Code Smells (Milestone 5)

1. Lazy class: Tower Interface

```java
package com.example.demo;

import javafx.scene.image.ImageView;

public interface Tower {
    int getX();
    int getY();
    ImageView getSprite();
    int[][] affects();
}
```

This class was at first implemented as an option for our towers to implement. In the end, we never used this class or referenced it to be used.
The class now has been deleted from the repo. This has removed the lazy class and thus the code smell.

2. Data class: PlayerInformation

```java
public PlayerInformation() {

}
public PlayerInformation(String name, String difficulty) {
    score = 0;
    PlayerInformation.name = name;
    PlayerInformation.difficulty = difficulty;
    if (difficulty.equals("Easy")) {
        money = 1750;
        health = 200;
    } else if (difficulty.equals("Medium")) {
        money = 1250;
        health = 150;
    } else {
        money = 1000;
        health = 100;
    }
}
```

This acted as a global reference for frequently used values in the program. It allowed us to avoid passing in player information as a parameter for other methods, so we do not have any methods with long parameter lists.

```java
public void checkDamage(int damaged, Stage primaryStage) {
    this.setHealth(this.getHealth() - damaged);
    this.setScore(this.getScore() - score);
    if (this.getHealth() <= 0) {
        LoseScreen newScene = new LoseScreen();
        newScene.start(primaryStage);
    }
    ((Text) (this.tInfo.getChildren().get(1))).setText("Health: " + this.getHealth());
}
```

Here is a new method that we added into PlayerInformation. This was originally within the enemy class. It is more appropriate to move it into PlayerInformation since the monument health itself was stored within it. Now PlayerInformation is no longer a data class, removing the code smell.

3. Shotgun surgery: WizardTower, ArcherTower, WarriorTower

```java
public WizardTower() {
    String difficulty = info.getDifficulty();
    if (difficulty.equals("Easy")) {
        cost = 750;
    } else if (difficulty.equals("Medium")) {
        cost = 875;
    } else {
        cost = 1000;
    }
}
```

```java
public boolean buyWarrTower() {
    int currMoney = info.getMoney();
    if (currMoney >= cost) {
        info.setMoney(currMoney - cost);
        return true;
    } else {
        return false;
    }
```

These 3 Classes essentially do the same thing and all of them inherit from the Tower class. The Tower class did not define any methods for related to buying Towers and later the methods were added into each class due to them having different parameters for cost. When making any changes to the buy tower method, we need to manually change each of these methods individually.

Code relating to these methods have been moved within the Tower parent class. This removes the code smell by no longer having to make these small changes across all the classes.

4. Duplicated code: Brute, Skeleton, Zombie

```
public Path createPath() {
    Path path = new Path();
    path.getElements().add(new MoveTo(150, 0));
    path.getElements().add(new VLineTo(450));
    path.getElements().add(new HLineTo(250));
    path.getElements().add(new VLineTo(650));
    path.getElements().add(new HLineTo(450));
    path.getElements().add(new VLineTo(50));
    path.getElements().add(new HLineTo(650));
    path.getElements().add(new VLineTo(830));
    path.getElements().add(new HLineTo(850));
    path.getElements().add(new VLineTo(450));
    path.getElements().add(new HLineTo(1650));

    return path;
}
```

The above code defines the exact path that all the enemies must follow. It appears in the 3 enemy classes without any change and can be moved into the Enemy parent class.
To solve this issue, the code has been moved to the Enemy parent class. Now the duplicate code only shows up once, removing the code smell.

5. Refused bequest: TowerClass – affects()

```
/*
int[][] affects() {
    for (int i = x - range; i < x + range; i++) {
        for (int j = y - range; j < y + range; j++) {

        }
    }
}
 */
```

The Tower parent class uses an interface with the method affects() that was set up for future milestones involving damaging enemies. However, no tower implemented or wanted this method, making it a refused bequest.
Fixing this code smell was achieved by removing this method from TowerClass. It has been replaced by more specific methods inside the specific tower classes.