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

School of Computing and Information Systems
University of Melbourne
Semester 2, 2020

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:

- 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 deliver 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 (including zero!).

Figure 1: Artistic representation of one of the DS robots

DS also provides a **simulation** subsystem to show that Automail could operate to deliver mail items within a building. The subsystem runs a simulation based on a configuration file (or a property file) and shows the delivering log of robots and delivering statistics, e.g., how long will all the mails be delivered and total delay that the mail items are awaiting until delivered. The system generates a measure of the effectiveness of the system in delivering all mail items, taking into account time to deliver and 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 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 on the 5th floor, a robot will take 5 units of time for delivery, plus 1 unit of time for delivery, plus 5 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 social-distancing situation, *DS* sees a promising opportunity to expand the market! So, *DS* has hired your team to extend the latest version of their Automail to *include* the capability of delivering food items. To provide this new capability, *DS* has developed a new tube, called a **food tube**, which is a special tube with a heating system to ensure that the delivered food is still fresh and warm. Since a food tube is newly developed, it has an extra capacity to carry **three** food items at a time. However, there are a few limitations and restrictions on the robots when carrying food:

1. Food items are stacked in a food tube. So, the delivery order of the food items is last in first out.
2. Food items must be carried only by a food tube and other regular items must be carried by either a robot's hands or a regular tube.
3. When a food tube is used, the robot has to detach all the hands and the regular tube. That means the robot with a food tube can carry only food items.
4. When the first food item is assigned to a robot, the robot will be installed with a food tube (including detaching hands and a regular tube). When a robot returns from delivering food to the mailroom, a food tube is detached and the regular hands and a regular tube are attached back.
5. Since fuel for a heating system is expensive, the food tube should be full before delivery unless there are no more food items awaiting in the mail pool at that time point. Also, before a robot with food items is out for delivery, a food tube needs 5 units of time to boot a heating system.
6. To reduce the risk of food contamination, there should be no other robots delivering items at the same floor as the robot with a food tube. That means the destination floor of a food item should be locked when the destination floor is set until the food is delivered.

Your task is to update the latest version of Automail developed by *DS* to show how the food delivering ability could be incorporated into the robot management system. The behaviour of the system when delivering regular items *should not change,* meaning that the behaviour of the system remains **exactly** the same when no food items arrive, or the food delivering mode is turned off in the simulation.

In doing this you should apply your software engineering and patterns 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 incorporate or trial other uses of the ability in the future.

DS would also like you to adjust the log and add some statistics tracking to the software so that they can see how the food delivery ability is being used. This will help them to understand the wear and tear that their robots are going to go through.

```
T: 660 >   R0(0)-> [Mail Item:: ID:     15 | Arrival:   72 | Destination:  1 | Weight: 1308]
T: 660 > Delivered( 158) [Mail Item:: ID:    101 | Arrival:   46 | Destination:  1 | Weight:  289]
T: 660 >   R2(0) changed from DELIVERING to RETURNING
T: 661 > Delivered( 159) [Mail Item:: ID:     15 | Arrival:   72 | Destination:  1 | Weight: 1308]
T: 661 >   R0(0) changed from DELIVERING to RETURNING
T: 661 >   R2(0) changed from RETURNING to WAITING
T: 662 >   R0(0) changed from RETURNING to WAITING
T: 662 >   R2(1) changed from WAITING to DELIVERING
T: 662 >   R2(1)-> [Mail Item:: ID:     10 | Arrival:   93 | Destination:  1 | Weight: 1278]
T: 663 > Delivered( 160) [Mail Item:: ID:     10 | Arrival:   93 | Destination:  1 | Weight: 1278]
T: 663 >   R2(0)-> [Mail Item:: ID:    142 | Arrival:  117 | Destination:  1 | Weight: 1859]
T: 664 > Delivered( 161) [Mail Item:: ID:    142 | Arrival:  117 | Destination:  1 | Weight: 1859]
T: 664 >   R2(0) changed from DELIVERING to RETURNING
T: 665 | Simulation complete!
Final Delivery time: 665
Delay: 189033.98
```

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
For your version, you should show the number of items in either the regular tube or food tube. For example, `R0(2)` means Robot id 0 have 2 items in the tube.

In addition, the log should show the type of item in the following format
Mail item: `[Mail Item:: ID: 60 | Arrival: 95 | Destination: 5 | Weight: 1856]`
Food item: `[Food Item:: ID: 60 | Arrival: 95 | Destination: 5 | Weight: 1856]`

Statistics tracking:
DS would like you to record:

1. The number of regular items delivered.
2. The number of food items delivered.
3. The total weight of the regular items delivered
4. The total weight of the food items delivered.
5. How many times a food tube is used (i.e., being attached).

When the simulation ends you should print this information to the console. 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** (regular and food items) are stamped with their **time of arrival**.
- **All items** (regular and food items) arrive at the mail pool in batches, so all items in a batch receive the same timestamp.
- If a 'food delivery' mode is turned on (i.e., 'DeliverFood=true' in the property file), the simulation will generate food items. Otherwise, only regular items are generated.
- If an item is 'food', the system must handle the item as described above.
- 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.
- Attachment/detachment doesn't take extra time. It happens at the same time as the food items are loaded and at the same time when a robot arrives at the mailroom.

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 food delivery using a food tube.

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 similar to 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 seed, meaning the results will be random on every run. In order 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.

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 food items are not generated and statistical logging is turned off (i.e. DeliverFood=false; Statistics=false). Note that "preserve" implies identical output. We will use a file comparison tool to check this.
- Add the handling and delivering behaviour for food items when using a food tube as describe above.
- Add statistics tracking to show the total weight and number of food items and regular items delivered, and the total amount of time spent for attaching and detaching a food tube.

You do not need to understand or even read all the code provided; you only need to understand the parts you are changing and those they depend on. Moreover, you don't need to refactor the whole system. Your main focus should be extending the mail pool and Robots subsystem.

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 the 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.

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 responsibility** to ensure that you have thoroughly tested your software before you submit it.

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.