Part B - Mailbot 2: Judgment Day

SWEN30006, Semester 2 2018

Overview

Continuing your work as software contractors, your group has been rehired by *Robotic Mailing Solutions Inc. (RMS)* to address further issues with their product AutoMail. As you are aware, RMS uses a simulation framework to test out strategies, for example, for MailPool and Robot behaviour. RMS would like to be able to broaden the framework capability to support new robot types it is considering, to speed up delivery or to carry fragile mail items which currently have to be hand delivered.

Currently, the Automail framework supports two robot types:

- Standard: with a tube capacity of four, and effectively able to carry mail items of any weight.
- Weak: with a tube capacity of four, and able to carry mail items no more than 2000 grams.

Two additional robots being considered right now by RMS are:

- Big: with a tube capacity of six, and effectively able to carry mail items of any weight.
- Careful: with a tube capacity of three, and effectively able to carry mail items of any weight, but slower than the other robots.

Fragile mail items have not been deliverable by the Automail system as the movement of the existing robots results in breakage. The new careful robots are able to overcome this limitation. However, even these robots can only carry one fragile mail item at a time, and their movement is half the speed of the existing robots (see details below).

RMS's experience in improving strategies (with your help) has made them aware that the simulation framework itself, while functional, is not up to standard from a design perspective. The designers did not consistently apply good design principles, and have ended up with a system which is not sufficiently flexible or modifiable.

You have been assigned the task of extending the design and associated implementation to support the additional robot types, with consideration for future robot types. In the current framework, the robots used by Automail are hard-coded; the revised framework must allow them to be specified as a property. In addition, to ensure existing Automail sites are still represented by the simulation, and in order for the client to have confidence that changes have been made in a controlled manner, you are required to preserve the Automail simulation's existing behaviour.

Extended Design and Implementation

Your extended design and implementation must account for the following:

- Preserve the behaviour of the system for configurations of Standard and Weak robots, without fragile mail items. (Preserve = identical output)
- Preserve the existing fragile mail item breakage conditions.
- Add support for the two new robot types. The behaviour of a Careful robot is the same as that of other robots, except where it moves between floors for a delivery: the first step (where another type of robot would move one floor), the Careful robot will not move; the next step (and every subsequent second step), the careful robot will move one floor.
- Preserve support for the properties in the properties file, and add support for using the "Robots" property to set the simulation robots.

- Assuming Automail has been configured with a Careful robot, safely deliver all mail items with the inclusion of fragile ones.
- The mail pool should allow for all robots configured to be used in delivery; beyond that, there are no additional requirements on the performance of the delivery algorithm.

In support of your updated and extended design, you need to also provide the following diagrams:

1. A design class diagram (DCD) of all changed and directly related components, complete with visibility modifiers, attributes, associations, and public (at least) methods. You may use a tool to reverse engineer a design class model from your implementation as a basis for your submission. However, your diagram must contain all relevant associations and dependencies (few if any tools do this automatically), and be well laid-out/readable; you should not expect to receive any marks for a diagram that is reverse engineered and submitted unchanged.
2. A design sequence diagram (DSD) illustrating (and contrasting) the behaviour of a robot moving towards its destination (represented by the Robot.moveTowards() method in the existing simulation). The two different robot movement types must be shown, and exceptions should be illustrated (see the UML break frame).

Your submitted implementation must be consistent with your new design, and will be marked on code quality, so should include all appropriate comments, visibility modifiers, and code structure. As such, your submitted DCD and DSD must be consistent with your submitted implementation.

While making your changes, you will have observed limitations in the current design, some of which you had to overcome to make your changes. For some addition robot types beyond those requested (e.g. a robot which can't carry heavy items but can carry six items), it may be possible to add them without much additional overhead. Other robot types would be harder to add (e.g. a robot which could be called back to the mail pool if a high priority mail item turned up). You are also required to write a short design report, justifying the design changes you made, and recommending a small number of further changes that could be made which would make it easier to add additional robot types in the future. This report should use pattern languages to support the justifications/recommendations. Feel free to make up examples of robot types to illustrate your judgements.

Hints

There are three required tasks here–report, refactor, and extend–which are interrelated. You are likely to find that considering the changes required to refactor and extend the system will help you identify design recommendations for your report, and that aspects of the refactoring are best considered at the same time as you consider your approach to extending the system. However, be careful about the changes you make: reordering operations (esp. those involving generation of random numbers) will change the behaviour of the system and therefore the output. If you make such changes and proceed without detecting the problem, recovering may require substantial backtracking on the changes you've made.

Building and Running Your Program

We will be testing your application using the desktop environment, and need to be able to build and run your program automatically. 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** Your program **must** run on the University lab computers. It is **your responsibility** to ensure you have tested in this environment before your submit your project.

Marking Criteria

This project will account for 10 marks out of the total 100 available for this subject. These will be broken down as follows:

Criterion	Mark
Design Report	2.5 mark
Implementation of New Design	4.5 marks
Design Class Diagram	1.0 mark
Design Sequence Diagram	2.0 mark

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

Further, we expect to see good variable names, well commented functions, inline comments for complicated code. We also expect good object oriented design principles and functional decomposition.

For UML diagrams you are expected to use UML 2+ notation.

Finally, if we find any significant issues with code quality we may deduct further marks.

> **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 a **group** project. More information can be found here: (https://academichonesty.unimelb.edu.au/advice.html).

Submission

Only one member from your group should submit your project. You should submit four items at the same time (see checklist below) using a single submit command. You must include your group number in all of your pdf submissions, and as a comment in all changed or new source code files provided as part of your submission.

Submission Checklist

1. Your zipped complete updated source code package reflecting your new design and implementation (all Java source: top-level folder in zip file called "swen30006").
2. Design Analysis Report (pdf).
3. DCD reflecting your new design and implementation (pdf or png).
4. DSD reflecting your new design and implementation (pdf or png).

Submission Date

This project is due at **11:59 p.m. on Tue 18th of September.** Any late submissions will incur a 1 mark penalty per day unless you have supporting documents. If you have any issues with submission, please email Marko at marko.mihic@unimelb.edu.au, before the submission deadline.