Name: Louden Demers

Questions

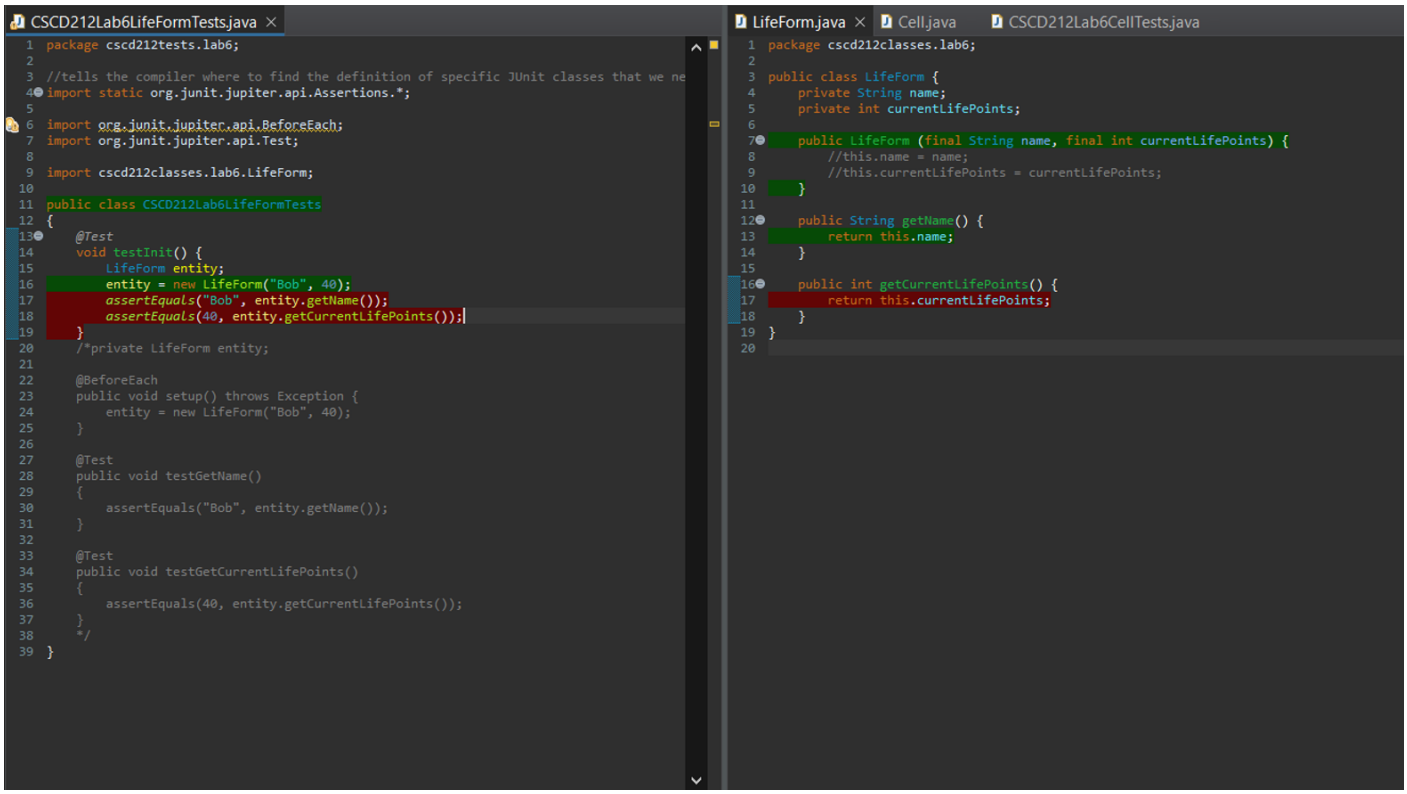
1) The test DOES violate the Single Responsibility Principle because it tests more than one thing. The code actually tests both the getName() and getCurrentLifePoints() methods, when there should be two separate tests for this.
2) Copying the code into another test WILL violate the Code Smell of duplicated code because each test will still contain the declaration of the variable *entity* and the assignment of *entity*. This can be fixed by doing these actions at the top of the class before all tests are run.

