ICS 372 Object-Oriented Design and Implementation
Group Project 2 Iteration 1&2

Due: 04 August 2016
Points: 100

The Problem

Consider a refrigerator with two parts. (The definitions may not be fully accepted in conversation, but this is the terminology we use.)

Fridge: A unit where the temperature is kept between 37 and 41 degree Fahrenheit.
Freezer: A unit where the temperature is kept between 0 and -9 degree Fahrenheit.

Independent compressors and related components work to maintain the desired temperatures in the respective units.

We will use the Fahrenheit scale for all temperatures.

The user may use a control to keep the temperatures of the two units at any one of the temperatures available. For example, the user may choose to keep the fridge temperature at 38 degrees and the freezer temperature at -3 degrees.

The outside temperature (the temperature of the room where the refrigerator is kept) may vary between 50 degrees and 110 degrees.

Assume that we have the following scenario. The user sets the fridge temperature at 39 degrees and the freezer temperature at -4 degrees. Suppose those temperatures have been attained. Let the outside temperature be 70 degrees. Since the fridge and the freezer are at the desired temperatures, they are not actively cooled at the moment. Although there is insulation, the fact that the room temperature is much higher than the inside temperatures of the two units, the freezer and the fridge temperatures increase and when they become too high, the cooling units (compressor , etc.) start working again.

There are independent lights within the two units that go on when the respective units are opened.

When a unit is opened, it is exposed to the room and the temperature rises more rapidly than it would if the door were closed.

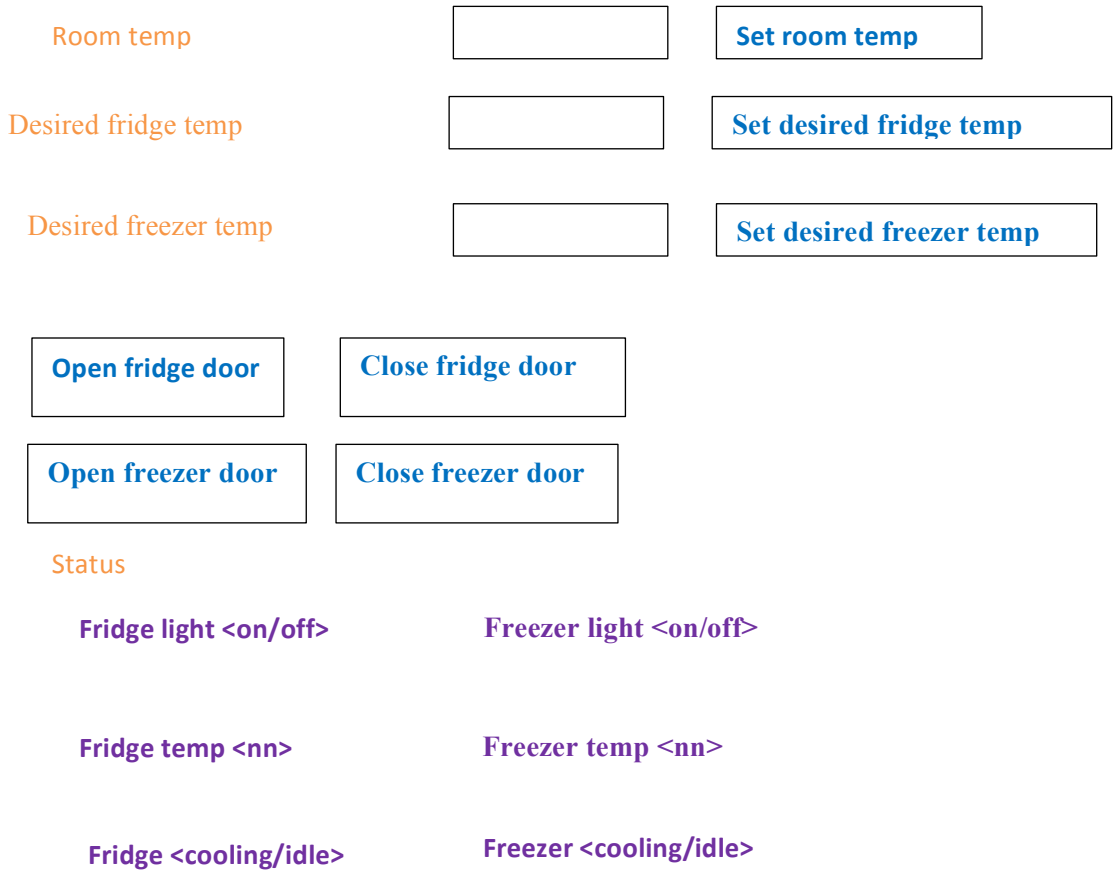
Your task in the first pass is to model the refrigerator and its two units, the fridge and the freezer.

