COMPSCI 230: Assignment 3

In this assignment, you will create a Swing simulation in Java. This application will extend the application developed in Assignment 2 to simulate the ambulances picking up and delivering patients. This will give you hands-on experience working with Java in *Eclipse* to develop a Swing application.

This assignment is due on **Friday 21 October, 2016, 11:59pm** and is worth 5% of your final course marks.

Estimated time to complete this assignment is 15-20 hours.

This is version 3 of this specification – the changes are indicated in purple.

Background

The ambulance service for the city of Chromatropolis is in the process of updating their computer systems. One of their challenges is where the ambulances should be stationed in order to provide the best coverage. In assignment two you developed an application to retrieve and update ambulance and patient details.

In this assignment, you are developing an application that simulates ambulances picking up patients and delivering them to Chromatropolis Hospital. This application uses the data from the files from assignment two. It does **not** use the application itself.

Submission

To prepare your solution for submission, ensure it is working on the lab computers. Provide an Eclipse project with code files that executes straight away. Your code should not depend on any external packages, e.g. it needs to compile and execute with the standard JAVA SDK and JRE.

Submit via the Assignment Dropbox (). Create a single .zip file of the entire project directory and submit that **one** file via the Dropbox. Note:

- Ensure you submit for the correct course.
- If you resubmit, please include **all** files in the .zip file of your re-submission.

Requirements

The application will provide a simple simulation of ambulances picking up patients and delivering them to the local hospital. It visualises the simulation in a GUI window.

The data for this application comes from two CSV files called “ambulances.csv” and “patients.csv”. Your application will need to load and parse these files. The data uses the same format as in assignment 2. Sample ambulances.csv and patients.csv files are available on Canvas.

A) Simulation

The first task is to implement a simple simulation of the ambulances. This needs to be in a self-contained module so it can be tested easily. The module will read the ambulances.csv and patients.csv file, run for a set period and then output the updated information to ambulances-2.csv and patients-2.csv (both input and output files use the same data formats).

The simulation has five elements:

- The city: this is a grid with dimensions of 0 to 100 in both the x and y dimensions.
- The ambulances: these are loaded from ambulances.csv. Each ambulance has an (x,y) location and a status (see below).

- The patients: these are loaded from patients.csv. Each patient has an (x, y) location and a status (see below).
- The hospital: this exists on the grid at location (50, 50)
- The ambulance stations: there are three stations at different locations on the grid:
    - Greenfields – (10, 0)
    - Bluelane – (30, 80)
    - Redvill – (90, 20)

The initial positions of the ambulances and patients are read in from the data files. The locations of the hospital and stations remain unchanged.

Every second, the ambulances will update their status and position. The new status and position depends on the current status:

- 'At station': check if there is a new patient to pick up, if so, assign the closest unassigned patient to the ambulance and change the status to 'Responding'. Otherwise, do nothing.
- 'Responding': move the ambulance towards the assigned patient by four moves. If the ambulance reaches the patient, change the status to 'At scene'.
- 'At scene': if the ambulance has been at the scene for four seconds, change the status to 'Transporting'. Otherwise, do nothing.
- 'Transporting': move the ambulance towards the hospital by three moves. If the ambulance reaches the hospital, change the status to 'At destination'.
- 'At destination': if the ambulance has been at the hospital for two seconds, change the status to 'Returning'. Otherwise, do nothing.
- 'Returning': move the ambulance towards the nearest available station by three moves. If the ambulance reaches the station, change the status to 'At station'.

In addition, some of the ambulance changes also update the status of the patients:

- When an ambulance is assigned to a patient, the patient's status is changed to 'Assigned'.
- When the ambulance starts to transport a patient (i.e. the ambulance status changes to 'Transporting'), the patient's status is also changed to 'Transporting'.
- When the ambulance reaches the hospital, the patient's status is changed to 'Completed'. At this time, the patient is unassigned from the ambulance.

Note: the patient's position will never change, only their status and the assigned ambulance.

Extra rules:

- The closest distance is defined as the Euclidean (straight-line) distance between two points.
- A move is a one unit change in either the x-coordinate or the y-coordinate. Ambulances will always move to reduce the distance towards their target (patient, hospital or station).
- The number of ambulances a station can accommodate is defined by the total number of ambulances divided by three, rounded up. For example, if there are four ambulances defined, each station can accommodate two ambulances (four divided by three, rounded up).
- A station is available if there are less ambulances at the station than it can accommodate.