***NOTE: The method being focused on in each screenshot is typically the one closest to the bottom of the program**

Screenshots (descriptions above each one or pair of fail/pass)

Screenshots Before Group 1

Getting LifeFormTests to compile



CSCD212Lab6LifeFormTests.java

```java
package cscd212tests.lab6;

//tells the compiler where to find the definition of specific JUnit classes that we ne
import static org.junit.jupiter.api.Assertions.*;

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import cscd212classes.lab6.LifeForm;

public class CSCD212Lab6LifeFormTests
{
    private LifeForm entity;

    @BeforeEach
    public void setup() throws Exception {
        entity = new LifeForm("Bob", 40);
    }

    @Test
    public void testGetName()
    {
        assertEquals("Bob", entity.getName());
    }

    @Test
    public void testGetCurrentLifePoints()
    {
        assertEquals(40, entity.getCurrentLifePoints());
    }
}
```

LifeForm.java

```java
package cscd212classes.lab6;

public class LifeForm {
    private String name;
    private int currentLifePoints;

    public LifeForm (final String name, final int currentLifePoints) {
        this.name = name;
        this.currentLifePoints = currentLifePoints;
    }

    public String getName() {
        return this.name;
    }

    public int getCurrentLifePoints() {
        return this.currentLifePoints;
    }
}
```