When the temperature within a unit is 1 degree above the desired temperature, the cooling system becomes active.

The System

Provide a graphical user interface. The controls are illustrated below. **Text in orange** denote **JLabel**s that never change. **Rectangles that surround text in blue** are **JButton** objects that can be

clicked to signify input to the program. Empty rectangles are JTextField objects for the user to enter information. Finally, **text in purple** are **JLabel** objects that represent the status of the system; the angled brackets denote what can change; for example, freezer light could be on or off, the



Configuration File: The numbers highlighted in yellow are shown as representative examples only. The actual parameters must be read from a *pure text file* that contains values for each of the following. (*The numbers must be in the following order, one per line, and the file name must be a parameter to the main method.*)

1) The lowest settable temperature for the fridge
2) The highest settable temperature for the fridge
3) The lowest settable temperature for the freezer
4) The highest settable temperature for the freezer
5) The lowest possible room temperature
6) The highest possible room temperature
7) The time in minutes needed for the fridge to rise by 1 degree when the cooling system is not active and the door is closed
8) The time in minutes needed for the fridge to rise by 1 degree when the cooling system is not active and the door is open
9) The time in minutes needed for the freezer to rise by 1 degree when the cooling system is not active and the door is closed

10) The time in minutes needed for the freezer to rise by 1 degree when the cooling system is not active and the door is open
11) The difference between the fridge temperature and its desired temperature for the cooling system in the fridge to become active
12) The difference between the freezer temperature and its desired temperature for the cooling system in the freezer to become active
13) The number of minutes needed for the fridge to cool by 1 degree.
14) The number of minutes needed for the freezer to cool by 1 degree.

You must use the values from the configuration file appropriately in the program. For example, if items 1 and 2 are 30 and 40 degrees respectively, you must refuse to accept temperature settings for the freezer that are less than 30 or greater than 40.

The following could be a configuration file.

FridgeLow=37
FridgeHigh=41
FreezerLow=-9
FreezerHigh=0
RoomLow=50
RoomHigh=75
FridgeRateLossDoorClosed=30
FridgeRateLossDoorOpen=2
FreezerRateLossDoorClosed=10
FreezerRateLossDoorOpen=1
FridgeCompressorStartDiff=2
FreezerCompressorStartDiff=1
FridgeCoolRate=20
FreezerCoolRate=30

If needed, you may assume that the minimum room temperature is always more than the desired temperatures of the fridge and the freezer.

Miscellaneous:

1) Initially, set the freezer and fridge temperatures the same as the outside temperature.
2) To make it possible to test the program in a reasonable amount of time, transform time periods in minutes to seconds. That is, let the clock generate a tick every second, rather than every minute.

Organization:

Develop the minimized state transition table and diagram. In this iteration, have at least the following classes:

a) GUI as described above
b) The logic to switch between the states
c) A thread to keep track of time

Submission

Submit the following:

1. A single Word or PDF file that contains the state transition diagram, state table, class and sequence diagrams
2. The source files for all classes

The above files should be zipped and submitted.

Grading

Criterion	Points
State transition diagram	6
State table	6
Class diagrams	8
Sequence diagrams	10
Use of a proper parameters file as specified and the ability to specify it as a parameter to the main method	6
Graphical User Interface: friendliness, appearance, correctness	6
Correctness implementation	40
Documentation and Readability	18
Total	100

Important:

If you don't use a parameters file, you are highly likely to lose lots of points for correctness because I want to test your program with my parameters.