COSC201 Assignment 2: making deliveries

Due: 11:59 p.m. Wednesday May 11, 2022

- an implementation of some schemes for "packet management", also in a grid, and a short report on the effectiveness of various strategies.

Making deliveries (15 points)

You're going to be managing a large warehouse whose floor area is divided into a grid. Within each cell of the grid sit some packets. Each packet belongs somewhere, possibly where it's sitting, and possibly elsewhere in the grid. There's also a single robot in each cell who, in a minute, can deliver one package to an adjoining cell (i.e. a cell sharing an edge with this one) and then return to its home cell.

Your mission, should you choose to accept it, is to develop a protocol for these robots that manages the movement of the packets to their destinations as efficiently as possible. Note that this delivery process happens synchronously. Each minute, all the robots choose which (if any) of their current packets to deliver and then they all carry out a delivery.

You remember hearing (in COMP162 perhaps from someone with an odd accent) that the original motivation for object-oriented programming was to handle large scale simulations and it seems like that's probably a good idea here. Surely, simulating the robots' behaviour and looking at various options will be easier (and less costly) than trying out experiments in the field.

In COSC201 we have discussed a variety of data structures and one of the main points of this part of the assignment is for you to choose from among them appropriately in the context of this problem. We aren't going to ask you to do extremely large simulations (the size of the warehouse is capped at 100×100 cells) so the most important criterion for this work is the correctness, and ease of use.

The raw materials

The following code framework is provided:

- Location.java A simple container class for two integers representing a location in the warehouse.

- Packet.java The representation of a packet. Each packet has a unique ID that identifies it. It contains a String (just to use as a description of its contents) and two Location fields representing its current location and its destination.
- PacketManager.java A packet management interface that includes appropriate methods for, well, for managing packets of course. Part of the work you do will be to implement various types of packet managers.
- PacketHoarder.java A useless packet manager included for illustrative purposes. In the robot context, it represents a robot who never makes any deliveries but just hoards all the packets it receives.
- RandomManger.java An almost useless packet manager. The associated robot maintains the current packets that need movement in a queue, picks the first one that needs to be moved and takes it to a random adjoining square (possibly crashing into walls!)
- RandomWarehouse.java Watch the fun and games in a random warehouse. This is just to give you a bit of an idea of how to make simulations run. Note the mechanism by which the "synchronous delivery rule" is implemented (by collecting all the deliveries into a queue, and only after they've all been collected actually doing the deliveries).

In general the data fields of the various classes have been given package (i.e., default) access to make it simple, e.g., to change the location of a packet. If you wish to add accessors and modifiers instead, that's fine.

A basic sensible manager (2 points)

Provide an implementation of PacketManager called BasicManager that operates on the following protocol.

- Packets are stored in a first in, first out queue
- sendPacket removes packets from the queue until one is found that needs to move. It moves it towards the destination preferring to change the row number if possible and then the column number (i.e., the route any packet follows will be up or down followed by left or right)
- receivePacket just adds the received packet to the queue.

A load-aware manager (3 points)

Provide an implementation of PacketManager called LoadAwareManager that operates on the following protocol.

- Packets are stored in a first in, first out queue which never contains packets that don't need to be moved
- sendPacket removes the first packet on the queue. If its destination is in a straight line from the current location, it just sends it one step in the appropriate direction. If it has two reasonable choices, it checks to see how many packets are held at those locations and sends to the smaller of the two. If the two are equal it prefers to change the row number i.e., prefers vertical moves over horizontal ones.
- receivePacket adds the packet to the queue unless this is its destination.

A priority manager (3 points)

Provide an implementation of PacketManager called PriorityManager that operates on the following protocol.

- Packets are stored in a priority queue which never contains packages that don't need to be moved. The priority of a packet is determined by the remaining total distance to its destination along the grid i.e., by the sum of the absolute differences of the current and destination row and column numbers.
- sendPacket Removes a packet of highest priority from the queue (i.e. one of the ones that needs to move furthest). If its destination is in a straight line from the current location, it just sends it one step in the appropriate direction. If it has two reasonable choices, it checks to see how many packets are held at those locations and sends to the smaller of the two. If the two are equal it prefers to change the row number i.e. prefers vertical moves over horizontal ones.
- receivePacket adds the packet to the queue unless this is its destination.

A manager of your choice (2 points)

Think of another (sensible) protocol and implement it. Make sure that the javadoc describes how the protocol works.

A report (5 points)

In a perfect world, every packet would be delivered in k minutes where k is the distance it needs to travel in the grid. For instance, this could be possible if each packet had its own robot attached to it. So, I'm going to define the *efficiency* of a delivery scheme to be the ratio

between the maximum distance that any single packet needs to travel and the total time required for all deliveries to be made.

For instance, if we had an 8×8 grid, and (at least one) packet needed to travel from one corner to the other (a distance of 14), but the total time required was 20 minutes in a given protocol then we'd say that, for this scenario, the protocol had 70% efficiency (since $14/20=0.7$ which is 70%).

Some obvious factors affecting the efficiency of a protocol are the total number of packets (if there are more packets than cells we can't ever even imagine attaining 100% efficiency) and the initial distribution of the packets (say we start with 100 packets in one cell all of which need to be moved to the same adjoining cell - we have a minimum possible time of 100 and so a maximum possible efficiency of 1%).

In a short report, discuss the efficiencies of the managers that you implemented in a few different contexts. Make sure to include at least the following cases:

- The number of packets and cells is the same. All the packets start in the upper left corner, and each packet has a different destination.
- The number of packets and cells is the same. Each cell contains one packet to begin with, and has a random destination (different packets might have the same destination).
- As the previous case, but each cell has only a 25% chance of containing a packet.

You should consider various sizes of grids (but need not consider any larger than 100×100.)