

## Compsci 230: Assignment 2

This assignment will involve creating a Swing application in Java, using the skills you have learnt in this course. The assignment will involve simulating a Police communication system, managing the picking up and detaining of suspects. This will give you hands-on experience working with Java in *Eclipse* to develop a multi-threaded Swing application.

This assignment is due on **Friday 27 October, 2017, 11:59pm** and is worth 5% of your final course marks.

### Background

The police force for the city of Parallelopolis have been hassled about slow response times and require an update of their computer systems to help keep the peace. One of their major challenges is organising which police units should respond to suspects, and which police station suspects should be taken to.

In this assignment, you will be developing an application that simulates police units picking up a police dog, attending crime scenes, picking up suspects, and dropping them off at a police station within the city of Parallelopolis.

### Submission

To prepare your solution for submission, first ensure it is working on the lab computers. Provide an Eclipse project with code files that execute 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.

Task one and two should be in two modules that can be run separately for testing purposes. Please have the CSV files be input from and output to the root directory of the project. Include a small text file named `Readme.txt` with a brief description of your project and how to run it.

Submit via the Assignment Dropbox method. Create a single zip file of the entire project directory and submit **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 resubmission

### Requirements

This application will be split into two main tasks. The first task will provide a simulation of police units picking up suspects and delivering them to the local police station. The second task will require you to visualise this simulation in a GUI windows.

The data input for this simulation will come from two CSV files, “`police.csv`” and “`suspects.csv`”. Your application will need to load these files in and parse them accordingly. Samples of these files will be available on Canvas, and a more detailed description of their contents is below.

### Task A) Simulation

The first task is to implement a simulation of the police units’ activity. This needs to be in a self-contained module, so it can be tested easily and separately from the second task. This module will read the “`police.csv`” and “`suspects.csv`”, run for a set period of time, then output the updated information to “`police-output.csv`” and “`suspects-output.csv`”. The output files will use the same format as the input file format.

The Simulation has the following elements:

- City of Parallelopolis: This is represented by a grid with dimensions of 0 to 100 in both the x and y dimensions.
- The Police Units: These police units are loaded in through the “police.csv” file. Each police unit has an ID number, an (x,y) location, a current status, a Yes/No field for if it has a police dog and an assigned suspect ID. Police units who haven’t been assigned a suspect will have an empty suspect ID.
- The Suspects: The suspects are loaded in through the “suspects.csv” file. Each suspect has an ID number, an (x,y) location, a current status and an assigned police unit ID. Suspects who haven’t yet been assigned a police unit will have an empty police unit ID.
- The Kennel: This holds a set of police dogs at location (50,50) in the city grid.
- The Police Stations: There are 4 stations in Parallelopolis at these locations:
  - Downtown – (25,5)
  - Midtown – (80,30)
  - Uptown – (10,90)
  - Lazytown – (70,80)

The position of the Kennel and the Police Stations will remain constant, while the initial positions of Police Units and Suspects will be read in from the data files.

Every second, the police units will update their status and position. Their actions will be determined based on their current status. The explanation for each police unit status is shown below:

- **‘Standby’**: Check if there are any suspects to pick up. Assign the closest unassigned suspect to this police unit. Change the police unit status to ‘Approaching Kennel’. If there are no available suspects, do nothing.
- **‘Approaching Kennel’**: Move the police unit towards the Kennel by 3 moves (see restrictions). If the police unit reaches the Kennel, change the status to ‘At Kennel’
- **‘At Kennel’**: If the police unit is collecting a police dog, remove one dog from the kennel and assign it to the police unit then change the status to ‘Approaching Suspect’. If the police unit is returning a police dog, unassign it from the police unit, return it to the kennel and change status to ‘Returning’.
- **‘Approaching Suspect’**: Move the police unit towards the assigned suspect by 4 moves. If the police unit reaches the suspect, change the status to ‘At Scene’
- **‘At Scene’**: If the police unit has been at the scene for four seconds, change the status to ‘Approaching Kennel’. Otherwise do nothing.
- **‘Returning’**: Move the police unit towards the nearest available station (see restrictions) by 3 moves. If the police unit reaches the station, change the status to ‘Standby’

Some of the police unit actions also change the status of their assigned suspect:

- When a police unit is assigned a suspect, the suspect’s status is changed to ‘Assigned’
- When the police unit reaches the suspect, the suspect’s status is changed to ‘Caught’
- When the police unit reaches the station, with the caught suspect, the suspect’s status is changed to ‘Jailed’. The suspect is then unassigned from the police unit.

