Metropolitan State University, St. Paul, MN
ICS 372 Object-Oriented Design and Implementation
Group Project 3

1 Due Dates and Goals

1.1 Due:

11:59 PM on May 4, 2019

1.2 Goals:

1. Perform finite-state modeling techniques to discover and specify the conceptual classes.
2. Use design principles to translate conceptual class design into an appropriate set of abstract and concrete classes and interfaces
3. Efficiently develop systems using design patterns including State and Observer
4. Use principles of the agile methodology by following the Unified Process
5. Use the Unified Modeling Language to document work
6. Implement a design utilizing structures such as classes and interfaces,
7. Work in small groups
8. Employ Java coding standards.

2 The Problem

Your task is to develop a finite state machine and a GUI interface to create a temperature controller. The controller regulates the temperature of a room. The user can set the desired room temperature and have the controller operate a device of choice to make the room temperature reach a desired value.

1. The temperature control system has three devices. A fan, an air conditioner, and a heater.
2. The user may ask the controller to operate one of the devices or none at all. The GUI allows the user to select the device (or no device).
3. The GUI allows the user to set the desired room temperature.
4. For purposes of testing, the GUI also allows the user to set (change) the current room temperature and the current outside temperature.

5. If no device is selected, the room temperature changes in the direction of the outside temperature at the rate of 1 degree per minute. As an example, if the current room temperature is 70 degrees and the outside temperature is 90 degrees and no device is in operation, the room temperature would reach the outside temperature in 20 minutes. That is, if the outside temperature remains at 90; It could change, of course, and then the time needed for the room temperature to reach the outside temperature would change accordingly.

6. If the fan is selected, it remains idle for 10 minutes and then blows air for 10 minutes. It then goes back to idle, staying in the idle state for 10 minutes, and the cycle repeats. If the fan is on, the room temperature moves in the direction of the outside temperature as in Item 5 above. Put in other words, the fan has no effect on the room temperature.

7. If the air conditioner is selected, it starts off idling, but the system immediately checks if the current room temperature is at least 3 degrees more than the desired room temperature. If so, the air conditioner springs into action and works until the desired room temperature is equal to the current room temperature, at which time it goes idle. The air conditioner is capable of cooling at a rate of 2 degree per minute if the outside temperature had no effect at all, but the outside temperature would at the same time drag the current room temperature in the opposite direction at the rate of 1 degree per minute. Once the air conditioner has brought the room temperature to the desired temperature, as long as it remains selected, the system ensures that the current room temperature is never above the desired room temperature by more than 3 degrees (with slight margins of error because of timing issues).

8. If the heater is selected, it starts off idling, but the system immediately checks if the current room temperature is at least 3 degrees less than the desired room temperature. If so, the heater starts heating and works until the desired room temperature is equal to the current room temperature, at which time it goes idle. The heater is capable of heating at a rate of 2 degree per minute if the outside temperature had no effect at all, but the outside temperature would at the same time drag the current room temperature in the opposite direction at the rate of 1 degree per minute. Once the heater has brought the room temperature to the desired temperature, as long as it remains selected, the system ensures that the current room temperature is never below the desired room temperature by more than 3 degrees (with slight margins of error because of timing issues).

3 What You Need to Do and Submit

You have to understand the problem well and then model, design, and implement in Java FX, a working solution to the problem.

1. Model the system as a finite state machine and come up with a minimized state transition diagram. Draw it as per UML standards clearly labeling the states, events, and all of the state transitions. Submit the state transition table.

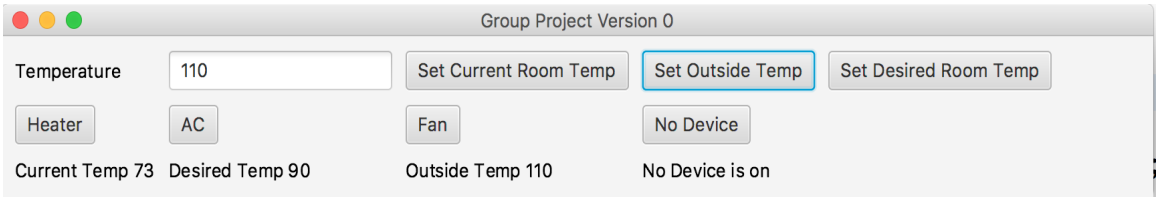
2. Design all the classes and interfaces to correctly implement the requirements. The implementation must follow the model you submit. It must have the minimum number of conditionals. The way the current room temperature changes in the direction of the outside temperature must be completely programmed independent of the effect of the air conditioner or the heater. In particular, don't compute and use the net cooling rate or heating rate.

3. If a device is currently selected, the GUI must show what it is doing and keep the display updated every second. If no device is selected, show that information.

4. Change the unit of time for the rate of increase and decrease of room temperature to seconds. This reduces the time needed for testing. For example, the program you submit must change the current temperature 1 degree per second, assuming that no device is selected. Similar requirements apply for the case when a device is selected.

5. The GUI must closely follow the layout given in the sample video and the picture below.

6. The program source files must be organized into packages in a clear and logical manner. There must be a package named start that contains a file named `Main.java` that can be used to start the program.



Submit a single zip file that contains the following.

1. A PDF file that contains the state transition table.

2. The JavaFX Eclipse project (with all the source files) organized as packages.

4 Grading

The project will be graded as below.

4.1 State Transition Table

The diagram will be graded as per the following rubric.

Criteria	Points
Correct States	10
Proper Organization	4
Correct Events	10
Correct Transitions	10

4.2 Implementation

The code will be graded as per the following rubric.

Criteria	**Points**
Proper documentation and coding conventions	10
Proper packaging and easiness of execution	10
A GUI Interface that is similar to the one given in this handout	10
Correctness	16
Proper structuring of code classes, methods, and code	20

Every class must have the name of the members of the group or the author's name and document the purpose of the class at the very top before the class header. All methods other than overrides and getters and setters must document the purpose of the method and any parameters and any return values.

The classes, methods, and code must be structured so it reflects proper use of the state and observer patterns.

If you don't minimize the number of conditionals, you will lose credit for correctness of execution and code structuring. That is, it is not enough to get correct behavior in a non-object-oriented way. Every instance of a selection that could have been avoided would be penalized. There are some selections that cannot be avoided: this includes things like deciding what to do based on the values of two temperatures.

The grade for correctness would be affected if you don't change the time unit for rate of change of temperature to seconds or use the wrong numerical values.

Improper submissions will be penalized 10 percent of the grade.