CMPT 276 Assignment 3 Report
Group 14: Darrick Gunawan, Mahyar Sharafi Laleh

During assignment 3, which takes place between March 21st and April 4th, the group for the project was split into smaller groups of 2 to refactor our codes. This is Darrick and Mahyar's report.

Overall Approach:

Given the long list of smells in the assignment file, our approach was pretty simple; go back to the code we each wrote and find atleast 3 code smells to fix. Each of us were to look through the code individually, identify a smell, fix it and rerun the program to ensure that it works and did not break anything.

Darrick's Refactor:

Commit: 23b7ee7ec4c00227b86312ec4166439a7bb67653

The first smell that was identified was in the player class. Being one of the first classes to be written, there was bound to be some refactoring needed in the code. The code that was refactored was dead code. A completely unused method named move() was still in the code, left over from the UML diagram even though we completely changed the way movement works during the implementation phase. Seeing as the code is not used anywhere else in the files, it was simply removed, and just to be safe, the program was ran again to ensure that the removal of code did not break anything.

Commit: 1d8f21204ada883af9231ef48356f025703d39ac

The second smell that was identified was in the Bonus class, which is a class for the bonus reward item. Seeing as this was also one of the first classes written due to it being a fundamental part of the game, there were bound to be some refactoring needed as it went through some changes to how methods and variables were used in the implementation phase. After looking through the class, an unused variable was identified. The unused variable was a 2d array named map, which would have been used to spawn the reward in that specific area of the map, however, that implementation was changed and the variable is now no longer used, so it has been removed. The program was once again ran just to ensure that the removal of the code did not break anything.

Commit: 8d9854048dd4af365079771cd33d0e8565a8e3df

The third smell that was identified was also in the Bonus class. As one of the first classes to be written, the documentation was not very good or sometimes even wrong, as implementations keep changing during phase 2 of the project. Because of this, there are a lot of instances of bad documentation for multiple methods in the class, which has all been fixed. An example was the functions were marked as an abandoned method, but it was clearly used and when I removed it the game would not work. Seeing as this was only editing comments, there is no need to test if anything would break as no actual code was changed.

Mahyar's Refactor:

1. **Refactor 1**

   Commit : f96b4bc3c6d439fbfa9c94c808df7b974cbb1c8d

   First smell I found was some poorly structured code in the regular rewards class. This class extends the rewards class which has x and y parameters. But x and y was initialized again within the regular class and made the regular class confusing due to the poor structure. By deleting the x and y from the regular class and using this.getX() and this.getY() methods from the super class I was able to get rid of this smell. After these changes the game ran as expected.

2. **Refactor 2 and 3**

   Commit: 1122e47ee3aa5853fbeee4a3f936b92d008a8e40

   The second smell consisted of more than 100 lines of dead code and unused variables in different classes. These lines of code made the classes more confusing and after deletion of these lines the game ran just like before but now our classes look more clean and concise.

3. The third bad smell I found was a couple of unused variables:
   a. First one was the speed variable within the Entity class. At first we were supposed to use this variable for our player and enemy classes but then changed the design to implement the speed variable for each class differently and forgot to delete this variable. After deleting this variable, I also found out our entity class is lacking getters and setters which I added them to the class code to make it more concise and understandable.
   b. Second instance was actually the two variables(prevX and prevY) in the enemy and player classes. This one also was forgotten to be deleted after a change in design. At first we implemented a wall collison method in which when the hitboxes of the enemy or the player collided with a hitbox of a wall it would return them to their previous position which would make the walls impenetrable by the player and the enemy class. But we figured out this design was buggy and changed it in a way that uses the map to see if the next tile the enemy and the player want to move to is a wall or not and if it is a wall the movement cannot be done. The implementation had changed but the code wa still there making the classes way bigger than they were supposed to be. The logic ran as expected after these deletions.

4. **Extra:**

   Commit: 60a4c7693ebcdbe461aef4ab7effcc91188de2d6

   As for an extra bad smell I found and solved was within the GameState interface. The methods were written as public but this was redundant as the class is an interface. Also found some more dead code in different classes and deleted them.