Code Refactor Report

John Wang

Allen Chen

Low cohesion:

The method setImage and setUp in zombie and player class does the same function.

Solution:

By adding a setup method in utility tool function, both player and zombie can call utility tool function setup method to get the png image from resources. By adding a if conditional statement the setup method can distinguish the difference between player and zombie and get images from different folders in the resource file.

Code duplication:

Duplicate code fragment in player and zombie's update classes.

Solution:

By extracting the code fragment, We created a new method in the utility tool where both update function calls. The function will check if the player or zombie is colliding with anything and if not the entity's x y coordinate will change according to keyboard input/method calculation.

Move the corresponding test in player and zombie test to utility test.

Lack of documentation:

Methods in trap class do not have enough documentation. We changed the image of the other two regular rewards so we added a regularReward2 class and regularReward3 class, but we forgot to add documentation for them.

Solution:

Wrote documentation for the hit method and trapDamage method in the trap class.

Wrote documentation for the second regular reward class as well as its constructor.

Wrote documentation for the third regular reward class as well as its constructor.

Dead code:

The GamePanel object in the special reward class was never used.

In the regular reward class, these following import statements were never used:

```
import java.util.*;
import com.CMPT276_Group1.project.object.*;
import javax.imageio.*;
import java.awt.image.BufferedImage;
```

Solution:

We deleted the GamePanel object in the special reward class and all the import statements in the regular reward class.

Unnecessary if/else or switch/case statements:

CollisonChecker stop entity method labelled rules' code block is redundant.

Solution:

In the stop entity method of collision checker replace switch case statements that spans 3 lines per case to 1 line per case:

```
switch (entity.direction) {
    case "up" -> entity.solidArea.y -= entity.speed;
    case "down" -> entity.solidArea.y += entity.speed;
    case "left" -> entity.solidArea.x -= entity.speed;
    case "right" -> entity.solidArea.x += entity.speed;
}
```