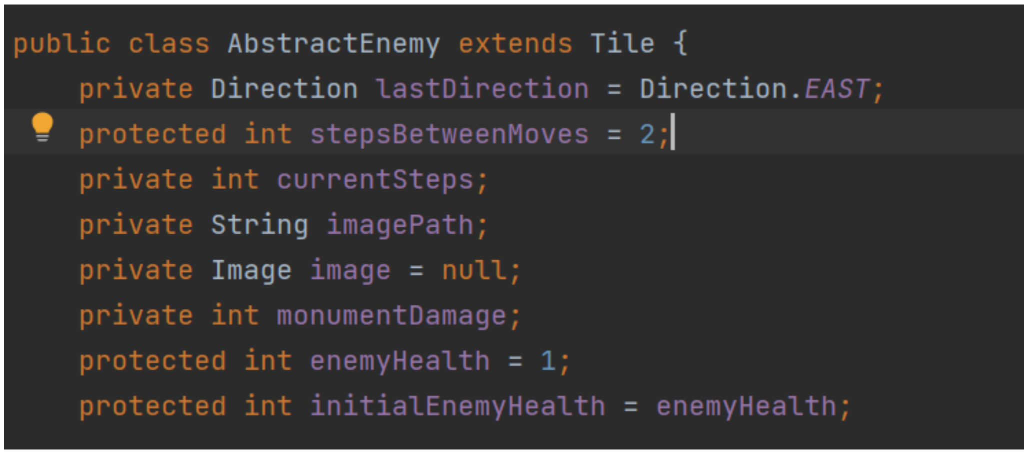
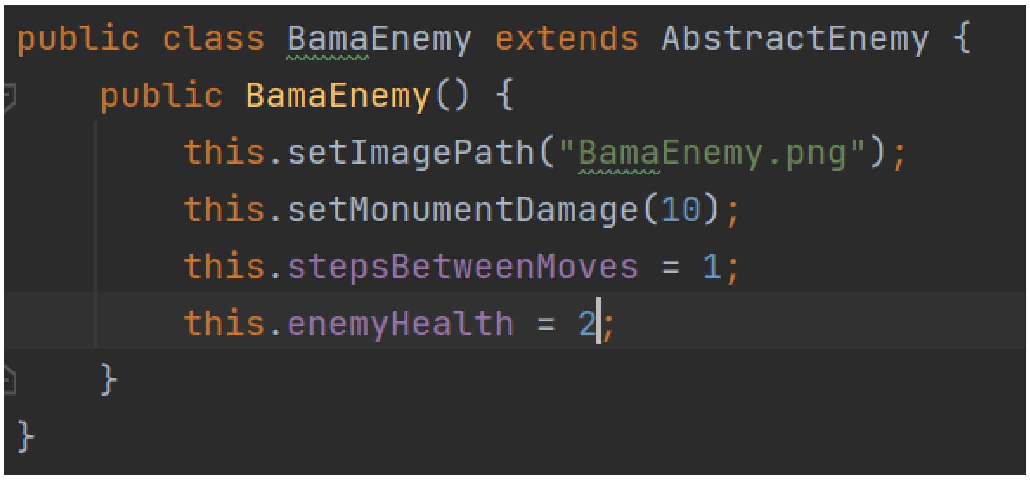
Code Smell - Refused Bequest - David Keefe

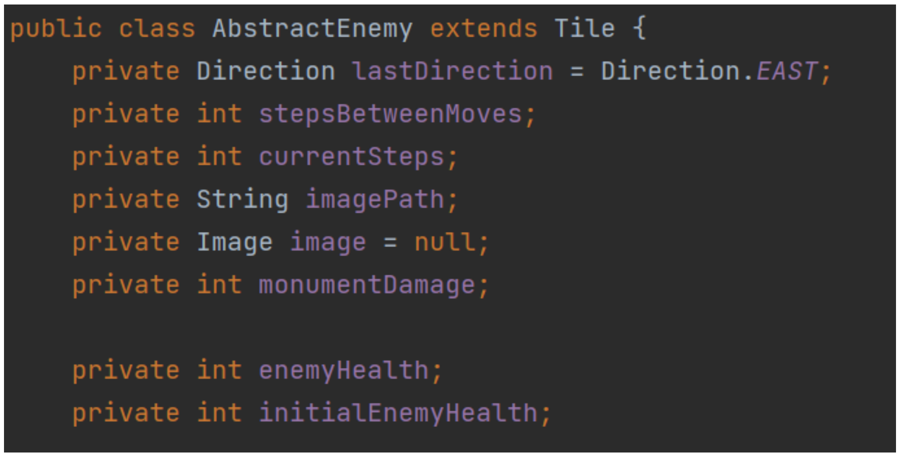


```java
public class AbstractEnemy extends Tile {
    private Direction lastDirection = Direction.EAST;
    protected int stepsBetweenMoves = 2;
    private int currentSteps;
    private String imagePath;
    private Image image = null;
    private int monumentDamage;
    protected int enemyHealth = 1;
    protected int initialEnemyHealth = enemyHealth;
```