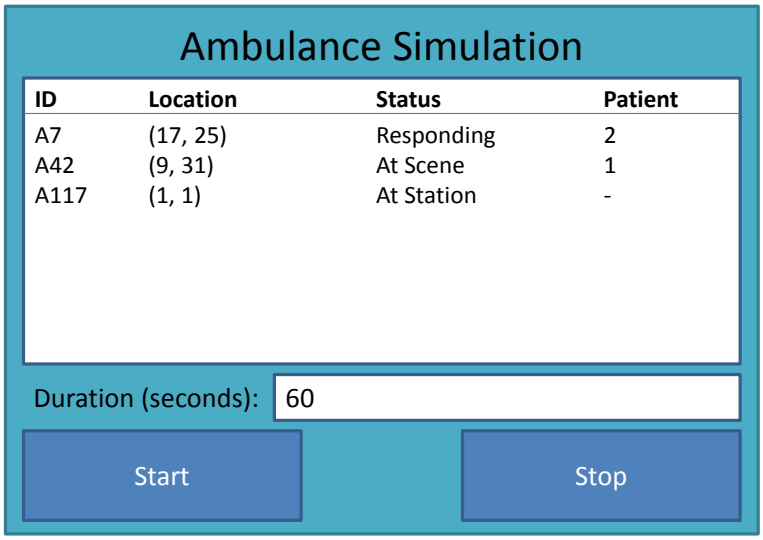
Each ambulance should have its own thread. These threads need to run independently of all other ambulances.

Think about which of the data structures need to be thread safe and implement accordingly.

The initial duration for the self-contained module is 60 seconds. This can be assumed constant for the first task, however the subsequent tasks it should be able to modify the duration.

B) GUI Interface

The second task is to implement a GUI that controls and displays the simulation:



The list displays the details of all the ambulances within the simulation. This should show a real-time display of where every ambulance is.

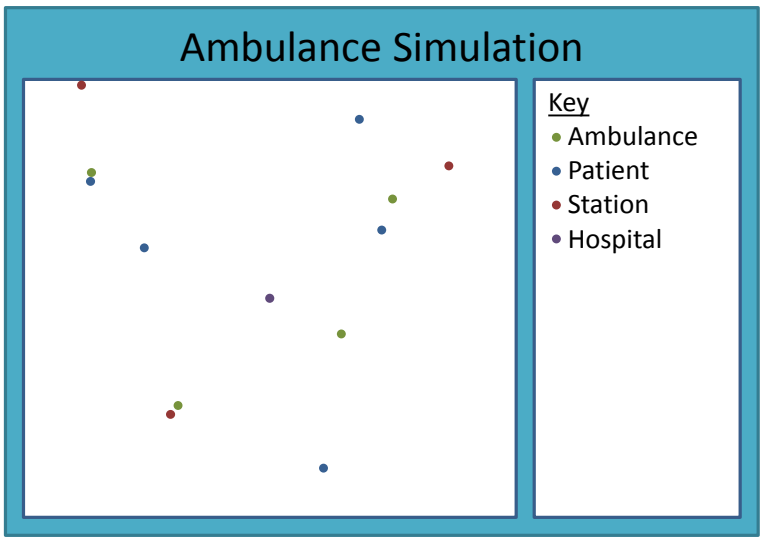
Remember, the simulation should run inside a background thread to prevent freezing the GUI. The GUI components should not be modified from the background thread directly to avoid unpredictable errors. Using SwingWorker from SwingUtilities is suggested to keep things simple.

The simulation is started by clicking on the 'Start' button and stopped by clicking on the 'Stop' button.

When the simulation is started, the 'Stop' button should be enabled and all other elements disabled. When the simulation is stopped, the 'Stop' button should be disabled and all other elements enabled.

C) Visualisation Interface

The final task is to implement a GUI to show the location of all the elements (ambulances, patients, stations and hospitals) in the simulation:



This should be displayed in a second frame, which is displayed at the same time as the GUI window. However the display only needs to be updated when the user clicks on the stop button.

Hint: see lecture 26 for an example of how to implement this screen.

Mark Scheme

The marker will be looking for the following:

Item	Points
Simulation implementation: • Correctly updates ambulance status • Correctly updates patient status • All rules implemented	 4 3 5
Multi-threaded implementation uses a thread per ambulance and synchronisation constructs for ensuring the data is not corrupted.	5
GUI interface displays the statuses of the ambulances and updates their details without freezing the UI.	4
Visualisation interface shows all the elements in the simulation and updates the positions when the user clicks on stop.	4
Total (maximum mark)	**25**

Penalties

Marks will be deducted from your total for the following:

Penalty	Description
-5	Error in the application – applies if your program throws a run-time exception from any normal input by the marker
Up to -25	Hard coding – if the marker finds that you've inappropriately entered data or case-specific responses directly into your Java code rather than reading/writing to the CSV files
??	Late submission – the lecturer may accept late submissions by a scheme of penalties announced only after the deadline has passed. Note that extensions without penalty are possible for medical or compassionate grounds (not including workload in other courses, normal employer requirement or elective travel), or through request of University Counselling Services. The dropbox will remain open for a substantial period after the initial deadline to allow receipt of late submissions.