Part B - Mailbot 2: Judgment Day

SWEN30006, Semester 2 2017

Overview

Continuing your work as software contractors, you have been rehired by *Robotic Mailing Solutions Inc. (RMS)* to address further issues with their product AutoMail. As you are aware, RMS use a simulation framework to test out strategies (for example, for MailPool and Robot operation). However, RMS's experience in improving such 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.

Your first task is to pass judgement on the existing software package, that is, to provide a detailed analysis of the package, highlighting inappropriate design decisions. Having completed this task, you will then be required to redesign and *refactor* the existing package applying good design principles and patterns to improve the flexibility and modifiability of the simulation platform to cope with RMS's changing needs. Finally, having improved the design, you are to extend the design to support some additional functionality and modify the existing simulation to match your modified design.

The Existing Simulation

You are familiar with the existing simulation from your earlier work. Other than the seed for random number generation, all parameters of the existing simulation are fixed in the code. This includes parameters such as the number of building floors, the number of mail items to be delivered, and the capability of the delivery robot. This does not allow RMS to vary the simulation to try out various combinations of parameters, such as: tall building with high mail volume and smart robot with communications.

The Task

There are three main components to your task, first an analysis of the existing design, second a new improved and refactored design and implementation of the simulation system, and third an extended design and implementation of the simulation system. We will now break these components down in detail.

Analysis of Existing Simulation

This task is a design analysis report on the provided simulation. You should focus on the design patterns used (if any) along with general object-oriented code quality. You should use the analysis exercises from workshop 4 as a guideline and focus on the GRASP patterns, on how well the provided code follows these patterns, and on where it does not. The focus here is on the design, rather than the coding; we don't want to hear, for example, about your preference for a particular Java coding idiom unless it is also relevant to the design.

In this report you should be critical of the existing software, but ensure that you backup your claims and statements with reasoned arguments about the design. Simply stating something is bad is not sufficient for this task. Neither is saying that your modified code (see below) is better. You should aim for between 1200 and 1500 words for this report.

Refactored Design and Implementation

Having now critically analysed the provided package, your next task is to provide a refactored software design and implementation that addresses the concerns listed in your report. The refactored implementation must also support using the properties file "automail.properties" provided to you with the source code as a means of setting the parameters specified in the properties file. At start has already been made towards using the properties file, however the code for this, in "Simulation.java", has been commented out.

Note that, as this is a refactoring, the behaviour of the system should remain unchanged, that is, the output for the same set of parameters (in code for the original vs. in the property file for the refactored version) should be identical.

Extended Design and Implementation

RMS has bought some second-hand robots from a failed competitor. These robots lack the communication capability of the existing robots, and so can't receive notifications. However, they do have an increased carrying capacity with a tube that holds up to six mail items. These robots have been dubbed the "Big_Simple" robots.

With the purchase of these robots, RMS has recognised the need to be able to model different robot configurations. RMS does not want support for arbitary configurations, only for a small defined set of configurations that can be modified or extended on an as needed basis. RMS wants the simulation extended to support three different robot configurations: "Small_Comms_Simple" (the original simple behaviour provided to you), "Small_Comms_Smart" (the new smart behaviour), and "Big_Simple" (the second-hand robots, with the new smart behaviour but without the ability to receive notifications). The choice of robot configuration used in the simulation should be made through the property file.

Note that, the addition of both the new robot configuration and of the means of selecting the desired robot configuration should not affect the behaviour of the system for the pre-existing robot configurations.

As part of your final updated and extended design, you need to provide the following diagrams:

(1) A design class model of all components, complete with visibility modifiers, associations and methods. You may use a tool to reverse engineer a design class model as a basis for your submission. However, your diagram model must contain all relevant associations and dependencies (few if any tools do this automatically) and be readable; a model that is reverse engineered and submitted unchanged is likely to receive few if any marks.

(2) For the Robot ("Small_Comms_Smart" configuration):

  (2a) A detailed UML state machine diagram. (Note: the states/triggers are straightforward, the guards/actions are more interesting.)

  (2b) A detailed UML design sequence diagram showing the behaviour of the robot for a scenario where it picks up two normal mail items and delivers one before returning and picking up a priority mail item which has arrived after the previous pickup.

These diagrams should be consistent with each other, and with the final updated implementation.

Implementation

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

Hints

While this work has been described as a sequence of three tasks (report, refactor, extend), you are likely to find that considering the changes required to refactor and extend will help you identify issues 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 should remain as "swen30006.automail.Simulation.main()". You should 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.

Submission Checklist

1. Design Analysis Report.
2. Class Diagram reflecting new design for the Automail simulation.
3. State Machine Diagram demonstrating the operation of the specified Robot configuration.
4. Sequence Diagram reflecting the operation of the specified Robot configuration for the specified scenario.
5. An updated source code package reflecting your new design.

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 Analysis Report for Provided Simulation	4 mark
Design Class Diagram	2 mark
State Machine Diagram	1 mark
Sequence Diagram	2 marks
Implementation of New Design	2 marks

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.

Finally, you are expected to follow UML 2.1+ syntax. You will lose marks for unclear UML diagram notation.

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 one zip file containing both the updated code package and the report and diagrams. *Your report and diagrams should be submitted as PDFs.* You must include your group number in all of your pdf submissions, and as a comment in all source code files provided as part of your project.

Submission Date

This project is due at **11:59 p.m. on Mon 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 William at tiow@unimelb.edu.au, before the submission deadline.