Restrictions:

- The closest distance is defined as the Euclidean (straight-line) distance between two points.

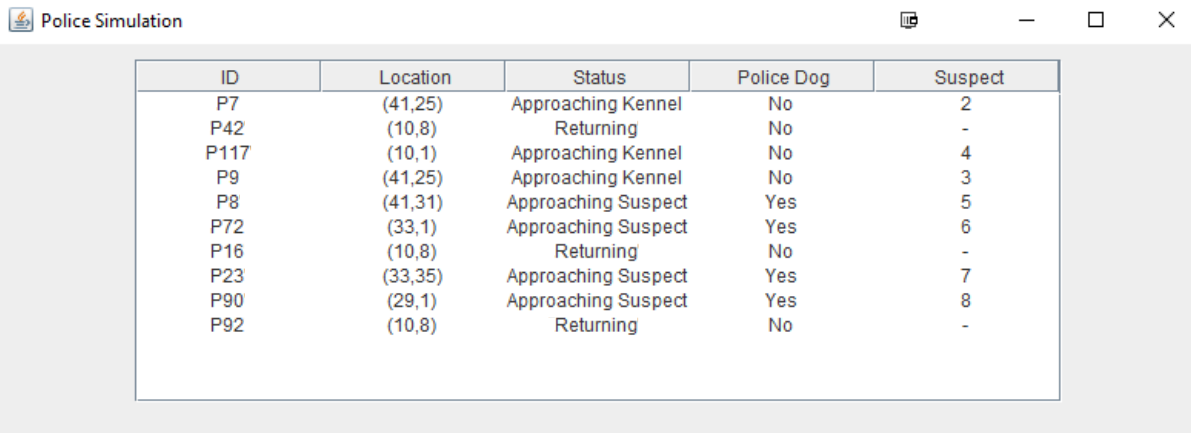
- A move is a one unit change in either the x-coordinate or the y-coordinate. Police units will always move to reduce the distance towards their target (kennel, suspect or station)
- The number of police units a station can accommodate is defined by the total number of police units divided by four, rounded up. E.g If there are 5 police units, each station can accommodate two police units.
- The total number of police dogs in the simulation is defined by the total number of suspects in the simulation divided by 2, rounded up. E.g if there are 7 suspects, the total number of police dogs available is 4.
- The task can be assumed to have a duration of 60 seconds.
- To simplify the task you can assume that all police units will start in the standby status with no assigned suspects. All suspects will hence then start in the 'Unassigned' status

#### Tips:

- Each police unit should have its own thread. These threads need to run independently of all the other police units.
- Consider which of your implemented data structures need to be threadsafe and implement accordingly.
- This is especially important when assigning unique resources – You don't want two threads thinking they own the same resource!

## Task B) GUI Interface

The second task is to implement a GUI that displays the simulation designed in Task A. An example of this GUI is shown below.



ID	Location	Status	Police Dog	Suspect
P7	(41,25)	Approaching Kennel	No	2
P42	(10,8)	Returning	No	-
P117	(10,1)	Approaching Kennel	No	4
P9	(41,25)	Approaching Kennel	No	3
P8	(41,31)	Approaching Suspect	Yes	5
P72	(33,1)	Approaching Suspect	Yes	6
P16	(10,8)	Returning	No	-
P23	(33,35)	Approaching Suspect	Yes	7
P90	(29,1)	Approaching Suspect	Yes	8
P92	(10,8)	Returning	No	-

The list displays the details of all the police units within the simulation. This will show a real-time display of where every police unit is, along with their status, whether they have a police dog with them and the id of their assigned suspect.

#### Tips:

- 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 and SwingUtilities is recommended to keep this simple

## Mark Scheme

Item	Points
Simulation Implementation: <ul style="list-style-type: none"><li>• Correctly updates Police Unit status</li><li>• Correctly updates Suspect status</li></ul>	6 4
Multi-threaded implementation uses one thread per police unit and correct synchronisation constructs to ensure data is not corrupted.	5
GUI Interface displays the status of Police Units and updates their details without freezing the UI	5
<b>Total (maximum mark)</b>	<b>20</b>

## Penalties

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	Inappropriate 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 the receipt of late submissions