Project Details

My implementation code is included in the attached zip file. The HVAC.java class still contains the main method.The test cases are in the attached PDF. You will find several testing notes at the top of the document. As far as the implementation goes, there were several observations/notes that I had about the framework / state machine / requirements. These are included below:

1. The primary heating unit is never turned off when mode is changed from heating to off.

2. In the state machine, it specifies that the emergency heat should come on only based on time. Basically, after I turn the primary heat on, I should wait 5 minutes and then turn on the emergency heat for a 2 minute period (as long as I have not reached the desired temperature). Then, I repeat the cycle of turning off the backup heat, waiting 5 more minutes, and then leaving it on for another 2 minute period. In My simulation, I follow this behavior, but I set the times as 5 seconds, and 3 seconds, respectively. This makes the behavior easier to see and test. Of course, these values are completely configurable, and will can be changed without worrying about breaking the state machine logic.

3. There are 5 places in the system where I am enforcing thread safety. The main temperature control logic is synchronized so that the callback for the emergency heat timers cannot mess up My state machine calculations. This is implemented by using the synchronized keyword on the executeTemperatureControl() method. b.The code that runs when an emergency heat timer is finished waiting. The startEmergencyHeatTimerEvent() and stopEmergencyHeatTimerEvent() are synchronized for this reason. This prevents the main monitoring loop from interfering with the new system state. c.The producer/consumer queue for communicating mode change events to the from the CentralControl to the TemperatureControl. Fortunately, the BlockingQueue implementation is already thread safe, so there is nothing more I need to do for this. d.Updates to the GUI must be done on the EDT, so I added a couple of helper methods to enforce this by putting My update logic inside a runnable block that I pass to SwingUtilities.invokeLater(). The current temperature and desired temperature values are read/written from different threads. More specifically, the GUI updates the desired temperature, and the TemperatureSensor updates the current temperature. The existing framework code already contains the synchronization for this.

4. In this system, the user can switch between heating and cooling modes. The state machine dictates that the user must be in the “Off” mode to enter the “Heating” or “Cooling” mode. However, for the purposes of usability, I allow the users to switch between the modes freely. In My implementation however, the transition between “Heating” and “Cooling” (or vice versa) dictates that I transition to the off mode before starting the new mode. This simplifies My logic and makes My state machine more likely to be correct. In addition, it makes sense, because any components that are active in the original mode will be inactive in the opposite mode, so I don’t lose anything by just turning everything off.

5. The SystemState is determined whenever I turn the heating / cooling devices on or off. This basically works, but there is a bug that occurs whenever I turn the backup heat off and the SystemState is also set to OFF. If I was transitioning from BackupHeat -> PrimaryHeat (because the timer expired), then I need the SystemState to remain as HEATING. However, I do not have enough information to determine that information in this project. SystemState is only used by the TempSensor to assign a new temperature once per second. Therefore, even if this component is not functioning as expected, it will not impact the correctness of My state machine implementation. In the TempSensor class, the random number generation seemed to be broken for a couple of the cases, so I tweaked it slightly to make it work as expected. The “isOn” variable needs to get set in the device classes (PrimaryHeat, BackupHeat, Fan, and Cooling).