SWEN30006-Project-1 Report

Written by *Workshop16Team02*

The report is going to introduce the additional features of the new Automail system and how the program be modified by our team. It would be written into two sections, Append and Modify.

Append

Charge class:

Charge is added to record the required information of charging fee details. Since recording charge is designed for the additional feature, all data would not be stored in the *MailItem*, instead it would be a new class which can maintenance the high cohesion.

Charger class:

Charger is a tool for creating and calculating all charges. *Charger* would have a hash map for recording latest service fee for each floor, so Automail can still charge the service fee when the external system fails to lookup service fee. Since the service fee is always equal to or larger to the previous service fee, the overcharging problem would not be concerned. According to low coupling pattern, the methods related to charge behaviour is belonging to *Charger*. Therefore, if there is a requirement of modifying charging calculation (such as considering mail item weight or penalty of delay time) in the future, it would be easier to do so. The class, which is extracted from *Charge* follows pure fabrication pattern, so it would maintain high cohesion. Furthermore, the Charger would be responsible to create a *Charge* for a *MailItem* (if client requires). Since *Charge* would record and aggregate the results of charges (in *finalCharge*), the creator would be assigned to *Charger*. In *Charger,* number of delivered item, total billable activity, total activity cost, total service cost and total successful & failed lookups would be recorded as static attributes. In addition, the billable activity would include all successful & failed lookup activities, yet activity cost would be only charge for 0.1 activity unit as required.

StatisticsTracker class:

StatisticsTracker is used to collect chargers from each robot, thus the delivering summary can be generated. This is an artificial class based on pure fabrication pattern for achieving high cohesion. The class can be extended for looking up delivering history of a specific robot or for aggregating different tracking data in the future.

ModemAdapter class:

ModemAdapter is a simple instance for assisting *Charger* to utilise *wifiModem.* By Indirection, the responsibility of communicating with *wifiModem* should be assigned

to the intermediate class named *ModemAdapter.* This ensures the program to be low coupling and to have more flexibility when a new external service be introduced afterward.

Modify

MailPool class:

MailPool contains a new attribute called *centralCharger* (Charger) and a new comparator named *ItemComparatorPriority*. *Charger* class is used to create charge for each mail item before it enters the pool, and the new comparator is used to sort the mail item decided by the charge threshold. In *MailPool*, it used a private inner class *Item* to hold the mail item and those attributes which may become the factor for new sorting method. In extension, the charge and charge threshold are applied to sort the priority queue. The private inner class *Item* follows protected variation. In this case, the variation of sorting reference could be modified by DS company.

MailItem class:

MailItem would contain a new attribute called charge (*Charge*) to collect the required data like service fee, activity cost, etc. For the future extension, an attribute named *delayTime* and a method for computing delay time have been added to the class. Eventually, the delay time of mail item could be computed when it is delivered by the *Robot*.

Robot class:

Robot would have a new attribute named charger (*Charger*), which can be used to calculate the final charge cost for the delivered mail items. Moreover, while computing the final charge cost for each MailItem use charger, some attribute on the charger like *totBillableActivity* and *totActivityCost* would be updated as well. The reason why each robot would have an exclusive charger is that there is a potential extension of tracking additional statistical information relating to single robot performance in the future.

Simulation class:

In this class, the deliver method would be updated to print out the charge information appending to the log when the feature is toggled on. Also, a statistics tracker is added to *Simulation*. To switch on or off the toggle, the new extension would be controlled by the two properties, *chargeThreshold* and *chargeDisplay*.