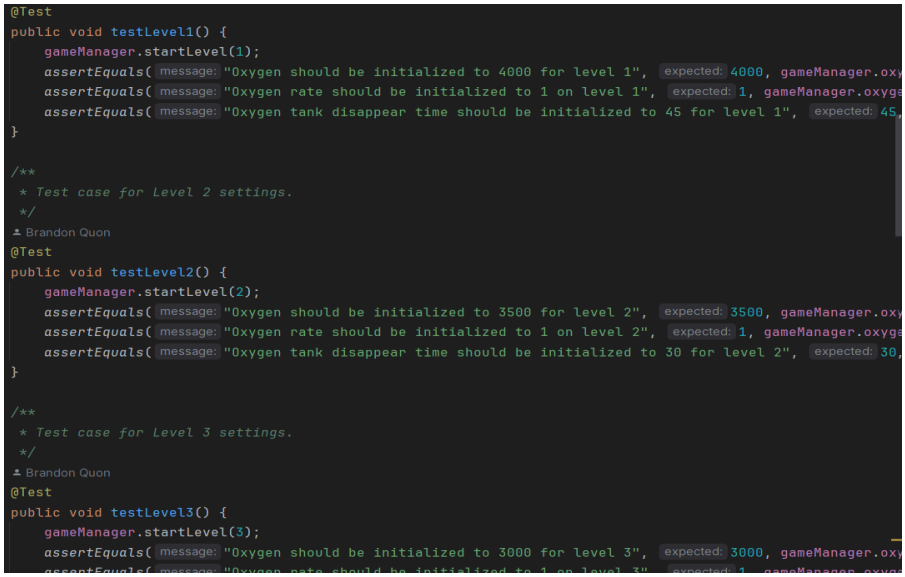
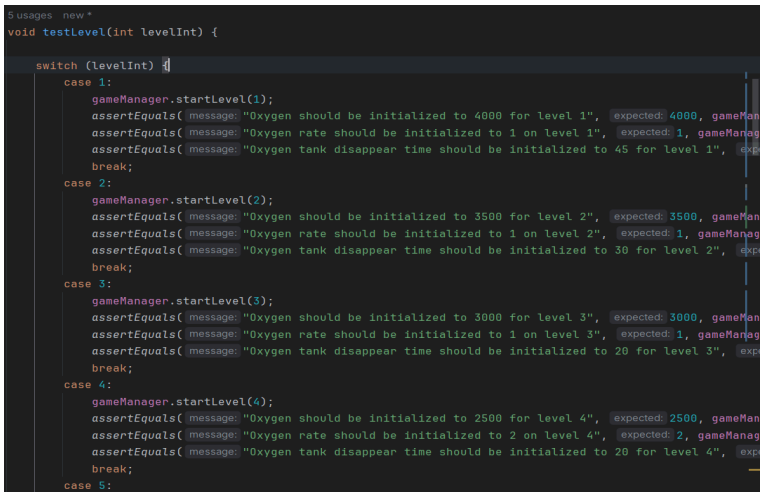
CodeReview - Sepehr Borji, Brandon Quon

1. Commit made: Code Smell: TestGameManager.java: Multiple functions to perform very similar tasks
   Before:

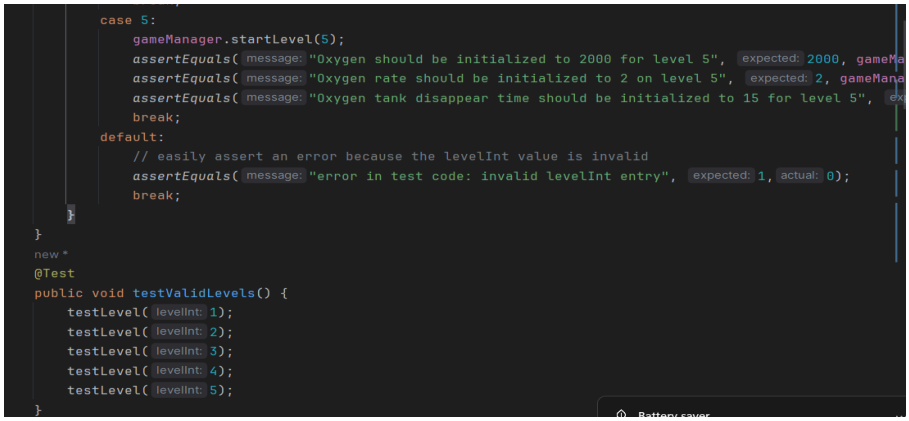


```
@Test
public void testLevel1() {
    gameManager.startLevel(1);
    assertEquals( message: "Oxygen should be initialized to 4000 for level 1", expected: 4000, gameManager.oxy
    assertEquals( message: "Oxygen rate should be initialized to 1 on level 1", expected: 1, gameManager.oxyg
    assertEquals( message: "Oxygen tank disappear time should be initialized to 45 for level 1", expected: 45,
}

/**
 * Test case for Level 2 settings.
 */
 Brandon Quon
@Test
public void testLevel2() {
    gameManager.startLevel(2);
    assertEquals( message: "Oxygen should be initialized to 3500 for level 2", expected: 3500, gameManager.oxy
    assertEquals( message: "Oxygen rate should be initialized to 1 on level 2", expected: 1, gameManager.oxyg
    assertEquals( message: "Oxygen tank disappear time should be initialized to 30 for level 2", expected: 30,
}

/**
 * Test case for Level 3 settings.
 */
 Brandon Quon
@Test
public void testLevel3() {
    gameManager.startLevel(3);
    assertEquals( message: "Oxygen should be initialized to 3000 for level 3", expected: 3000, gameManager.oxy
```

