SWEN30006 Software Modelling and Design
Project 1: Automail
- Project Specification -

Team Project
Weight: 12%
Due: Wed 21st April

School of Computing and Information Systems
University of Melbourne
Semester 1, 2021

Background: Automail

Delivering Solutions Inc. (DS) has recently developed and provided a Robotic Mail Delivery system called *Automail* to the market. Automail is an automated mail sorting and delivery system designed to operate in a large building that has dedicated mail rooms. The system offers end-to-end receipt and delivery of mail items within the building and can be tweaked to fit many different installation environments. The system consists of two key components:

Figure 1: Artistic representation of one of our robots

- A **MailPool** subsystem which holds mail items after their arrival at the building's mail room. The mail pool decides the order in which mail items should be delivered.

- **Delivery Robots** which take mail items from the mail room and delivers them throughout the building. Each robot has two hands and one tube, i.e., a backpack-like container attached to each robot for carrying items (see Figure 1). The robot can hold one item in its hands (i.e., two hands carry one item) and one item in its tube. If a robot is holding two items (i.e., one in its hands and one it its tube) it will always deliver the item in its hands first. An installation of Automail can manage a team of delivery robots of any reasonable size.

DS also provides a **simulation** subsystem to show that Automail can operate to deliver mail items within a building. The subsystem runs a simulation based on a property file and shows the delivery log of the robots and delivery statistics, e.g., the elapsed time before each mail items is delivered and how long has it taken for all the mail items to be delivered. The system generates a measure of the effectiveness of the system in delivering all mail items, considering time to deliver and the types of mail items. *You do not need to be concerned about the detail of how this measure is calculated.*

The simulation subsystem uses clock to simulate operations of the mail pool and robot subsystems. Broadly speaking, for each tick of the clock (i.e. one unit of time), the mail pool subsystem will load items to the robots if there are robots available at the mailroom; and the robots will either move to deliver an item (if there are items in its hands or tube), deliver an item, or move to return to the mailroom (if all items are delivered). Currently, the

robots offered by *DS* will take one unit of time when moving one step (i.e., moving up or down one floor in a building). For example, if a mail destination is four floors from the mailroom, a robot will take 4 units of time for delivery, plus 1 unit of time for delivery, plus 4 units of time for returning to the mailroom.

You can assume that the hardware of this system has been well tested and will work with the Robot subsystem. The current software seems to perform reasonably well. However, the system is not well documented.

Your Task

Due to the COVID-19 and economic downturn, the government now allows building owners to charge additional service fees to tenants. Therefore, customers of DS have requested DS to update their robot mail system to support a charge capability and to pilot a change towards charging service fees. DS has hired your team to extend the latest version of Automail to include the ability to have robots **charge** tenants upon successful delivery of mail items. The Charge is formulated as follows:

$$\text{Charge} = \textit{Cost} * (1 + \textit{Markup Percentage})$$

where

- ***Cost*** *= Service Fee + Activity Cost*
  - ***Service Fee*** is varied by servicing floors (*the destination of a mail item*) and controlled by the building owner thus **robots** need to perform remote lookups to an external Building Management System (BMS) at the time of delivery (further details below).
  - ***Activity Cost =*** *Activity Units * Activity Unit Price* and is calculated based on the billable activities required for delivery. Currently, the following activities are identified as billable:
    - **Robot's Movement**: for a robot to move up/down a floor costs **5 activity units**.
    - **Remote lookup:** performing a lookup on the Service Fee costs **0.1 activity units independent** of whether the lookup is successful.
    - Activity Unit Price is set to 0.224 AUD **per activity unit** and subject to change (**configurable**).
    - A future version of the system may factor mail item weight into the Activity Cost, and may include a penalty for delivery delays.
- ***Markup percentage*** is set to 5.9% and subject to change (**configurable**)**.**

In addition, to help make tenants happier about the change, DS customers want to be able to provide priority delivery on high-cost mail items. To support this, they want to be able to set a Charge threshold; **any mail items with an expected Charge exceeding the Charge threshold (considered to be "priority mail items") are to be delivered ahead of the rest of the non-priority mail items**.

