CMPT 276 Assignment 3

Jinshuo Zhang, Sana Dallalzadeh Atoufi

For this assignment our group was further divided in two as per the instructions for this assignment. We decided to choose our groups based on two criteria, one being how familiar we were with that person's code and the second being how related the two classes/code segments are. As this would mean there is a better understanding between the two people about what the code should do and therefore resulting in better and more effective refactoring. For this reason we decided to do a code review, while the other two members of our group did the same. Everyone helped to implement each package in the game, but we mostly focused on the "Enemy.java" and the "Reward" package. So, we identified our code smells mostly in those packages and made the refactoring inside them. During this code review, we adjusted the code and tests related to these two packages. We found some problems, they can be divided into the following subtitles…

Code duplication:
In entity.Enemy class, we find many duplicate codes in these branches, which will make the function difficult to test them. So we changed them into four different functions which can be called in these branches.

Lack of Documentation:
In entity.Enemy class, there is a lack of documentation for the function enemyCheckCollision(). Missing documentation that explains the usage of this function. This function is used to check the tile 1 position away from this Entity is a wall tile for all 4 directions. If it is, set the variable enemyCollision value to True, and the direction to null in order to stop or help enemies to choose directions.

Unused or Useless Variables:
We find out there is an unused variable "tempGCost" under entity.Enemy class. This is a variable designed to contain the value of GCost, and pass this value to the testing function in order to test if the GCost function is working or not. But we used another way to test the GCost function, therefore this variable is unused and useless. It can be deleted now from the code.

Poorly Structured Code
In entity.Enemy class, under the function searchPath(int, int). This is the part of making decisions of going up, down, left, or right when the enemies are moving from one room to another room. We find out this is very hard to test all branches because variables are private and cannot be changed inside a function. And also, the function searchPath(int, int) try to do too much in one function. Therefore, we created a new function for enemies' moving, in order to make the different functions to do different things.

Unnecessary Use of Unsafe or Unsound Constructs

In RewardGenerator.java, we also made the array lists we used to store the Reward objects “final” since they are only being initialized once, which can make our data safer.

Poorly Structured Code

It is important to sort the imports to avoid unnecessary merge conflicts. Ordering the imports makes conflicts less likely. We find out that under entity.Enemy class and all classes under the reward package, imports are not well sorted, which may cause difficulty in debugging or analyzing the code. Therefore, we sorted the imports in both “Enemy.java” and “Reward” packages.