Group no 63:
Dipesh Hirani
Takuhiro Kikuchi
Congyi Wang

Design Analysis Report

As Robotic Mailing solutions Inc. had issues with their Automail product, we had to come up with a new design to make it more efficient. Addition of two more robots has been considered which are:

- Big: tube capacity of six and ability to carry items of any weight
- Careful: capacity of three but slower than other robots

With our new design, there are many improvements listed below:

- The old design would not support the fragile items causing breakage, where fragile items were assigned to non-careful robots, and the program would throw a fragile item broken exception because they all move floors simultaneously without the implementation of slow movement. The new design would be implementing careful robots to which fragile items would be assigned. This would avoid fragile item broken exceptions because only careful robots handle fragile items, and they move independently to non-careful robots with half speed of those.
- Non-careful robots would move independently compared to the careful robots. The non-careful robots take one turn calling the main class once and then the careful robot calls the main class twice to move one floor. This cycle repeats throughout the whole program. In essence, only either one of non-careful or careful robots calls the moveTowards() method at a time, and careful robots move at half speed of non-careful robots.
- The tube size was fixed to four in the previous design while in the new design, different robots have different tube sizes giving them enough flexibility. The previous design threw a tube full exception whenever the number of mail items of a tube attached to a robot reaches 4, as opposed to the new design where each type of robot has its own tube length allowing each robot to have different number of mail items based on their tube size. This would avoid tube full exceptions and excessive delivery exceptions.
- The constructor of object robot did not consider whether the robot kind was careful or non-careful. In the new added design, it analyses and identifies the robot type(E.g Boolean value strong is true if the robot can carry non-fragile items and Boolean value is strong will be false if the items are fragile. This change allows the program to implement robots with different features including the new and old ones. For example, careful robots have Boolean strong == true, Boolean careful == true and Boolean big == false, which is translated to careful robots can carry items of any weight including fragile items.
- The precious design would throw a fragile item broken exception when it finds out that a single robot has two fragile items. However, the new design handles these situations by recording whether tubes already have a fragile item.

Boolean strong = false (non-fragile items of weight under 2000 grams with tube size 4)
Boolean careful = true (carry any fragile items with half speed with tube size 3)

Some of the recommendations in order to make adding additional robot types easier in future, are

- Make the original robot class as a parent class and create more specified robot types as its children classes. In the current design, boolean variables, such as “strong”, “big” and “careful”, are used to control the behaviour of each robot type. However, if the number of robot types increases, it will be easier to create children classes than adding boolean variables for each additional robot type. This is because inheritance relationships enable a more readable and lower coupling coding style.
- Create different fillStorageTube() in MyMainPool class and addItem() in StorageTube class, for each robot type if the number of robot types increases. This leads to more readable coding style because “if, else if, and else” are used to identify each robot type and control their behaviour of each robot type in the current design. However, it will be more complicated and difficult to read the code, as the number of robot types increase.