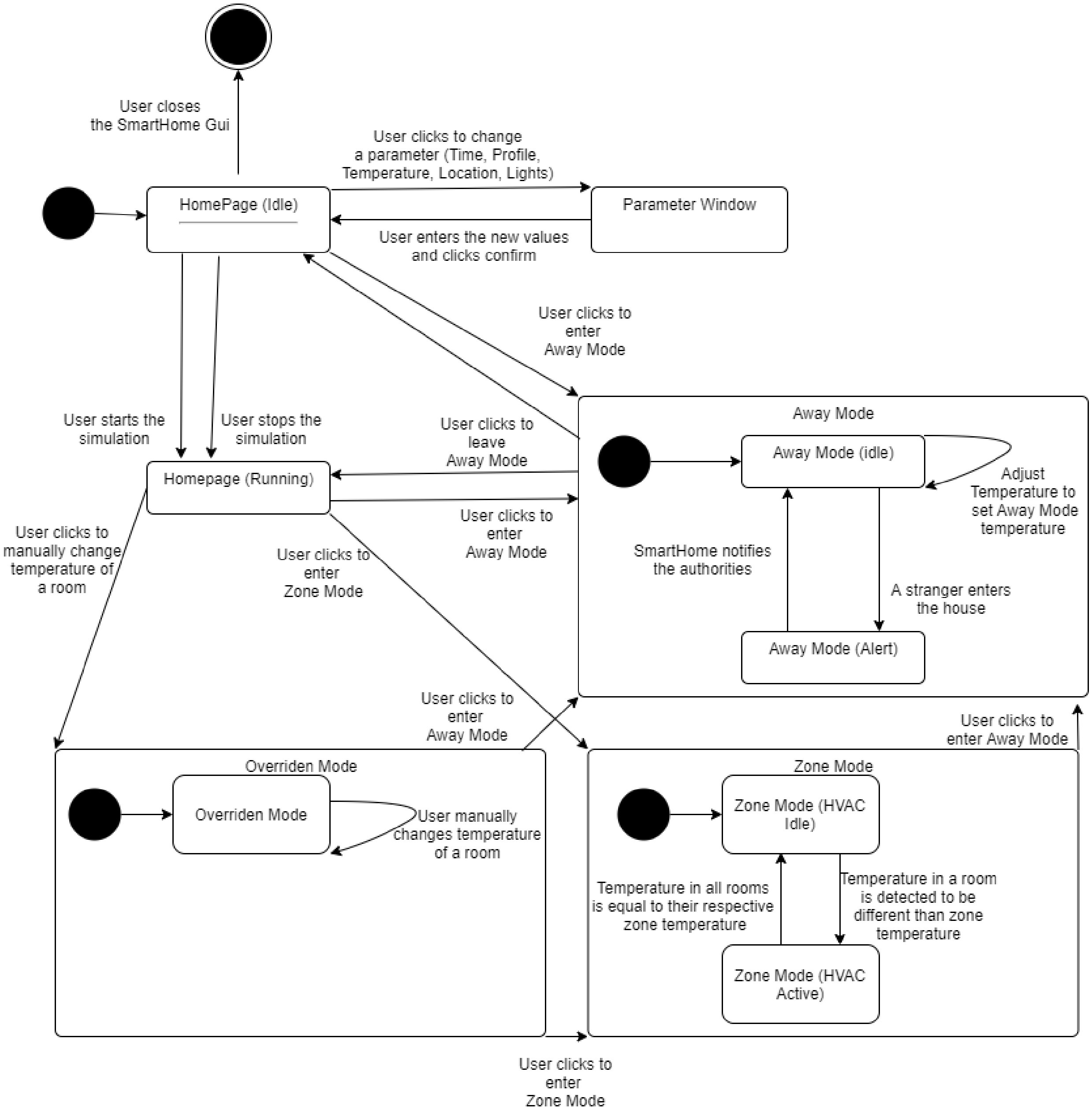


UC7 - Alert user to temperature issue

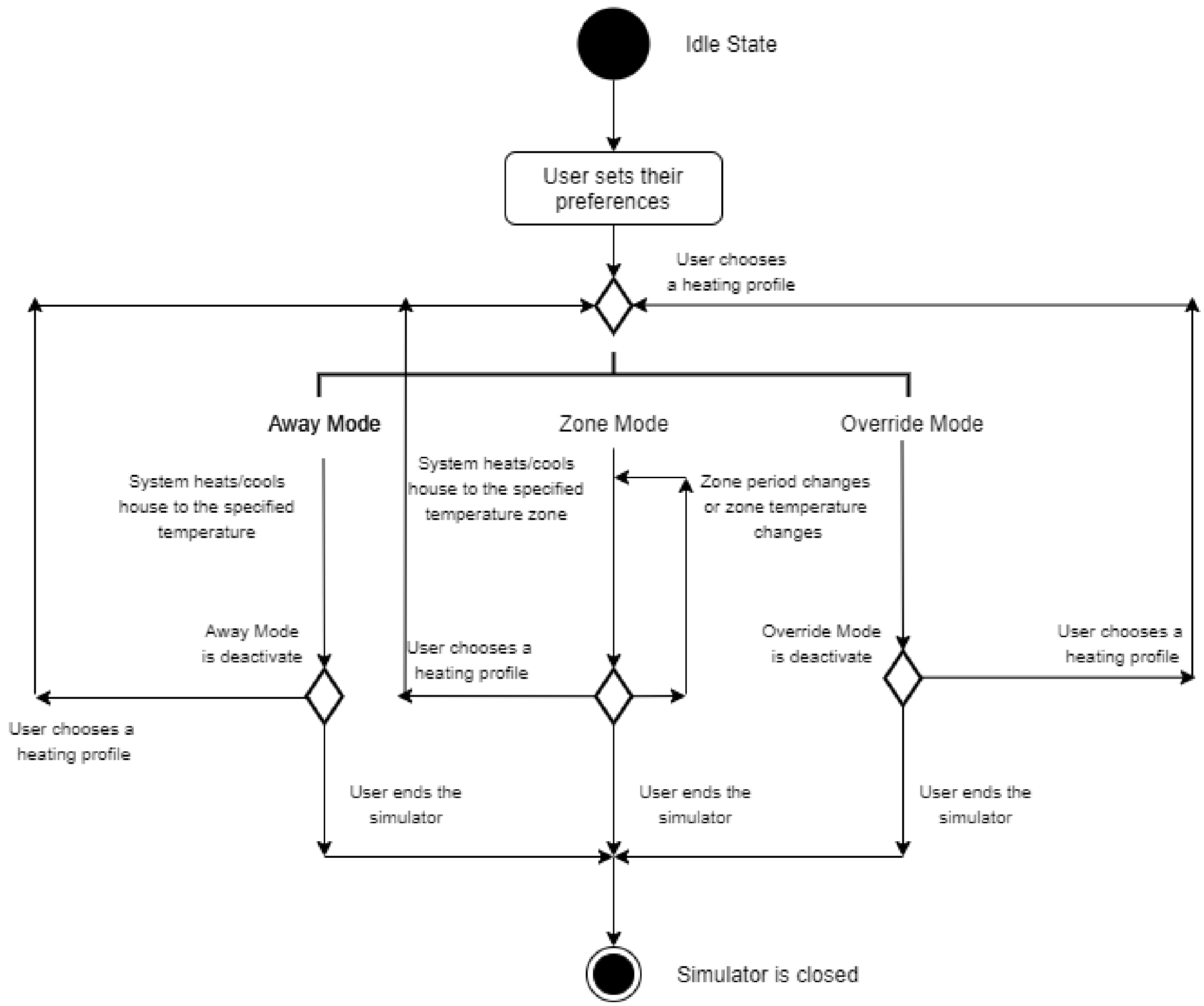


Actor
TemperatureManager
ZoneManager
AlertManager
Zone
1:changeZoneTemperature
1.1:getZone
1.2:getRoomsInZone
1.3:λ→
1.3.1:checkFreezingTemperature
1.3.1.1:freezingTemperatureAlert
1.3.2:deleteZone
1.3.2.1:transferRoomBetweenZones
1.3.2.1.1:removeRoomFromZone
1.3.2.1.1.1:removeRoomFromZone
1.3.2.1.2:addRoomToZone
1.3.2.1.2.1:addRoomToZone
1.3.3:setRoomsToOutsideTemperature

State Machine Diagram



Activity Diagram


Idle State
User sets their preferences
User chooses a heating profile
Away Mode
Zone Mode
Override Mode
System heats/cools house to the specified temperature
System heats/cools house to the specified temperature zone
Zone period changes or zone temperature changes
Away Mode is deactivate
User chooses a heating profile
Override Mode is deactivate
User chooses a heating profile
User chooses a heating profile
User ends the simulator
User ends the simulator
User ends the simulator
Simulator is closed