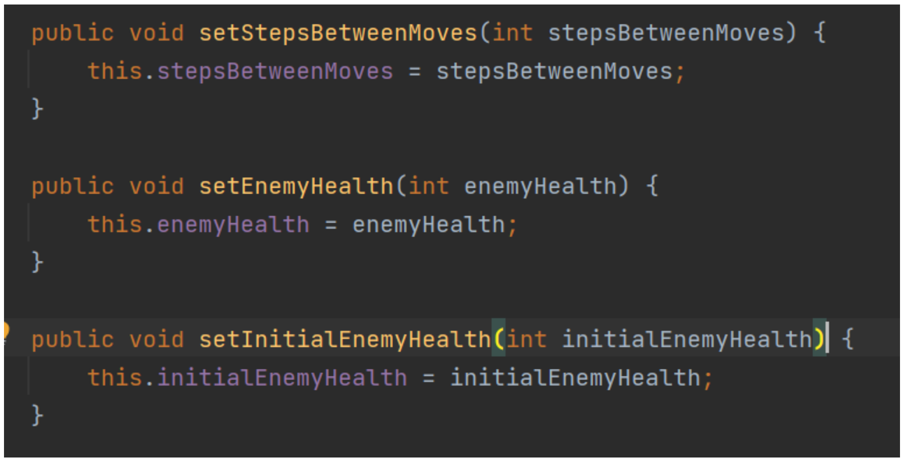


```java
public class BamaEnemy extends AbstractEnemy {
    public BamaEnemy() {
        this.setImagePath("BamaEnemy.png");
        this.setMonumentDamage(10);
        this.stepsBetweenMoves = 1;
        this.enemyHealth = 2;
    }
}
```

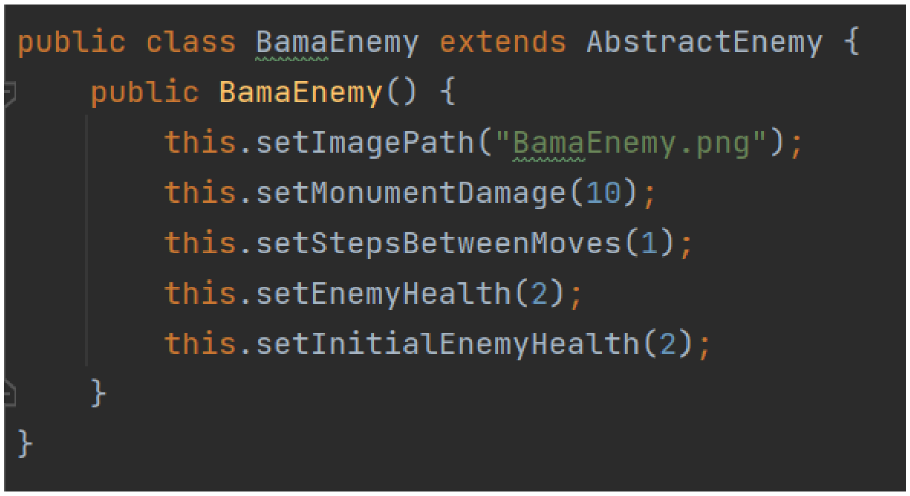
In our AbstractEnemy class, the *stepsBetweenMoves*, *enemyHealth*, and *initialEnemyHealth* variables were set as protected ints and initialized to specific values. However, these values did not apply to all of our enemy classes that extend AbstractEnemy, which required us to manually change the values within each enemy class. This is a clear example of refused bequest because the subclasses of AbstractEnemy were given data that they did not want and had to manually change. This code smell exists because the AbstractEnemy class was written before any of its subclasses, and those values were hardcoded in before it was decided that each enemy may have different values for these variables.



```java
public class AbstractEnemy extends Tile {
    private Direction lastDirection = Direction.EAST;
    private int stepsBetweenMoves;
    private int currentSteps;
    private String imagePath;
    private Image image = null;
    private int monumentDamage;

    private int enemyHealth;
    private int initialEnemyHealth;
```



```java
public void setStepsBetweenMoves(int stepsBetweenMoves) {
    this.stepsBetweenMoves = stepsBetweenMoves;
}

public void setEnemyHealth(int enemyHealth) {
    this.enemyHealth = enemyHealth;
}

public void setInitialEnemyHealth(int initialEnemyHealth) {
    this.initialEnemyHealth = initialEnemyHealth;
}
```



```java
public class BamaEnemy extends AbstractEnemy {
    public BamaEnemy() {
        this.setImagePath("BamaEnemy.png");
        this.setMonumentDamage(10);
        this.setStepsBetweenMoves(1);
        this.setEnemyHealth(2);
        this.setInitialEnemyHealth(2);
    }
}
```

I fixed this code smell by making the three variables private, removing their initializations within AbstractEnemy, and creating setter methods that allow each enemy to set their values. This is the same way that ImagePath and MonumentDamage are set within these classes, so it made sense to match the procedure that was already in place. This change eliminated the existence of refused bequest between AbstractEnemy and its subclasses.