Further, during the piloting period, DS customers want to be able to precisely control the behaviour of Automail to ensure that the addition of charges does not break the system. To assist with this, it is required that: when the Charge threshold is set to zero (used to mean that there are *no priority mail items*), *the behaviour of the revised system should be identical to that of the original system*. By identical, we mean the output should be unchanged, and therefore the timing and order of deliveries should also be unchanged.

Accessing the BMS via Modem

For security reasons, the building owner will provide **a modem** which has a built-in function programmed to forward service fee lookup requests to the BMS; you can assume this modem is installed and available. Note that:

- Service fees are not static; they can increase (but not decrease) during a delivery session.
- Lookups depend on network connectivity and so can fail, with failure reported by the lookup method.
- As Automail is responsible for the infrastructure, it is only reasonable to charge the tenant one such lookup fee per mail item delivery.

- Customers require the Automail update be built for fault tolerance: the delivery **must** be successful with fees charged. Overcharging must not occur; using the most recent service fee retrieved for a floor is acceptable if necessary, to make a delivery.
- It may be necessary to incur costs which can't be passed on to the tenants. Your design should account for this and your justification should include why these costs were necessary. (Note: we are not expecting you to derive an optimal solution regarding these costs.)

Output Log

DS would also like you to **adjust** the log **and add some statistics tracking** to the software so that they can see how the charge capability is being used. This charge information will help them to understand the wear and tear that their robots are going to go through as well as provide a tally of fee related aspects.

```
T: 438 >   R2(0)-> [Mail Item:: ID:    101 | Arrival:   46 | Destination:  1 | Weight:  289]
T: 439 >   R0(1) changed from WAITING to DELIVERING
T: 439 >   R0(1)-> [Mail Item:: ID:     73 | Arrival:   48 | Destination:  1 | Weight: 1188]
T: 439 >   R1(1) changed from WAITING to DELIVERING
T: 439 >   R1(1)-> [Mail Item:: ID:     10 | Arrival:   93 | Destination:  1 | Weight: 1278]
T: 439 > Delivered( 157) [Mail Item:: ID:    101 | Arrival:   46 | Destination:  1 | Weight:  289]
T: 439 >   R2(0) changed from DELIVERING to RETURNING
T: 440 > Delivered( 158) [Mail Item:: ID:     73 | Arrival:   48 | Destination:  1 | Weight: 1188]
T: 440 >   R0(0)-> [Mail Item:: ID:     15 | Arrival:   72 | Destination:  1 | Weight: 1308]
T: 440 > Delivered( 159) [Mail Item:: ID:     10 | Arrival:   93 | Destination:  1 | Weight: 1278]
T: 440 >   R1(0)-> [Mail Item:: ID:    142 | Arrival:  117 | Destination:  1 | Weight: 1859]
T: 440 >   R2(0) changed from RETURNING to WAITING
T: 441 > Delivered( 160) [Mail Item:: ID:     15 | Arrival:   72 | Destination:  1 | Weight: 1308]
T: 441 >   R0(0) changed from DELIVERING to RETURNING
T: 441 > Delivered( 161) [Mail Item:: ID:    142 | Arrival:  117 | Destination:  1 | Weight: 1859]
T: 441 >   R1(0) changed from DELIVERING to RETURNING
T: 442 | Simulation complete!
Final Delivery time: 442
Delay: 103952.89
```

Figure 2: Sample log and output of the current version.

Log:

The original version records the number of items in the tube (see the number in parenthesis):
R0(0) means Robot id 0 have no item in the tube.
R0(1) means Robot id 0 has one item in the tube.

Additional information should be appended to the current log item upon a successful delivery as shown below in **blue** for the following example:

[Mail Item:: ID: 60 | Arrival: 95 | Destination: 5 | Weight: 1856 **| Charge: 12.51 | Cost: 8.73 | Fee: 2.75 | Activity: 87.93**]

Statistics tracking:

DS would like you to record, and output at the end of the mail run:

1. The total number of items delivered.
2. The total billable activity.
3. The total activity cost.
4. The total service cost.
5. The total number of lookups with a clear breakdown of success & failure.