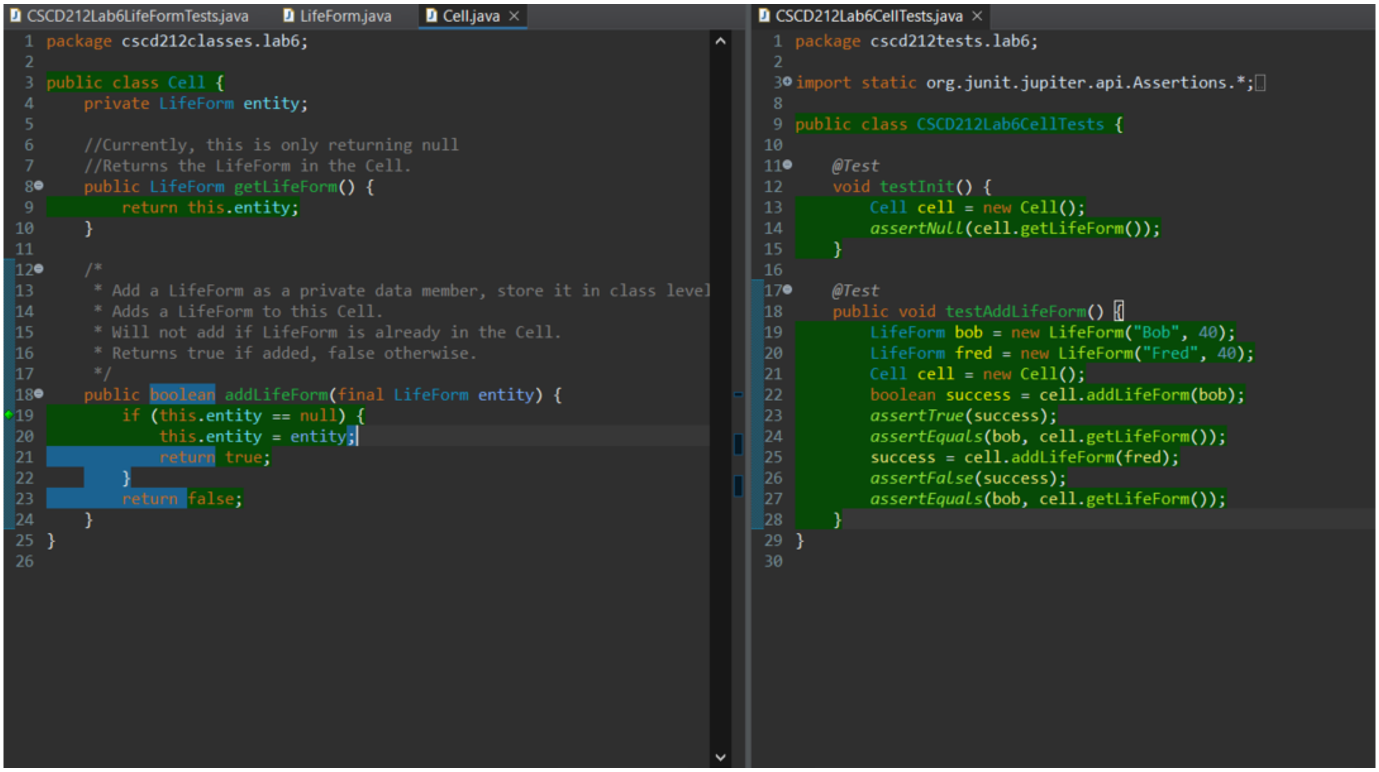
Testing addLifeForm in CellsTests

Cell.java

```java
package cscd212classes.lab6;

public class Cell {
    private LifeForm entity;

    //Currently, this is only returning null
    //Returns the LifeForm in the Cell.
    public LifeForm getLifeForm() {
        return this.entity;
    }

    /*
     * Add a LifeForm as a private data member, store
     * Adds a LifeForm to this Cell.
     * Will not add if LifeForm is already in the Cell
     * Returns true if added, false otherwise.
     */
    public boolean addLifeForm(final LifeForm entity)
        return false;
    }
}
```

CSCD212Lab6CellTests.java

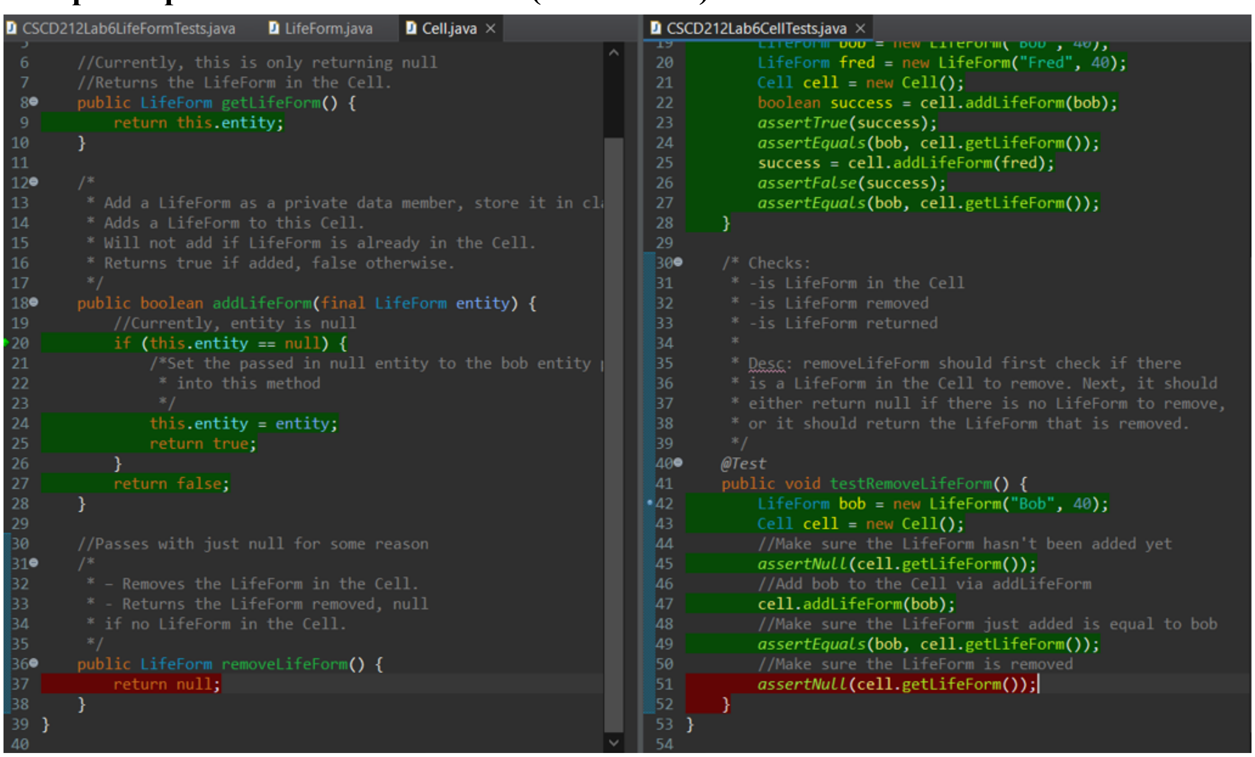
```java
package cscd212tests.lab6;

import static org.junit.jupiter.api.Assertions.*;

public class CSCD212Lab6CellTests {

    @Test
    void testInit() {
        Cell cell = new Cell();
        assertNull(cell.getLifeForm());
    }

    @Test
    public void testAddLifeForm() {
        LifeForm bob = new LifeForm("Bob", 40);
        LifeForm fred = new LifeForm("Fred", 40);
        Cell cell = new Cell();
        boolean success = cell.addLifeForm(bob);
        assertTrue(success);
        assertEquals(bob, cell.getLifeForm());
        success = cell.addLifeForm(fred);
        assertFalse(success);
        assertEquals(bob, cell.getLifeForm());
    }
}
```

