Assignment 3

For this assignment, we decided to look at the EndMenuLose and EndMenuWin classes in our game. We have found six code smells to fix among these two classes. The smells and our fixes are described below.

1. Confusing variable name - EndMenuLose - refactored endmenu
Before:

```
Image scaled = image.getScaledInstance(1000,800 ,
java.awt.Image.SCALE_SMOOTH);
```

The problem with this section of code is that it's unclear what scaled exactly means. To fix this problem, we will change to a more descriptive variable name.

After:

```
Image scaledImage = image.getScaledInstance(1000,800 ,
java.awt.Image.SCALE_SMOOTH);
```

2. Lack of documentation - EndMenuLose lines 96-100 - refactored endmenu
Before:

```
setUndecorated(true);
getContentPane().add((buttonsC));
this.getContentPane().setLayout(null);

setVisible(true);
```

It's hard to know what this code does, especially if the reader is unfamiliar with Java Swing. Adding a few comments explaining what this section of code does will help make the code more readable.

After

```
// Removes any decorations from the frame and adds the buttons
setUndecorated(true);
getContentPane().add((buttonsC));
this.getContentPane().setLayout(null);

setVisible(true);
```

3. Code Duplication - EndMenuLose and EndMenuWin - Delete EndMenuWin.java

These two classes are almost identical in code and have very similar functions. One fix might be to combine these into one class. Perhaps this can be done with an if statement checking

whether that player has won or lost. As most of the code is shared, only a couple things will have to be switched around.

We deleted one of the EndMenus and then added a parameter to the function to differentiate between if the player won or lost.

```
EndMenu(int plScore, boolean result){
```

We then added an if statement to display the right image.

```
try{
            if(result == true){
                backGround = new
ImageIcon(ImageIO.read(this.getClass().getResource("/WinBac.png")));
            }
            else{
                backGround = new
ImageIcon(ImageIO.read(this.getClass().getResource("/LoseBac.png")));
            }
        }
```

4. Poorly Organized Code - EndMenuLose - refactored endmenu

Before:

```
// Create elements to be added to the JFrame
        ClickButton startButton = new ClickButton("Main Menu");
        ClickButton restartButton = new ClickButton("Restart");
        ClickButton exitButton = new ClickButton("Exit Game");
        JLabel score = new JLabel("SCORE: " + plScore);
        score.setFont((new Font("Serif", Font.PLAIN, 50)));
        score.setForeground(Color.black);
        JLabel time = new JLabel("TIME: " + Timer.time);
        time.setFont((new Font("Serif", Font.PLAIN, 50)));
        time.setForeground(Color.black);
```

We picked this section of code as it's hard to read. It's just several lines of code in one big chunk. This can be solved by using line breaks to separate the Buttons, score and time into 3 chunks and using comments to clarify what the code does for each chunk.

After:

```
// Create elements to be added to the JFrame
        ClickButton startButton = new ClickButton("Main Menu");
        ClickButton restartButton = new ClickButton("Restart");
        ClickButton exitButton = new ClickButton("Exit Game");

        JLabel score = new JLabel("SCORE: " + plScore);
        score.setFont((new Font("Serif", Font.PLAIN, 50)));
        score.setForeground(Color.black);
```

```
JLabel time = new JLabel("TIME: " + Timer.time);
time.setFont((new Font("Serif", Font.PLAIN, 50)));
time.setForeground(Color.black);
```

5. Code Cleanup/Adding Comments - Throughout EndMenu - Additional comments in EndMenu.java

Several comments were added throughout EndMenu to make the code easier to read. Some comments were edited to further clarify the function of the code. Before some chunks of code were mostly uncommented which made the purpose of some code unclear. This includes new comments on lines 52,57,62,70,73,77,82,84,88 among others.

6. Useless variable - EndMenuLose- refactored endmenu
Before:

```
GameInstance test = new GameInstance(factory1);
```

Instead of having new GameInstance assigned to a variable that's new used, we can just call a new GameInstance

After:

```
new GameInstance(factory1);
```