In addition**,** the changes in log and stats tracking can be toggled on & off as required (see Configuration below).

When the simulation ends you should print this information to the system output stream. As with the behaviour, you should apply software engineering and patterns knowledge to support this statistics tracking. Your design should consider that DS may want to track additional statistical information relating to robot performance in the

future. Once you have made your changes, your revised system will be benchmarked against the original system to provide feedback on your results to DS.

Other useful information:

- The **mailroom** is on the ground floor (floor 1).
- **All items** are stamped with their **time of arrival**.
- **All items** arrive at the mail pool in batches, so all items in a batch receive the same timestamp.
- The mail pool is responsible for sorting and assigning mail items to the robots for delivery.
- All mail items have a **weight**. However, the weight of an individual item is not heavier than 2,000 units of weight. Otherwise, exceptions will be thrown.

Your task is to update the latest version of Automail developed by DS to show how the charge capability could be incorporated into the robot management system. In doing this you should apply your software engineering and patterns/principles knowledge to refactor and extend the system to support this new behaviour. Note that the behaviour described in this document is just one possible use of this functionality. When designing your solution, you should consider that *DS* may want to extend the capability in the future.

The Base Package

You have been provided with an Eclipse project export representing the current version of the system, including an example configuration file. This export includes the full software simulation for the Automail product, which will allow you to implement your approach to supporting the charge capability.

To begin:

1. Import the project zip file as you did for Workshop 1.
2. Try running by right clicking on the project and selecting **Run as... Java Application**.
3. You should see output like that in Figure 2 showing you the current behaviour of the Automail system.

This simulation should be used as a starting point. Please carefully study the sample package and ensure that you are confident you understand how it is set up and functions before continuing. If you have any questions, please make use of the discussion board or ask your tutor directly as soon as possible; it is assumed that you will be comfortable with the package provided.

Note: By default, the simulation will run without a fixed random seed, meaning the results will be different on every run. To have a consistent test outcome, you need to specify the seed. You can do this in the configuration file or by editing the Run Configurations (Arguments tab) under Eclipse and adding an argument. Any integer value will be accepted, e.g. 11111 or 30006.

Configuration and Project Deliverables

As discussed above, and for the users of Automail to have confidence that changes have been made in a controlled manner, you are required to preserve the Automail simulation's existing behaviour. Your extended design and implementation must account for the following:

- Preserve the behaviour of the system for configurations where the charge capability is turned off and the display of charge information is turned off (i.e. ChargeThreshold=0; ChargeDisplay=false). Note that "preserve" implies identical output. We will use a file comparison tool to check this.
- Add the handling and delivering behaviour for the charge capability as described above.
- Add statistics tracking as requested above.

It's recommended that you understand the high-level design of current system so that you can effectively identify & update relevant parts. And, you also don't need to refactor the whole system.

Note: Your implementation must not violate the principle of the simulation by using information that would not be available in the system being simulated. For example, it would not be appropriate to use information about mail items which have not yet been delivered to the mail pool. It would also not be appropriate to violate the implied physical limitations, for example by having robots teleport. We also reserve the right to award or deduct marks for clever or very poor code quality on a case-by-case basis outside of the prescribed marking scheme.

Finally, DS wants you to provide a pdf outlining your design changes, where you made them in the system, why they were made there (in terms of GRASP), and, where relevant, how they account for possible future system changes. The text should be around 1 page in length and should not exceed 2 pages. Diagrams can be included to support your explanation, if useful, outside of these limits.

Testing Your Solution

We will be testing your application programmatically, so we need to be able to build and run your program without using an integrated development environment. The entry point must remain as "swen30006.automail.Simulation.main()". You must not change the names of properties in the provided property file or require the presence of additional properties.

Note: It is **your team's responsibility** to ensure that the team has thoroughly tested their software before submission.

Submission

Detailed submission instructions will be posted on the LMS. You must include your team number in all your pdf submissions, and as a comment in all changed or new source code files provided as part of your submission.