Part A - Mailbot Blues

SWEN30006 Semester 1 2018

Overview

You, an independent Software Contractor, have been hired by *Robotic Mailing Solutions Inc. (RMS)* to provide some much needed assistance in delivering the latest version of their product Automail to market. The Automail is an automated mail sorting and delivery system designed to operate in large buildings that have dedicated Mail rooms. It offers end to end receipt and delivery of all mail within the large building, and can be tweaked to fit many different installation environments. The system consists of four key components:

- **A Mail Pool** system which can hold a number of mail items which have arrived at the building.
- A **Delivery Robot** (see Figure 1) which delivers mail items from the mail pool throughout the building.
- A **Storage Unit**, that is, a 'backpack' which is attached to the delivery robot. It can contain at most four mail items; these mail items are in delivery order.
- **A Mail Selecting** system which decides what mail items should go into a robot's backpack for delivery and in what order, and in some cases which items should come out.

Figure 1: Artistic representation of our robot

The hardware of this system has been well tested, and the system engineers have confidence in their design. Recently a new robot was acquired which removed the mail item weight limitation of the earlier model. *RMS* wants to run two robots together, an old one and a new one, for a more effective solution. Unfortunately, the performance seen so far still seems well below optimal, and sometimes results in heavy items being given to the weak robot. *RMS* has traditionally been a hardware company, and as such do not have much software development experience. As a result, the strategies that they are using to organise the mail and select the mail for delivery are very poor.

Your job is to apply your software engineering knowledge to fix the mail sorting and to develop a better strategy for selecting the mail for delivery, all with the aim to improve the performance of their system. Once you have made your changes, they will be benchmarked to provide feedback on your performance to *Robotic Mailing Solutions*.

Other useful information

- The **mailroom** is on the ground floor.
- All **mail items** are stamped with their **time of arrival**.
- Some mail items are **priority** mail items, and carry a priority of either **100 high or 10 low**.
- Priority mail items are delivered and registered one at a time, so **timestamps are unique for priority items**.
- Normal (non-priority) mail items are delivered in a batches; **all items in a normal batch receive the same timestamp**.
- A **Delivery Robot** carries a **Storage Unit** which can contain **at most four mail items**.
- A Delivery Robot can be sent to deliver mail even if the Storage Unit is not full.
- A Delivery Robot **delivers the mail in the order it appears in the storage unit**.
- All mail items have a **weight** from **200 to 5000 grams**; the weak delivery robot can only deliver items **up to 2000 grams**.
- The system is judged on the sum across all mail items of a measure of the time taken to deliver the item
  - the measure has a power factor of 1.1 to make late items more urgent, and is scaled up for priority items by square root of the priority (e.g. 11 times for priority 100)
  - Specifically: (delivery time - arrivalTime)^1.1 * (1+sqrt(priority))

The Sample Package

You have been provided with a zip file containing all you will need to start your work for *Robotic Mailing Solutions*. This zip file includes the full software simulation for the Automail product, which will allow you to test your strategies in an experimental environment. This zip file provides you the application as an Eclipse project. To begin:

1. import the project zip file as you did for Workshop 1.
2. Try running by right clicking on the project `Part A` and selection `Run as... Java Application`.
3. You should see an output similar to that below, showing you the current performance of the Automail system (or an exception, depending on the particular run).

```
T: 190 | Delivered     Mail Item: | ID: 1670675563| Destination: 6| Arrival: 21| Weight: 411
T: 190 | R1746572565 changed from DELIVERING to RETURNING
T: 191 | Simulation complete!
Final Delivery time: 191
Final Score: 5377.78
```

Figure 2: sample results

This simulation should be used as a starting point in developing your benchmark program for your strategies. You have also been provided with compiled versions of the Javadoc for this package, located in the `doc` folder. You should use this as your guide in developing your solution.

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.

On Calculation of Stats

The sample package has the capability to run simulations and provide a stats summary after the simulation has finished. It is important that you understand how these statistics are calculated so that there is no misunderstanding that leads you to incorrect assumptions about the behaviour of your strategies. In particular, the "Final Score" is the performance measure you need to improve: lower for this score is better.

> **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 by editing the Run Configurations (Arguments tab) under Eclipse and adding an argument. Any integer value will be accepted, e.g. 30006.

Strategies Package Diagram

Seen below is a diagram of the strategies interfaces and related elements. Be aware that this does **not** represent all the elements in Simulation. This diagram is provided below to aid in the understanding of the application.

