**Problem Description:**

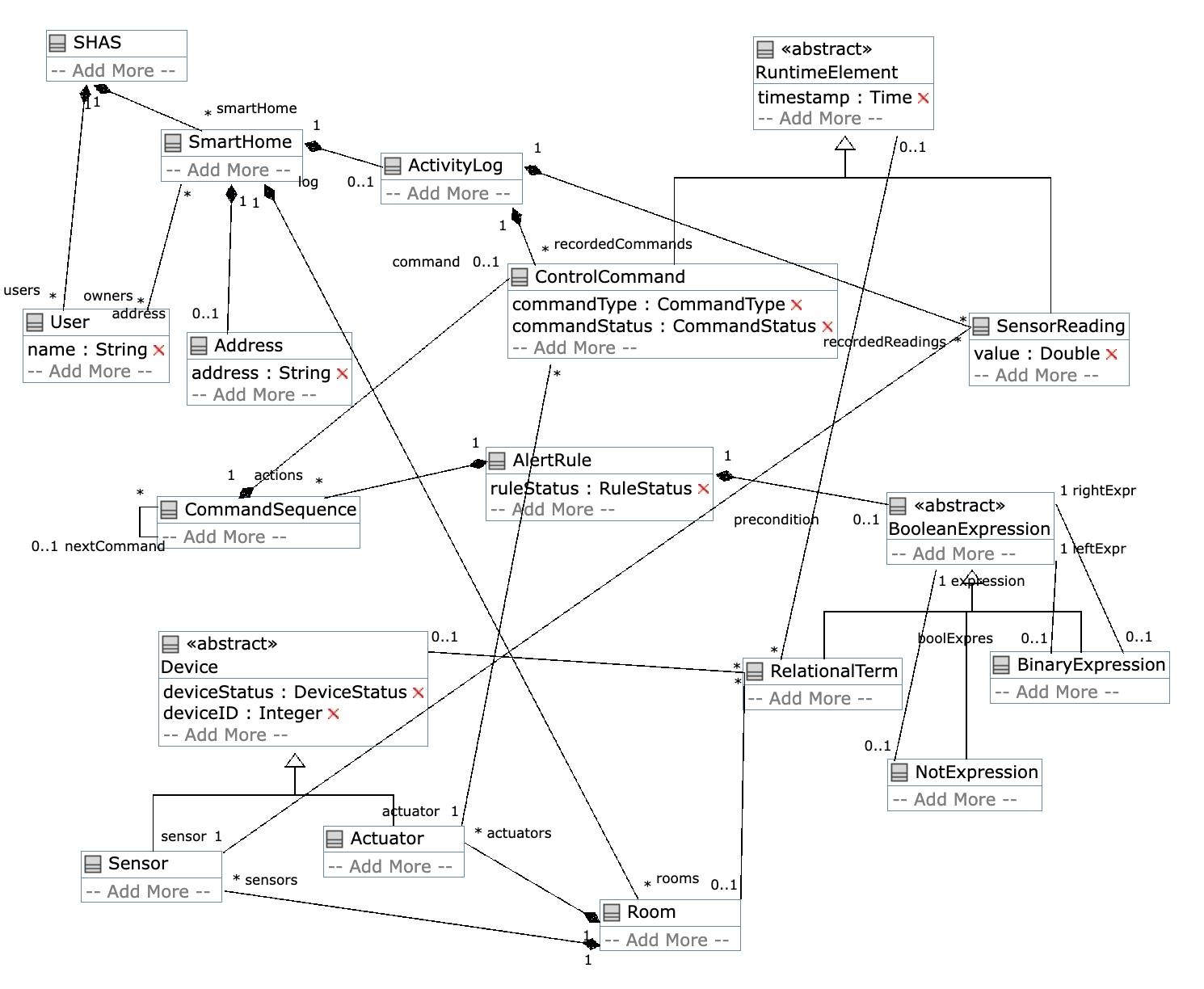
A smart home automation system (SHAS) offers various users to automatically manage smart home automation tasks. A smart home (located at a physical address) consists of several rooms, each of which may contain sensor devices and actuator (controller) devices of different types (e.g. temperature sensor, movement sensor, light controller, lock controller). Each sensor and actuator have a unique device identifier. Once a new sensor or actuator is activated or deactivated, SHAS will recognize the change and update its infrastructure map.

When SHAS is operational, a sensor device periodically provides sensor readings (recording the measured value and the timestamp). Similarly, a predefined set of control commands (e.g. *lockDoor*, *turnOnHeating*) can be sent to the actuator devices with the timestamp and the status of the command (e.g. requested, completed, failed, etc.). All sensor readings and control commands for a smart home are recorded by SHAS in an activity log.

Relevant alerts in a smart home can be set up and managed by its owner by setting up automation rules. An automation rule has a *precondition* and an *action*. The precondition is a Boolean expression constructed from *relational terms* connected by basic *Boolean operators* (AND, OR, NOT). Atomic relational terms may refer to rooms, sensors, actuators, sensor readings and control commands. The action is a sequence of control commands. For example, a sample rule could specify:  
*when* actualTemperature by Device #1244 in Living Room < 18 *and* window is closed  
*then* turnOnHeating in Living Room

Automation rules can be created, edited, activated and deactivated by owners. Only deactivated rules can be edited. Rules can also depend on or conflict with other rules, thus a complex rule hierarchy can be designed. SHAS records whenever an active rule was triggered using a timestamp.

**Reference Solution:**

****

**The detail is shown in the umple code:**

class SHAS {

1 <@>- 0..\* SmartHome smartHome;

1 <@>- \* User users;

}

class SmartHome {

1 <@>- 0..1 Address address;

1 <@>- 0..\* Room rooms;

1 <@>- 0..1 ActivityLog log;

\* -- \* User owners;}

class Address {

String address;

}

class User {

String name;

}

class Room {

1 <@>- \* Sensor sensors;

1 <@>- \* Actuator actuators;}

class Device {

abstract;

enum DeviceStatus { Activated, Deactivated };

DeviceStatus deviceStatus;

Integer deviceID;}

class ActivityLog {

1 <@>- \* SensorReading recordedReadings;

1 <@>- \* ControlCommand recordedCommands;}

class Sensor {

isA Device; }

class Actuator {

isA Device;}

class CommandSequence {

\* -> 0..1 CommandSequence nextCommand;

1 <@>- 0..1 ControlCommand command;}

class RuntimeElement {

abstract;

Time timestamp;}

class SensorReading {

isA RuntimeElement;

Double value;

\* -- 1 Sensor sensor;}

class ControlCommand {

isA RuntimeElement;

enum CommandType {lockDoor, turnOnHeating};

CommandType commandType;

enum CommandStatus {Requested, Completed, Failed};

CommandStatus commandStatus;

\* -- 1 Actuator actuator;}

class AlertRule {

enum RuleStatus {created, edited, activated, deactivated };

RuleStatus ruleStatus;

1 <@>- 0..1 BooleanExpression precondition;

1 <@>- \* CommandSequence actions;}

class BooleanExpression {

abstract;}

class RelationalTerm {

isA BooleanExpression;

\* -- 0..1 Room;

\* -- 0..1 Device;

\* -- 0..1 RuntimeElement;}

class NotExpression {

isA BooleanExpression;

0..1 -- 1 BooleanExpression expression;}

class BinaryExpression {

isA BooleanExpression;

enum BinaryOp {AND, OR };

0..1 boolExpres-- 1 BooleanExpression leftExpr;

0..1 -- 1 BooleanExpression rightExpr;

}