SWEN30006 PROJECT 1 REPORT

Antony Tragas: 1080479

Rajneesh Gokool: 1101512

Anjaney Chirag Mahajan: 1119668

The purpose of this report is to describe the changes and implementations for the Building Automail system. The program was extended to implement two new types of robots and adding a charging system for customers, that can be toggled in the "automail.properties" file.

1. Robot class.

To incorporate the new types of robots, the existing class Robot was changed to an abstract class. Which was then used to implement the existing regular robot type and two new robot types extending Robot. This was done to add new robot types to the system without creating many dependencies on the specific robot types itself as all the robot types will share some common function and only the specific function will have to be implemented. Pure Fabrication is also achieved by making an Abstract Robot class for all robot types which decreases coupling in our design and increases cohesion between each specific Robot class.

2. Robot subclasses (FastRobot, BulkRobot, RegRobot)

The subclasses of Robot use polymorphism (overriding) to call the method appropriate to the specific subclass. This increases extensibility as in order to add a new Robot class a software designer simply needs to override the abstract methods and set the attributes to desired values. Each robot type now has a particular Speed attribute which dictates how many times they can move per tick to deal with the FastRobot.

Robot also holds information about how many items they are holding which allows us to keep track of the index of the last item added to the tube. An array was implemented using an array for BulkRobot to hold MailItem objects, when accessing MailItems we modified methods to emulate a stack such that the last object added would be the first

out. Further Fast Robot was implemented such that the delivery speed of a MailItem is faster than that of a regular robot, this was done by changing the robot SPEED to be three times that of a regular robot. Additionally, the tube attribute was not added to Fast Robot and methods were modified to ensure items were not added to a tube.

3. Automail class

Automail now stores an array of Abstract Robot which will be used to store the different robot classes in the building. It is responsible for creating a robot since Automail uses the configuration properties passed into the system to initialize the different types of robots. Further we extended the functionality of Automail to add a “calculateFeeString” which receives Robot and Floor classes and returns a formatted string.

4. MailPool class

The Robot array in MailPool holds all different classes of Robot in one array. This is doable because each Robot class inherits from the abstract parent Robot class. This is an example of a use of indirection that allows us to add as many new Robot classes as we like without having to rewrite the MailPool class.

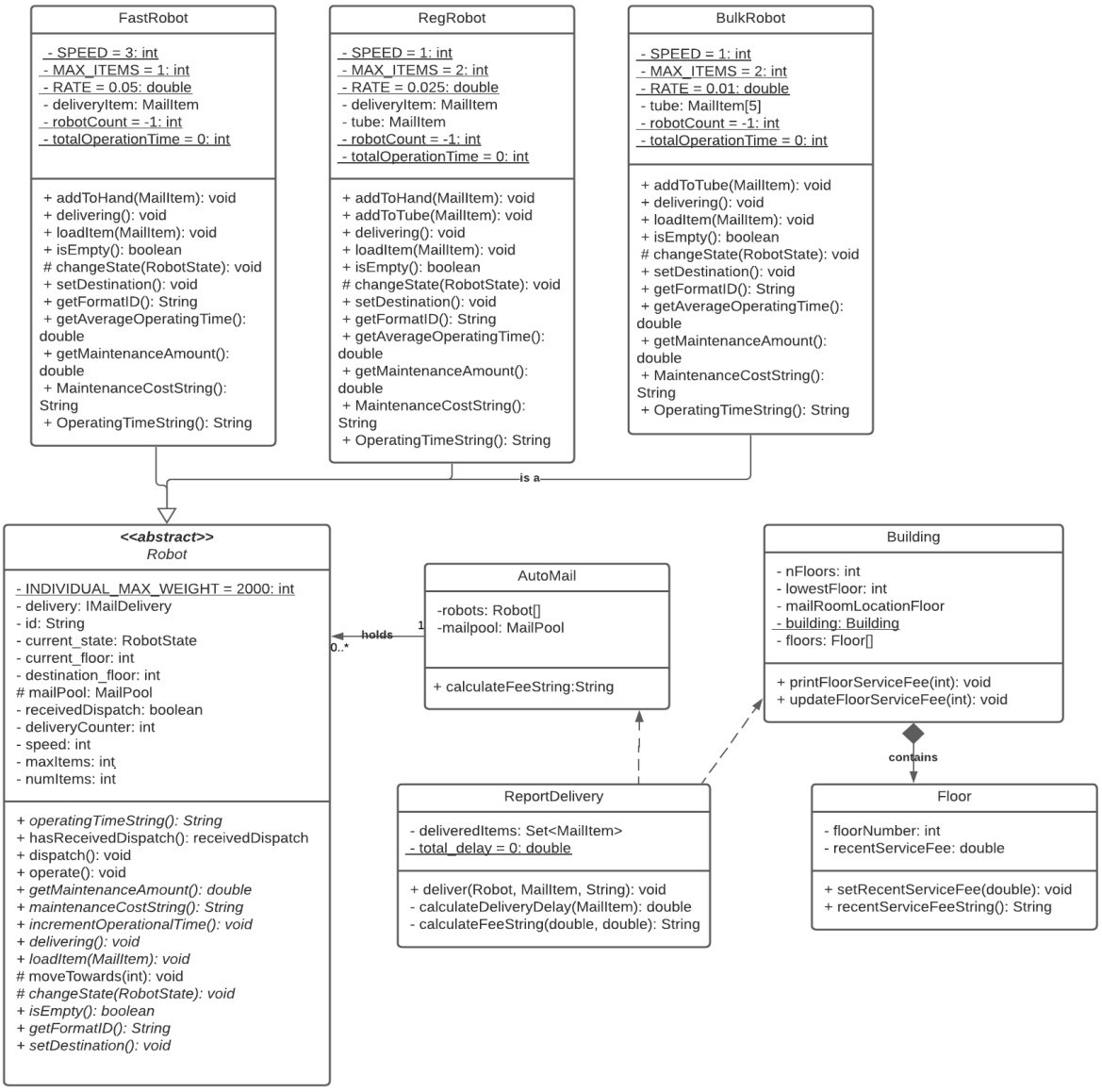
5. Building Class

Building class has been updated to contain an array of Floors which are stored in the Building object making use of the Indirection Principle to update the Service Cost of a particular floor when a robot has delivered an item. This helps to keep track of the recent service fee of each floor and increases cohesion in our design by supporting this central purpose.

6. Floor Class

The floor class was modified to use delegation to store the service fee. Rather than the Building class store the fees and send it to the ReportDelivery class, we mediate the two with the Floor Class. This helps reduce coupling, dependencies and preserves previous functionality.

Design Class Diagram



Design Sequence Diagram