Assignment 3 Report

Group Members:

- Regan Li
- Zhaoyue Yuan

Code Smell 1 - Low Cohesion: detectEnemyCatchPlayer method should be in CollisionDetector class
Commit ID: 3c2eca90

Problem: detectEnemyCatchPlayer is a method in the Screen class, used to check whether enemies have captured the player. According to the functional independence principle in the Lecture 7: Design Principles, we should always strive for high cohesion. Although the screen class contains general game logic, we specifically have a CollisionDetector logic class, which detectEnemyCatchPlayer would be much more suitable in. Therefore, we have moved this method into the CollisionDetector class, because the CollisionDetector class is responsible for detecting collision conflicts in game, such as if an enemy has made contact with the player.

Fix: Move detectEnemyCatchPlayer method from Screen class to CollisionDetector class, and update related method call code.

Code Smell 2 – Unused or Useless Variables: remove directionEntered variable from AStarFindPath
Commit ID: 3c2eca90

Problem: In the initial design, the directionEntered variable was used to record the direction of the next node to be reached. However, later changes were made so that the coordinates of the next node were directly used rather than the direction. Since the logic had been altered, the variable directionEntered became redundant. The directionEntered variable is not used anymore, so it needed to be removed.

Fix: Remove directionEntered variable from ExpansionList constructor in AStarFindPath.java file, and update the ExpansionList constructor call in AStarFindPath.solve method.

Code Smell 3 – Bad/Confusing Variable Names: Change name of AStarFindPath class to PathFinder
Commit ID: 3c2eca90

Problem: Poor variable names are confusing for a viewer that is not involved in the creation process, and may lead to difficulty understanding the logic behind methods. AStarFindPath is unsuitable as a class name because it outlines the algorithm used but not its actual purpose. Meanwhile, the new class name PathFinder is a noun that is more self explanatory and hence more suitable for class name.

Fix: Rename AStarFindPath to PathFinder

Code Smell 4 - Unsound use of Construct: Adjust the construct of PathFinder
Commit ID: 057d8c7c

> **Problem:** Unsound constructs may not produce errors, however they can be made more efficient or understandable among many other things. In the original design, a PathFinder object was created every time when the shortest path needed to be searched, resulting in a large amount of memory overhead. However, we only need to update the start and goal to find the shortest path which results in a more efficient method.
>
> **Fix:** PathFinder retains only one construct parameter maze.

Code Smell 5 - Poorly Structured Code: Change the void solve() method in PathFinder to LinkedList<int[]> shortestPath (int [] start, int [] goal)

Commit ID: 7bf364dc

> **Problem:** Poorly structured code can be detrimental when debugging or testing code in general because it can be difficult to understand, or simply because there is a better method of implementation. The original solve() method did not have parameters, and each path finding requires recreating the PathFinder object. Now, it is passed in start and goal instead. In addition, the original solve method did not have a return value. To obtain the path, you should also call the GetPath method. Now, only need to call shortestPath method to get path.
>
> **Fix:** Change method name, parameters, return values, and related call. After finishing computation of the optimal path, backward find all nodes visited from source to target.

Code Smell 6 - Method Benefits from Being Refactored: change public LinkedList<int[]> getPath() in PathFinder to private LinkedList<int[]> solutin2Path(ExpansionList solution) {

Commit ID: 7bf364dc

> **Problem:** While methods may not constitute errors, they can always be improved upon to become more logical and/or efficient. The original getPath method implementation has no parameters and uses the pathfinding results of the entire class for path calculation. After changing the pathfinding method to a single call, it is necessary to change the method to an algorithm that backtracks the path based on the target value.
>
> **Fix:** Cooperate with the shortestPath method to receive the target node as a parameter and trace back from the target node to the source node.