

CMPT 276 Assignment 3
Group 14 Refactoring

Hanxi Chen (301417307)
Boyu Zhang (301414438)

2.Code Duplication

```

public class rewards extends Items
{
    3 usages
    protected int value;
    3 usages
    private int x,y;

-  @RequestMapping(methods={RequestMethod.GET, RequestMethod.POST})
+  @RequestMapping(methods={RequestMethod.GET})
+  public class bonus extends rewards {
    18 18 private BufferedReader[] sprites;
    21 21 private boolean pickable;
    22 - private int x,y;
    23 22 private int currentFrame;
    24 24 private TileManager tileManager;
    25 24
    26 --- @Override public class bonus extends rewards {
    43 43 //
    44 44 public boolean isX, int y, int rewardWidth, int rewardHeight, int value, int lifespan, int
    45 45 requestTime, TileManager tileManager) {
    46 46 super(x, y, rewardWidth, rewardHeight, value);
    47 47 - this.x=y;
    48 48 - this.y=y;
    49 49 this.lifespan = lifespan;
    50 49 this.requestTime = requestTime;
    51 50 this.isRespawning = false;
    52 51
    53 --- @Override public class bonus extends rewards {
    180 180 //
    181 181 }
    182 181 }
    183 182 public boolean isRespawn() {
    184 183 return lifespan.currentFrameMillis() - creationTime == lifespan
    185 184 }
    186 185 @Override public void bonus extends rewards {
    187 186 setCreationTime() = lifespan + diffTime;
    188 187 setEvolutionTime() = lifespan + pOffset;
    189 188 currentFrame = lifespan.getEvolutionTime(), nextFrameTime();
    190 189 setEvolutionTimeTemplate(x,y, getEvolutionTime(), getEvolutionTime());
    191 190 setEvolutionTimeTemplate(this.getEvolutionTime(), this.getEvolutionTime(), getEvolutionTime());
    192 191 }
    193 192 }
    194 193 }
}

```

4. Long List of method parameters

```
private void HitTrapEnemy(Enemy enemy){...}

1 usage = dga48 + 2
private void HitChestBox(Items item){...}

1 usage = dga48 + 2
private void HitBonus(Items item){...}

//
* game running state runner
*/
no usages = 0
public void Run(){
    // Render the running state.
    while (true) {
        for(Enemy enemy : enemies){
            if(enemy instanceof MovingEnemy){
                HitMovingEnemy(enemy);
            }

            if(enemy instanceof TrapEnemy){
                HitTrapEnemy(enemy);
            }
        }

        for(Items item : Items){
            if(item instanceof regular){
                HitChestBox(item);
            }

            if(item instanceof bonus){
                HitBonus(item);
            }
        }
    }
}
```

```

1  @src/main/java/it/try4/TrapFrame.java
2
3  ...
4  @TUT-07-01-01 public class TrapFrame extends Enemy {
5
6      private int x;
7
8      private int y;
9
10     private int currentFrame;
11
12     ...
13
14     private static final int enemyWidth = 30;
15
16     private static final int enemyHeight = 17;
17
18     private static final int trapDamage = 1;
19
20     ...
21
22     // Enemy constructor with all fields as parameters
23     // @param x dimension x
24     // @param y dimension y
25     // @param enemyWidth trap enemy width
26     // @param enemyHeight trap enemy height
27     // @param damage trap enemy damage
28
29     ...
30
31     public TrapFrame(int x, int y, int enemyWidth, int enemyHeight, int damage) {
32         super(x, enemyWidth, enemyHeight, damage);
33
34     }
35
36     public TrapFrame(int x, int y) {
37         super(x, enemyWidth, enemyHeight, trapDamage);
38         this.x = x;
39         this.y = y;
40         currentFrame = 0;
41     }
42
43     ...
44 }

```

```

v src/main/java/GameStates/RunningState.java
...   @@ -45,6 +45,35 @@ public class RunningState extends JPanel implements Gam
45   45
46   46     private Clip gameMusicClip;
47   47
48   48     private void initEnemies(){
49   49         enemies = new ArrayList<>();
50   50         enemies.add(new MovingEnemy(600, 512, 32, 32, 1, 10000));
51   51         enemies.add(new TrapEnemy(160, 40));
52   52         enemies.add(new TrapEnemy(460, 60));
53   53         enemies.add(new TrapEnemy(400, 850));
54   54         enemies.add(new TrapEnemy(150, 180));
55   55         enemies.add(new TrapEnemy(700, 400));
56   56

```

Right: After

6.Bad/Confusing Variable Name

```
return isTileSolid(playerLeft, playerTop)
    || isTileSolid(playerRight, playerTop)
    || isTileSolid(playerLeft, playerBottom)
    || isTileSolid(playerRight, playerBottom);
return isTileWall(playerLeft, playerTop)
    || isTileWall(playerRight, playerTop)
    || isTileWall(playerLeft, playerBottom)
    || isTileWall(playerRight, playerBottom);
```

More explanation on the next page

1. Unused or Useless Variables

We set up a Player Class for the movable character (Player controlled) in the game. We originally planned to set fatal moving enemies and ordinary moving enemies, but we did not implement a variety of moving enemies during development. The deleted useless variables are designed to record whether the player can continue to receive damage after receiving damage. Currently in the game moving enemies that give damage as critical hits so these variables are no longer needed. So I delete the useless variables during refactoring.

2. Code Duplication

The reward system of the game has an abstract "reward" class, and "bonus" rewards should inherit the characteristics of the "reward" class. But we additionally overwrite the x and y coordinates of "bonus" reward class that the "reward" class should have. (The coordinates are used to set the position of the item on the map). During refactoring I delete the variables we overwrite.

3. Poorly Structure Code

When the game is running to detect the interaction between the player and the map elements, we originally stacked all the codes and all the conditions (cases) together in the running state class. After refactoring, the run method became concise and easy to understand, and all collisions Conditions are written as additional methods that will be called when necessary.

4. Long List of method parameters

Our original trap enemy had too many parameters, which resulted in the need to enter many lines of repetitive code when creating it. Since we only have one kind of fixed trap in the game, I set the length, width and height variables for it before the trap's constructor. In this way, when the game is running, we only need to input the x and y axis coordinates as parameters when we want to generate the trap.

5. Poorly Structure Code

In the running state class of our generated game, the original version has no structure and piles up all the code used to generate map elements. After the refactor, I created a separate method for the generator of each element. When it is necessary to change the generated element, I only need to go to the generator to change without adjusting the running class. At the same time the code becomes more understandable and manageable.

6. Bad/Confusing Variable Name

In our initial map design, the game had a variety of terrain and barriers, but during the development process only walls were kept as obstacles. The map is composed of a circle of walls and has a door as an exit. The method originally designed to detect whether the tile where the obstacle is located can be collided has not been changed in time, resulting in the code review process to think that "IsTileSolid" includes both collision monster and trap conditions. "isTileWall" becomes more understandable after refactoring.

Commit id: df72c3bfe6adbaed5f35bfb3be4db36a39709cf1

Gitlab Commit link: https://csil-git1.cs.surrey.sfu.ca/cmpt276s23_group14/276-project/-/commit/df72c3bfe6adbaed5f35bfb3be4db36a39709cf1