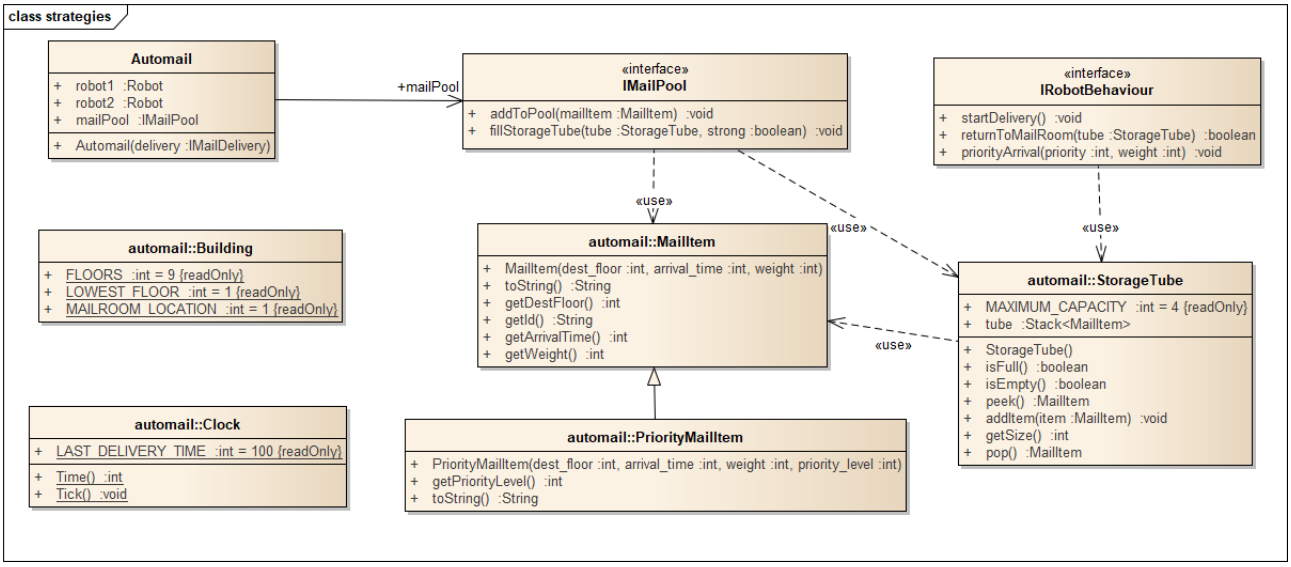

Figure 3: Sample Diagram

The Task

As you will notice, the current strategies used for sorting mail are incorrect and the current strategies for selecting mail and responding to priority arrivals are simplistic and not optimised for any particular use case. Thankfully, *Robotic Mailing Solutions Inc.* has made use of the Strategy pattern (see reference **here**) to make their software more flexible, by creating a common interface for dealing with mail in the mailpool and for the robot behaviour in selecting mail.

Your task is to develop new sorting and selecting strategies for mail within the Automail system. *Robotic Mailing Solutions inc.* is interested in correcting the `MailPool`, and improving the `RobotBehaviour`, as internal testing has shown these to be a significant factor in limiting current performance. Specifically you must:

1. Create a class `MyMailPool` in the strategies package implementing the `IMailPool` interface.
2. Create a class `MyRobotBehaviour` in the strategies package implementing the `IRobotBehaviour` interface.
3. Modify Automail.java to use these classes (code already provided in the file as comments).
4. Test your modified version of the system (including on a University lab computer to ensure compatibility).
5. Ensure all your code is commented and of good quality (see below)

You must include in your code, comments explaining the rationale for your data structure and algorithm choices and how these work toward achieving your goals for the `MyMailPool` and `MyRobotBehaviour` implementations. It is important to note that the strategies you provide *must* be different from those provided to you in the sample package, and must achieve a better (lower) value for the "Final Score" statistic.

> **Note:** This is not an algorithms subject so we do not expect optimal planning solutions. Your solution must match the provided interface specifications and demonstrate that you have made an attempt to address the problem by organising the incoming mail appropriately in the mailpool and using that organisation in selecting mail for each robot, and by having the robots exhibit a sensible response to receiving notification of priority mail. In addition, you must not violate the principle of the simulation by using information that would not be available in the system being simulated, for example by using information about mail items which have not yet been delivered to the mail pool.

We will be using our own version of the sample package during marking in which we will use our parameters, such as random seed and last delivery time. You will be submitting **only** the two files listed above (1. and 2.) , which must work with the rest of the sample package. You should not otherwise modify the behaviour of the sample package.

Submission

You will submit a zip file (e.g. "submission.zip") containing exactly two files, "MyMailPool.java" and "MyRobotBehaviour.java".

- We will use an extract/build/run script to extract these files based on their names and test them with the rest of the simulation framework.
- Your file must not be in another compression format or contain other folders or files: if our script doesn't find the two required files, then you will receive 0 (zero) marks.
- If our script cannot build and run your program you will receive at most 1 mark.

  Note: Your program **must** run on the University lab computers (using Java 8). It is **your responsibility** to ensure you have tested in this environment before your submit your project.

Marking Criterion

This project will account for 5 marks out of the total 100 available for this subject. These will be broken down as follows:

Criterion	Mark
Simulation compiles and runs correctly as per specification above	3 marks
Code quality, as described below.	2 marks

We expect to see good variable names, well commented functions, and inline comments for complicated code. We also expect good object oriented design principles and functional decomposition. If we find any significant issues with code quality we may deduct further marks.

We also reserve the right to award or deduct marks for clever or poor code quality on a case by case basis outside of the prescribed marking scheme.

There is also a *bonus mark* available for submissions in the top 5% of performance. We will be judging performance by comparing the "Final Score" statistic for your strategy. (The lower the score the better).

> **On Plagiarism:** We take plagiarism very seriously in this subject. You are not permitted to submit the work of others under your own name. This is an **individual** project. More information can be found here: (https://academichonesty.unimelb.edu.au/advice.html).

Submission Date

This project is due at **11:59pm on Tuesday 20th of March.** Any late submissions will incur a 1 mark penalty per day unless you have an appropriate reason with supporting documents. If you have any issues with submission, please email Peter Eze at peter.eze@unimelb.edu.au, before the submission date.