CSE 5322: SOFTWARE DESIGN PATTERNS

April 5, 2023

Homework 4, 15%

Due: 04/25/2023 11:59PM

1 Introduction

This individual homework requires the student to apply the state and observer patterns to design, and implement in Java, a simplistic lawn mower emulator. The mower can mow a rectangular lawn by cutting the grass one row after another. The mower stops when all of the rows are cut. When the application is launched, it shows a window with two buttons (Start and Stop) and a lawn. When the Start button is clicked, the mower begins to cut the lawn, resulting in changes of the color of the lawn area as shown in the following figure:

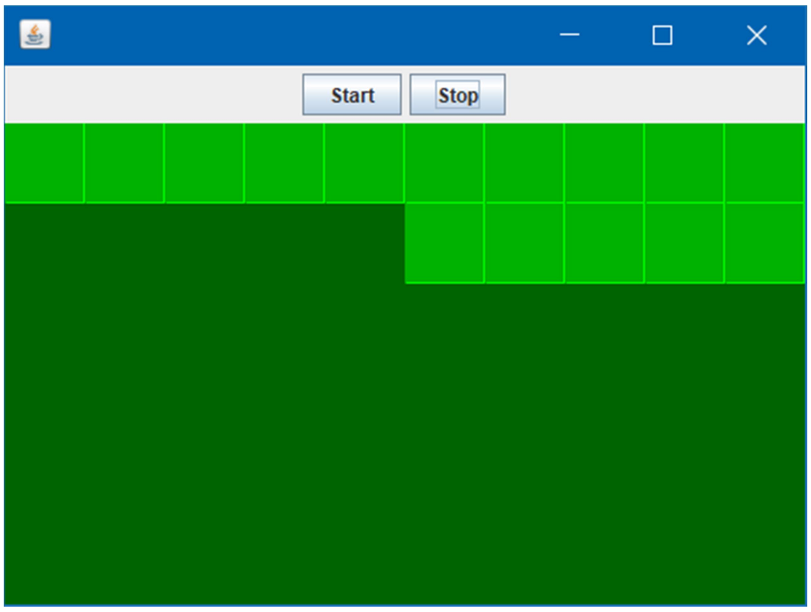

Figure 1: A lawn-mower simulator is cutting a lawn

The mower consists of a motor, a grass cutter, a sensor (e.g., a camera or a photoelectric sensor), a steering hardware for making left and right turns as well as control software. The motor moves the mower forward at a constant speed such as 2 feet per second and drives the cutter to cut the grass underneath.

Conceptually, you can view the lawn as consisting of n by m squares or an n x m array, denoted by A. You can assume that each square is 2 feet by 2 feet, approximately the size of a residential lawn mower. You can assume that the mower always starts cutting A[0,0] and moves from left to right (or west to east) on even rows 0, 2, 4, 6, ... and from right to left (or east to west) on odd rows 1, 3, 5, 7, ... Figure 2 illustrates the movements of the lawn mower.

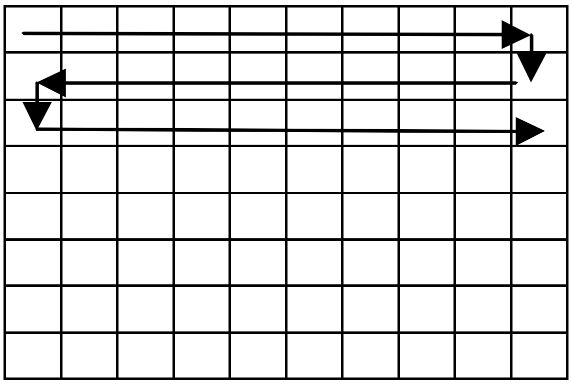

Figure 2: Movements of the lawn mower

The control software uses a timer (e.g., javax.swing.Timer) to generate an action event every second — implying that the mower has traveled a distance of 2 feet. When the control software receives the action event, it checks with the sensor to see if the right, left or bottom edge of the lawn is reached. If the right edge is reached, the software directs the steering hardware to make a right turn, advance 2 feet and make another right turn to cut the next row in the opposite direction. If the left edge is reached, the software directs the steering hardware to make a left turn, advance 2 feet and make another left turn. If the bottom edge is reached, the software directs the mower to turn off.

You can view the mower as moving toward three directions: left, right and down, or alternatively east, west and south. You can use integers i=0, 1, 2, ..., and j=0, 1, 2, ... to keep track of the location of the mower. For example, when the mower moves east on the first row, each action event will increment j and the software will check with the sensor to see if the right edge is reached. When the mower is moving west, then each action event will decrease j. When the mower makes a left or right turn, the software also checks if the bottom edge is reached before it attempts to advance to the next row (i.e., i=i+1). You can use a function, say edgeReached():boolean, to simulate checking with the sensor. The implementation of this simulated function should be very easy:

```
public boolean edgeReached() {
   return j == 0 || // left edge reached
          j == m-1 || // right edge reached
          i == n-1; // bottom edge reached
}
```

You can think of the mower has four different states, east, west, southeast (i.e., south after a right turn), and southwest (i.e., south after a left turn). You may also think that it also has an initial state. Depending on the current state of the mower, an action event will cause different actions to take place, for example, increment j as well as making a state transition (possibly remain in the same state). Such behavior can be described by using a state transition machine/diagram. This homework requires the student to produce the state transition diagram that accurately and adequately describe the behavior of this simplistic lawn mower. In addition, this homework requires the student to apply the state pattern and the observer pattern to produce a design for the state behavior.

2 What to Do and Submit

This homework requires the student to do the following (equal weights):

1. Produce a state machine to describe the state-dependent behavior of the control software of the lawn mower.
2. Convert the state machine into a UML class diagram by applying the state pattern. The UML class diagram must show the classes, methods of the classes and relationships between the classes. Use UML notes to show how the methods of the classes will be implemented.
3. Produce a design of the application shown in Figure 1 that applies the state pattern and the observer pattern.

   Hint: you may need to use the controller pattern to handle the actor requests: start the mower and stop the mower. You also need to include a window that contains a drawing area representing the lawn. The drawing area may be the observer while the mower may be the observable. The mower notifies the observer whenever it advances a square.
4. Implement, compile and run your application and save screen shots showing the working of the lawn mower. See also the next section.

3 Implementation

1. The implementation must be in Java. You may use Swing or AWT, whichever you prefer. Provide comments in your code to show the pattern applied.
2. The implementation must include a graphical user interface (GUI) that shows the movements of the lawn mower at a reasonable speed (not too fast and not too slow, this can be done by adjusting the timer interrupt interval).
3. The design must correctly apply the state pattern and the observer pattern as well as implementing the patterns correctly. A partially correct implementation is considered incorrect. You may use/reuse the observable and observer APIs from Java; you don't need to implement these patterns. Of course, you need to implement the update method of the observer API.

   Note: an implementation that runs and produces the correct result may not correctly implement the patterns.
4. Compile and run your application. Produce and submit several consecutive screen shots to show the movements of the lawn mower.
5. Submit the complete source code and an executable jar file, which can be clicked to run your software. Make sure that your software can run on different machines, not just in your development environment.

4 How To Submit

Submit your analysis and design in one pdf or doc(x) file. Name your file as follows:

Lastname_Firstname_CSE5322_S23_HW4.pdf, or
Lastname_Firstname_CSE5322_S23_HW4.doc(x)

Compress your implementation files into one file and name it as follows:

Lastname_Firstname_CSE5322_S23_HW4_impl.rar, or
Lastname_Firstname_CSE5322_S23_HW4_impl.zip

Additional submission instructions may be given by the GTA.