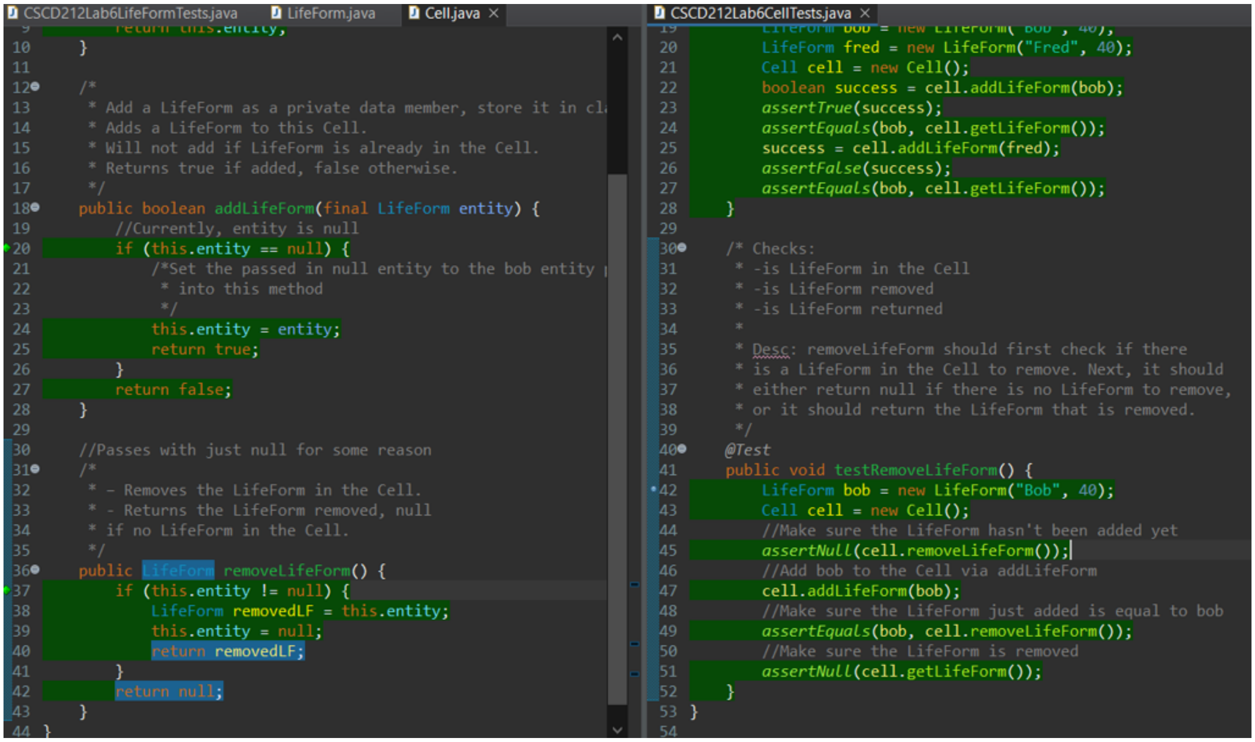


~~~~~~

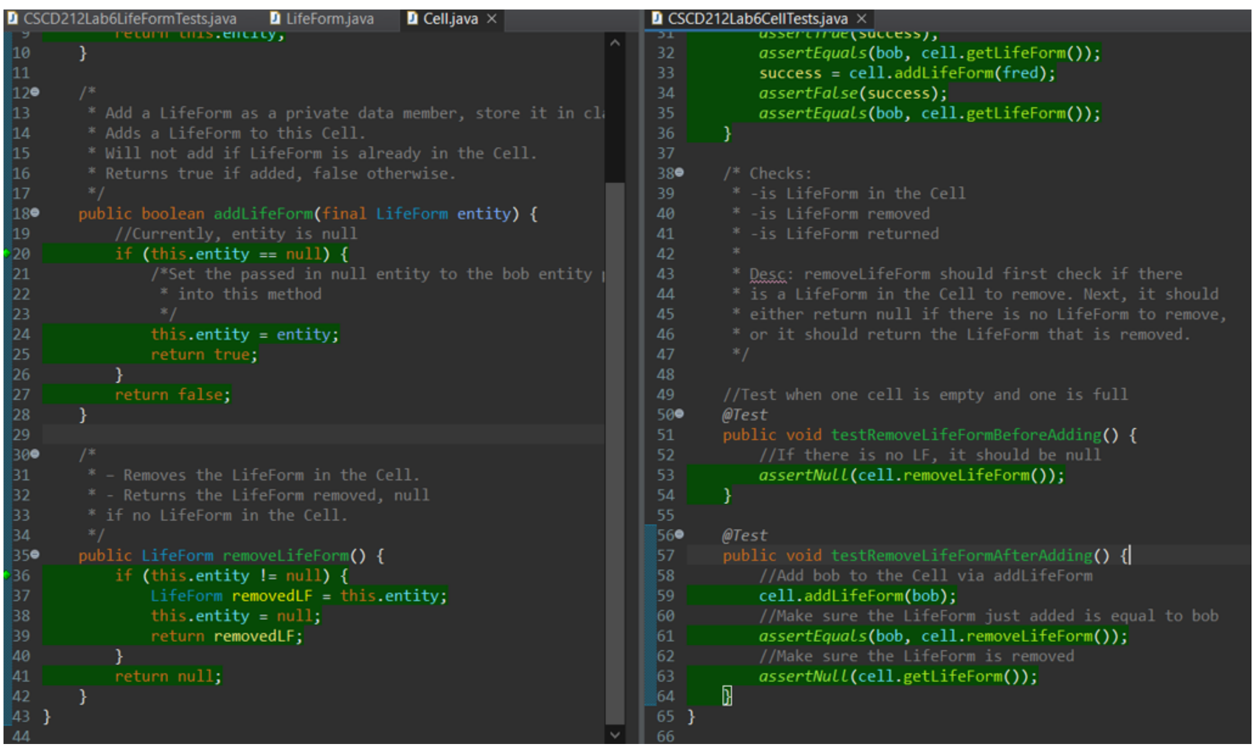
<u>Group 1 Screenshots</u>

Group 1 Step 4 - testRemoveLifeForm (Fail -> Pass)


