CMPT 276 - Group 17
Assignment 4: Code Review

Created by:
Brendan Shen
Sheila Nicholson

We completed a manual code review. The following eight code smells were identified and subsequently refactored. Each identified code smell includes a write-up indicating how the underlying issue causing the code smell was rectified, including a reference to the applicable Git commit(s). All tests were re-run after each refactor to ensure that the observable behaviour of the program did not change.

1. **Code-smell identified: Code duplication in APlusPaper and Vortex classes**

APlusPaper and Vortex classes both had separate implementations for a similar action. The action was re-spawning the location of either itself (in the case of APlusPaper) or the hero (in the case of the Vortex). I noticed that both classes had code that created a new random position and checked for three conditions: 1) does the position correlate to a floor tile, 2) is there not an item currently present at the position, 3) is there, not a character (i.e. enemy or hero) present at the position. To deal with this duplicate code I extracted the two separate methods and moved them up to the Item superclass ultimately combining them into a single method. The new method in Item was named validSpawnPosition(). By doing this I was able to remove unnecessary duplicate code from the collisionAction method in Vortex and the UpdateItemScore method in APlusPaper (note: this method was renamed to reSpawnPosition during refactoring). To view these changes refer to commit 3b1700e.

2. **Code-smell identified: Code duplication in collisionChecker class**

During the previously mentioned refactoring (1.) I encountered more duplicate code in the CollisionChecker class within the methods isHeroIntersecting and isEnemyIntersecting. I decided to extract the methods and combine them into a new method named isCharacterIntersecting, which checks if the position of an Item parameter intersects with Hero or Enemy objects. To view these changes refer to commit 3b1700e.

3. **Code-smell identified: Confusing class hierarchy in the Items package**

When re-evaluating the class hierarchies I noticed that the RewardItem and PunishmentItem classes were redundant and confusing. The only unique information they were contributing to their subclasses was assigning the appropriate ItemType enum to the variable itemType. To fix this I had all of their children classes inherit from Item directly, essentially removing an unnecessary level of inheritance. I also noticed that the damagePoints and rewardBonus variables could be removed and replaced by a new variable scoreEffect created in the superclass Item. To facilitate this I set the itemType and scoreEffect variables in a newly created method of Item called setScoreEffect. The method setScoreEffect is called in the constructor of all subclasses of Item. By changing the class hierarchy I was also able to remove duplicate code in the collisionAction method. By redefining its basic functionality in the superclass most of the subclasses could call this version of the method directly. The subclasses that required special functionality (Vortex, Coffee) for collisionAction call super on the method and then override the method to add implementation that is unique to their class's special effect. To view these changes refer to commit 54c0254 and bdbdec8.

4. **Code-smell identified: Unnecessary switch/case statements and code duplication in checkTile and checkItem methods of the TileManager class**

I noticed that the checkTile method used a switch/case statement instead of an if/else statement as well as having duplicated code inside each case. The purpose of the duplicated code was to set flags for the position object passed into it concerning whether the player has reached the end of the level, collided with a wall, or walked over a smoke tile. I extracted the code repeated inside the switch/case statements into a method called setPositionTags and changed the switch/case statement to an if/if else/else statement instead. I then noticed that the checkItem method had a very similar case of code duplication within switch/case statements, so I also extracted its repeated code into a method called handleItemCollsion. To view these changes, refer to commits 8e4ba22 and bc07112.

5. **Code-smell identified: Dead code in BadTile class**

I saw that the BadTile and NeutralTile classes, which both extend the abstract Tile class, were never actually used in our final implementation of the game. We had originally intended for a mix of tiles that were beneficial to step on (GoodTiles), but did nothing when stepped on (NeutralTiles) and tiles that were not valid to step on and would decrease your score (BadTiles). Instead, we made it so that the only things that would decrease the player's score would be PunishmentItems, and therefore the classes went unused and took up space in the project. I remedied this by simply deleting the two classes. To view these changes, refer to commit 8a93ab8.

6. **Code-smell identified: Dead code in Tile class**

I found an unused method called getTileTypeIndex, which had not been used in our implementation. This method would have originally been used to get the index of the type of type passed in through the parameter in the Tile array created in TileManager. This feature was never used in our project and was therefore dead code. I remedied this by simply deleting the method. To view this change, refer to commit b28743a.

7. **Code-smell identified: Unjustified use of primitives in Tile class**

After fixing code-smell number 5, I noticed that all tile types were piggybacking off of the GoodTiile class and that the game and tests were identifying the type of tile any given index in the map Tile array by using a string variable field in the Tile class and setting it upon loading in the map data from a file. I remedied this by making new classes for FloorTile, EndTile, and NotSteppableTile which are used for the entrance tile, outer wall tiles and locker tiles. I then got rid of the old string-based tileType field, and all getter and setter methods related to it. Lastly, I refactored all functions and tests that relied on this string-based system to instead use instanceof to verify that a tile of a certain tile type. To view these changes, refer to commits b28743a and 11c13cc.

8. **Code-smell identified: Dead code in the Hero class**

During the code review, I noticed that the boolean "collectedAllRewardItems" in the Hero class was not being updated. When the variable was declared it was initialized as true. Following its initialization was a for loop where its state was never updated, followed by a conditional statement checking if it was true. Since it was never updated it would always be true. Therefore not only was it not contributing any functionality to the code it also had the potential to cause a fault in the program. Therefore the variable was removed. To view these changes refer to commit 54c0254.