   How to fix this? The first thing to do is to create one function for testing the levels and using a switch statement to go to the proper case based on the level selected. We also make another function which is the Test, this test function calls the function with the switch statements and assesses each condition. Now we have 2 functions instead of 5, more flexibility to extend, increased readability, and we are exercising the principle of DRY - don't repeat yourself.
   After:



```
5 usages  new *
void testLevel(int levelInt) {

    switch (levelInt) {
        case 1:
            gameManager.startLevel(1);
            assertEquals( message: "Oxygen should be initialized to 4000 for level 1", expected: 4000, gameMan
            assertEquals( message: "Oxygen rate should be initialized to 1 on level 1", expected: 1, gameManag
            assertEquals( message: "Oxygen tank disappear time should be initialized to 45 for level 1", exp
            break;
        case 2:
            gameManager.startLevel(2);
            assertEquals( message: "Oxygen should be initialized to 3500 for level 2", expected: 3500, gameMan
            assertEquals( message: "Oxygen rate should be initialized to 1 on level 2", expected: 1, gameManag
            assertEquals( message: "Oxygen tank disappear time should be initialized to 30 for level 2", exp
            break;
        case 3:
            gameManager.startLevel(3);
            assertEquals( message: "Oxygen should be initialized to 3000 for level 3", expected: 3000, gameMan
            assertEquals( message: "Oxygen rate should be initialized to 1 on level 3", expected: 1, gameManag
            assertEquals( message: "Oxygen tank disappear time should be initialized to 20 for level 3", exp
            break;
        case 4:
            gameManager.startLevel(4);
            assertEquals( message: "Oxygen should be initialized to 2500 for level 4", expected: 2500, gameMan
            assertEquals( message: "Oxygen rate should be initialized to 2 on level 4", expected: 2, gameManag
            assertEquals( message: "Oxygen tank disappear time should be initialized to 20 for level 4", exp
            break;
        case 5:
```



2. Commit made: Code Smell: HighScoreManagerTests.java: function testGetScores() could be more readable
   Before:

```
@Test
public void testGetScores() {
    scoreboard.updateScoreboard( score: 100,  level: 6);
    scoreboard.updateScoreboard( score: 200,  level: 6);
    scoreboard.updateScoreboard( score: 300,  level: 6);
    assertEquals( expected: "300\n200\n100\n", scoreboard.getScores( level: 6));
    scoreboard.clearTestFile();
}
```

   After:

```
@Test
public void testGetScores() {
    int levelToTest = 6;

    scoreboard.updateScoreboard( score: 100, levelToTest);
    scoreboard.updateScoreboard( score: 200, levelToTest);
    scoreboard.updateScoreboard( score: 300, levelToTest);

    String expectedOutput = "300\n200\n100\n";
    String actualOutput = scoreboard.getScores(levelToTest);

    assertEquals(expectedOutput, actualOutput);
    scoreboard.clearTestFile();
}
```

   As the code shows, the parameters have just been given local variable names so it is easier to understand the code's intention at a glance. When the code is more readable, it is easier for others to use it and understand it. It's also less error-prone to use the variable multiple times than to type in the literal because a simple typo will result in a logic error, but the compiler will catch a typo with the variable name. Testing is done after each change to ensure refactoring does not cause differences in functionality.

3. Commit made: Code smell: TestGameManager.java: setup() function contains unused variable
   Before:

```
/**
 * Set up method to initialize GameManager.GameManager and mock dependencies.
 */
 Brandon Quon +1
@Before
public void setUp() {
    // Create a mock for Gameobject
    Gameobject gameobjectMock = mock(Gameobject.class);

    // Mock the static init method
    Gameobject.init(mock(Processing.class));

    // Initialize GameManager.GameManager
    gameManager = new GameManager();
}
```

Hovering over the green squiggly line under “gameobjectMock” tells us this variable is unused and we can verify this by the fact that it is a locally declared variable and doesn’t get used in the only scope in which it exists. It is sufficient to remove this variable.
How did this happen? Notice how the function has been modified by more than one person. The intention of why this variable exists either got lost in communication or a fix was made where this variable was no longer required and it just wasn’t removed. Either way for ease of clarity to anyone else who reads this code, it's best to just remove this variable as it is not doing anything.
After:

```
/**
 * Set up method to initialize GameManager.GameManager and mock dependencies.
 */
 Brandon Quon +1 *
@Before
public void setUp() {
    // Mock the static init method
    Gameobject.init(mock(Processing.class));

    // Initialize GameManager.GameManager
    gameManager = new GameManager();
}
```