CMPT 276: Introduction to Software Engineering

REPORT PHASE 3 - GROUP 09

Minh Phat Tran	301297286
Miles Burkholder	301405683

I. Introduction

In this homework 3, we have performed code review for our group project based on some ideas for things such as methods that are too long, low cohesion, high coupling, etc. Code reviewing can minimize our coding mistakes as well as promote coding standards throughout the project, which could reduce maintenance by making our project more readable and easier to debug and refractory the coding project. Developers are also easier to monitor project quality and requirements. During the in class manual code review we noticed several bad code smells. The largest and most striking of these bad smells was that our Maze.java class was very long and tried to do too much. As a result of this, our class was not cohesive, highly coupled, had lots of data clumps, and was overall poorly designed. As it is required,our group of 4 members will be divided into 2 subgroups. Our group includes Minh Phat Tran and Miles Burkholder. Therefore, we have created a separate branch for our own group named MilesMinhRefactoring in GitLab.

II. Improved features

Large class that tries to do to much:

Our maze class was much longer than our other classes and handled too many things such as graphics, game control logic, map creation, handling all timers, and checking collisions. The class ended up being very incohesive because it was trying to handle all this seemingly unrelated functionality, while also containing large data clumps for keeping track of various timers. To help solve this, we extracted a Map class. We made the map class contain the map reading, creation, and initialization of all our static environment objects (walls, eggs, traps etc). Since the maze class often used the screenData 2D array (game map) we made this variable a public attribute of the map class which allowed our game functionality to remain the same. By extracting a Map class we were also able to extract many data constants used to represent environment objects in our text file which we then used to transform shorts into objects.The Maze class is now slightly more cohesive focusing more on its other functionality while being shorter and easier to follow. The relevant commits for these changes are `8f1643e3 “map class for reading map, and initializing environment, extracted from maze class”, and d2a9aef1 “Extracted and removed initLevel, and readLevel from maze class, added map class to handle this”.`

Low cohesion in maze class:

To once again reduce the size of our maze class and improve the cohesiveness we extracted a gameTimer class to handle all timers for the game such as (bonus duration/respawn/deletion, trap duration, game playing time etc). The maze class used to handle the creation and calculations for all these timers, we created the gameTimer class to have nice interfacing functions that are very

accessible for the maze to use such as boolean functions that return game playing time or bool functions that return if a specified timer is up. By creating a timer class we were also to extract a large data clump of various timer variables from the maze class. The maze class is now more cohesive focusing less on creating timers and more so on graphics and game logic, it is also much easier to test timer functionality. The relevant commits for these changes are `dda25626 "added a timer class to handle all timer functionality for game, extracted from maze"` and `ab56facd "updated use of timer's to use gameTimer class, cleaned up timer variables etc"`.

Long function and duplicate code:

During our code review we noticed that the checkCollisions method in the maze class was very long and it contained some duplicate code particularly when deleting each type of bonus. To help with this code smell we created a small short function called deleteBonus which contained the duplicated code, by calling this method we were able to delete the duplicated code slightly reducing the size of the checkCollsions method and making it more readable. The relevant commit is `07d3077b "Add function deleteBonus to check the activities after getting bonus operation in the game"`.

Low cohesion and data clump in maze class:

We noticed that the maze class had many variables, some of which should not have belonged to the maze such as BONUSDURATION and TRAPDURATION, these finals made the maze class less cohesive as it should not be handling these times. By refactoring we removed these from maze and placed these variables in their own respective classes. The maze class now has fewer data clumps and the maze, freezeBonus, speedBonus, and trapBonus classes are much more cohesive and independent. We also noticed that we had unused imports so we deleted them resulting in easier to follow classes. The relevant commits for these changes are `1e5b5670 "Moved bonus, trap, speed duration to their own class, deleted redundant imported library"` and `f31bc259 "deleted unused imports and unused variables and old function calls before adding Map class"`.

III. Conclusion

In a nutshell, our group has identified all possible code smells in our project that we could improve to increase our coding standards. We also addressed the detected smells and performed refactoring such as creating new classes, creating functions to avoid duplicated code in our project. As a result of this, our game project has a better design, while becoming easier to debug and maintain in the future with minor mistakes.