CS 416 Program 2P
EMT Assignment in Swing/AWT

February 8, 2017

Important Dates and Requirements

- The program is due Wednesday, February 15 at 23:59. Late points (Th -3, Fri -7, Sa,Su -12, Mo -15)
- The last submission day is Monday, February 20 at 23:59

1. Scenario

This program simulates (very poorly) the potential traveling patterns of an emergency service vehicle. The program state will be initialized with locations of some number of hospitals in a region that can accept emergency trauma patients, as well as a "home" location for the emergency vehicle. The basic action of the emergency vehicle is to react to an emergency call (a mouse click in the drawing panel indicates the location of an emergency site), proceed immediately to the site at "emergency" speed, pause at the site for a two seconds (to simulate picking up the patient) and then travel at emergency speed to the nearest hospital location. Once it arrives at the hospital, it must wait a two seconds (to drop off the patient) and then drive back to its home location at normal speed. There are no "streets" for this travel; the display area represents an abstraction of the real world where the path from one location to another is always represented by a straight line (presumably the length of the line on the display is proportional to the true distance in the real world.)

A couple of additional events can occur, however:

1. A mouse event in the panel creates new emergency sites that must be maintained in a first-in/first-out ordering. Hence, when the vehicle gets to the hospital, it should not head home, but should go to the next site on the list – if there is one.
2. A mouse event on the symbol that represents an emergency site indicates that that site is no longer an emergency site and can be removed from the vehicle's target list. (Presumably, another vehicle has reached that site, or it has been determined that it was a false alarm.)
3. Emergency sites can be dragged to new locations; this simulates a situation where the location of the site in the initial report of an accident was incorrect. However, dragging is not allowed if the emergency vehicle is currently moving towards that site (i.e., dragging should be disabled for the "current" site).

2. The Program

The *EMT vehicle* class should define a composite *JComponent* object that represents the vehicle, which should be created at its *home* location (20, 20), be no larger than 50x50, implement the *Animated* interface, and have a public *travelTo* method. While traveling to an emergency site or a hospital, the program should draw a thin red line from the current position of the vehicle to its target location.

The starter code has 2 classes that are complete, *EMTApp.java* and *Hospital.java*. It has four partially implemented classes: *Dispatcher, EMTVehicle, EMTPanel* and *EmergencySite*. Read the comments in these files carefully. You are allowed to create other classes if you like.

3. Testing strategy

It is very hard to test interactive programs in a way that provides a clear record of the tests performed and validation that the tests succeeded. Even so, there are some basic features of most interactive programs that can be tested with a reproducible semi-automated strategy. For this program these features include the different speeds of the vehicle, and the number and locations of the hospitals. One mechanism for testing features such as these is to provide command line options that identify these parameters at startup time. At startup the main program reads the command line arguments that determine these parameters with the default values shown in parentheses:

hiSpeed (30): number of pixels the EMT vehicle should move in each frame while heading to an emergency site or to the hospital with a patient

normalSpeed (10): number of pixels the EMT vehicle should move in each frame while heading home

nSites (3): the number of hospitals

All arguments are optional, but they must come in the specified order, so you must specify *hiSpeed* if you want to change *normalSpeed*, both speeds if you want to specify *nSites* and all parameters to specify the *nSites*.

4. Notes

1. This assignment uses A-classes, J-classes, the *FrameTimer* class and the interfaces, *Animated* and *Draggable*. **None of these are part of the starter package.** They are all included in the course library, *cs416.jar*, which is available at *~cs416/public/cs416.jar*. See lab 4 instructions for getting the library and setting it up in DrJava.
2. Lab 3 serves as a reasonable starting point for this assignment. It provides the framework for using the AWT *Timer* class (via *FrameTimer*), the *Animated* interface, mouse handling, and the *JWheels* classes.
3. The *EMT vehicle* class should extend *JComponent* and should contain at least 2 *JWheels* objects.
4. The line to the next site location should always start at the current EMT vehicle location, not the previous site location.
5. An *EmergencySite* object should be colored red at creation and should be changed to blue when the patient has been picked up or when the site has been disabled by a mouse click.
6. *EMT vehicle* and *EmergencySite* classes have simple *main* methods that test these classes by themselves. Lab 3 (especially) should be used as a beginning pattern to fill in the starter classes.
7. There are a several design decisions that you need to make that do not have "obvious" solutions, including: Who (which class) should be responsible for drawing the tracking line? Who should "know" about the collection of emergency sites? How much should the emergency vehicle "know"? I have suggested that you create a *Dispatcher* class that is largely responsible for controlling the vehicle, but there are several sub-issues you will have to address. These are discussed in the comments in the code.
8. Make sure your code is well-modularized. In other words, keep methods short and create appropriate *private* methods so that code is not replicated.
9. Pay attention to the style conventions; we will grade your style with the *styleChecker* program.

Tentative point allocation

10 Display an EMT vehicle object. We'll test this by executing the *EMTvehicle.main* method.
10 Command line arguments are recognized and used correctly in the code
10 Display an emergency site object and be able to drag it. We'll test this by executing the *EmergencySite.main* method.
10 Create emergency sites with mouse presses; the site can be disabled by a mouse press on the site; these points can be earned if the site changes color on the mouse press.
10 EMT vehicle moves from start to first emergency site at some speed
10 EMT vehicle pauses at the emergency site and then heads to nearest hospital.
EMT vehicle pauses at hospital, then heads for home location at slower speed.
10 Tracking line from EMT vehicle to emergency site and hospital works correctly
10 Speed parameters affect the travel speed correctly
10 Multiple site locations can be generated and visited
10 All special cases handled; current site can't be dragged; trip home interrupted immediately if new emergency site generated, etc.

For all categories, full credit depends on having a "good" implementation and the expectation that your implementation has been well-tested.

Submission

This is assignment 2P. Submit only *EMTVehicle.java*, *EmergencySite.java, Dispatcher.java*, *EMTPanel.java* and any additional classes you create.

Don't forget our *Statement.txt* also!