Assignment 3 Report

Samin Moradkhan - Vincente Buenaventura

We worked on the Refactoring of the project together. We figured out which parts of the code needed to be restructured, changed and modified to enhance the quality of the final product. We ensured that even after refactoring, our tests pass and there is no visible change to the observational behaviour of our code. We have continuously committed and pushed to the project repository.

Some of the problems that we noticed and fixed are listed below:

- **Poorly Structured Code**
  In the sound class, to return the index of which the desired sound effect is stored in the array, there was a function that would manually return the index which was a poor code design. Then I created setter and getter functions to set the index according to the sound effect and the get function to return the index for whichever sound effect is put in its input parameter. After this change I ran the SoundTest Class again, and all tests are fortunately passed hence no change has happened to the code behaviour.

- **Very Long Method in AssetCeator Called setObject**
  The setObject method was created to add all the objects including the banana, apple, traps, enemies and the door to the map and place them randomly. Having a few for loops right after each other with different conditions and numbers for each, made the setObject method very hard to look at and also unnecessarily long. So I divided the setObject function in 3 parts, addBanana, addApple and addTraps. In This way the function becomes more modular and in the case of the need to call any of the functions from another part of the code (for example for spawning apples randomly every 15 seconds) we can just use the existing function rather than having repetitive code to have the same functionality.

- **Bad/Confusing Variable Names**
  There was a variable called count in the Simulator class that is incremented in a loop (busy waiting) to place the apples on the map and then delete them after 15 seconds to create the bonus rewards. The name counter was not really a good fit because only by looking at the code in detail, the observer would understand the purpose. I changed the name to timer_counter so now it would be obvious that the variable is used for the timer.

- **Low Cohesion**
  In the Simulator class there is an update function that monitors the game continuously to check the state that we are in. it could be a win, lose or a pause state. And allows the user to play as long as it is in the play state. There was a for loop in that function that was responsible for adding the bonus rewards and deleting them however it was kind of unrelated to the purpose of the function. To make it more

understandable I created a separate method to keep track of the bonus rewards and I just call the function from the update function to make it more clear.

- **Code Duplication**
  In the banana and apple object classes, the main constructor method takes in two integers as parameters. One parameter is for the x coordinate and the other is for the y coordinate and they both are used to get the object placed on the map. Originally, there was a method constructor that did not have any parameters but the use of the method evolved into this new one. Some code in other classes did use the original constructor method but could be avoided by inputting random integers for the x and y coordinates. So I changed the calls from the original constructor to the new one and then deleted the original one. I ran the application and unit tests many times and the behaviour is the same as it was before.

- **Dead code**
  In the sound class, there was a method that would set the variable ‘sound_index’ to whatever integer that was passed through as an argument. However, searching throughout the whole source code, there is never once the method is used to set the variable to a different integer. Since this was the case, I deleted the whole method and ran the application along with the unit tests and everything seemed to be working just fine.

- **Useless variables**
  In the UI class, there was a method ‘drawTitle’ that had a string variable that wasn’t really being used or manipulated. It would be better to use the raw string value instead of storing it in a variable because the variable was just used to pass it through a method as an argument. No calculation or manipulation was used with the variable so I saw it was ‘useless’ to store it into a variable. Deleting the use of the variable would make the process of the method go faster too. After I removed the variable and changed all instances to the respective string values, I ran through the tests and the game itself and everything behaved normally.