~~~~~~



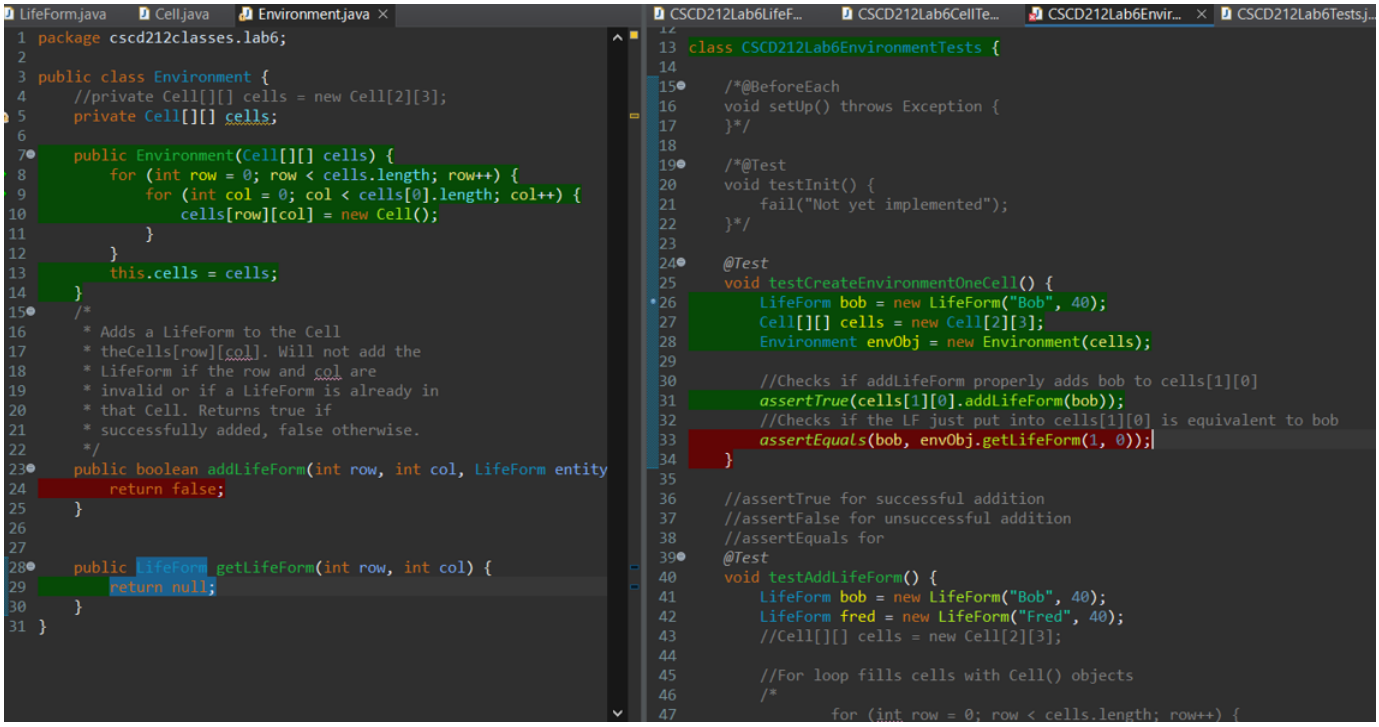
Group 1 Step 11 - testRemoveLifeForm (Refactored)



~~~~~~

<u>Group 2 Screenshots</u>

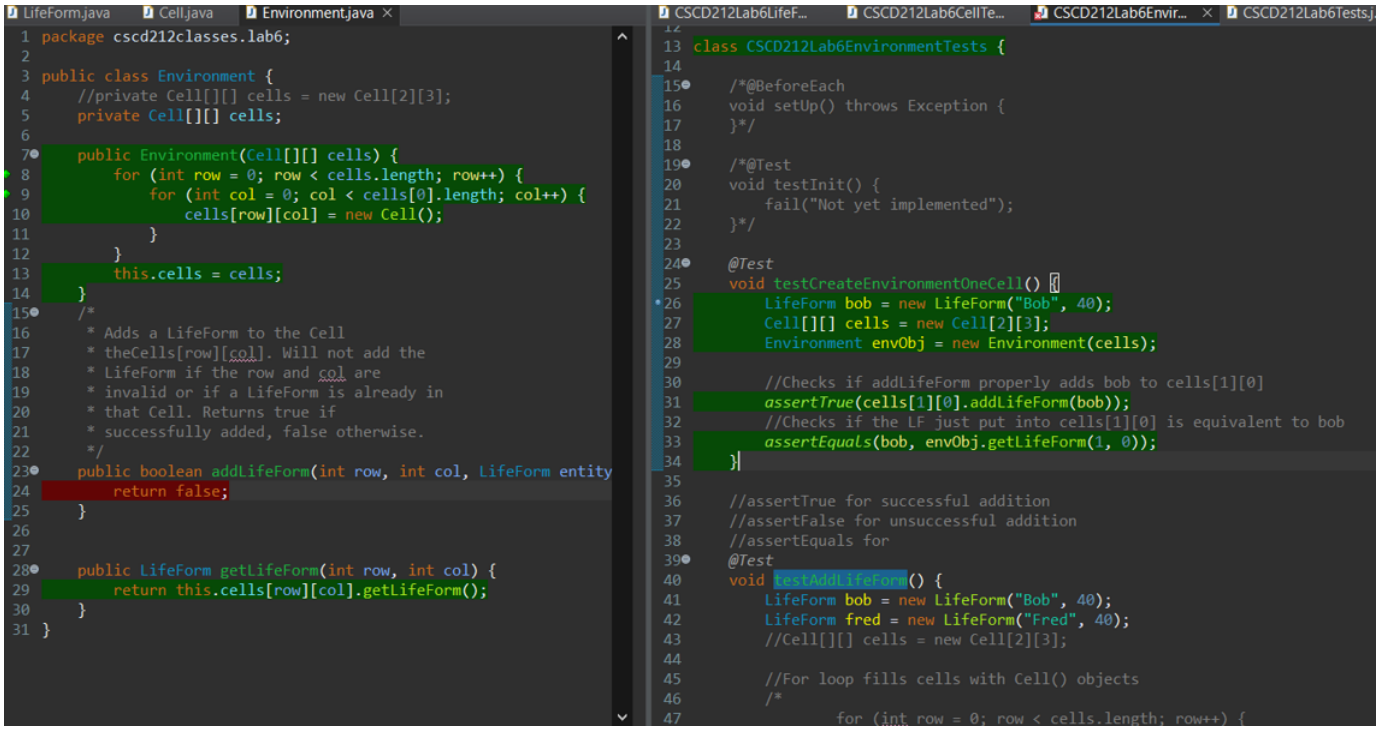
Group 2 Step 3 - Basic Environment Test (Fail)
~~~~~~



```java
package cscd212classes.lab6;

public class Environment {
    //private Cell[][] cells = new Cell[2][3];
    private Cell[][] cells;

    public Environment(Cell[][] cells) {
        for (int row = 0; row < cells.length; row++) {
            for (int col = 0; col < cells[0].length; col++) {
                cells[row][col] = new Cell();
            }
        }
        this.cells = cells;
    }
    /*
     * Adds a LifeForm to the Cell
     * theCells[row][col]. Will not add the
     * LifeForm if the row and col are
     * invalid or if a LifeForm is already in
     * that Cell. Returns true if
     * successfully added, false otherwise.
     */
    public boolean addLifeForm(int row, int col, LifeForm entity
        return false;
    }


    public LifeForm getLifeForm(int row, int col) {
        return null;
    }
}
```

```java
class CSCD212Lab6EnvironmentTests {

    /*@BeforeEach
    void setUp() throws Exception {
    }*/

    /*@Test
    void testInit() {
        fail("Not yet implemented");
    }*/

    @Test
    void testCreateEnvironmentOneCell() {
        LifeForm bob = new LifeForm("Bob", 40);
        Cell[][] cells = new Cell[2][3];
        Environment envObj = new Environment(cells);

        //Checks if addLifeForm properly adds bob to cells[1][0]
        assertTrue(cells[1][0].addLifeForm(bob));
        //Checks if the LF just put into cells[1][0] is equivalent to bob
        assertEquals(bob, envObj.getLifeForm(1, 0));
    }

    //assertTrue for successful addition
    //assertFalse for unsuccessful addition
    //assertEquals for
    @Test
    void testAddLifeForm() {
        LifeForm bob = new LifeForm("Bob", 40);
        LifeForm fred = new LifeForm("Fred", 40);
        //Cell[][] cells = new Cell[2][3];

        //For loop fills cells with Cell() objects
        /*
                for (int row = 0; row < cells.length; row++) {
```

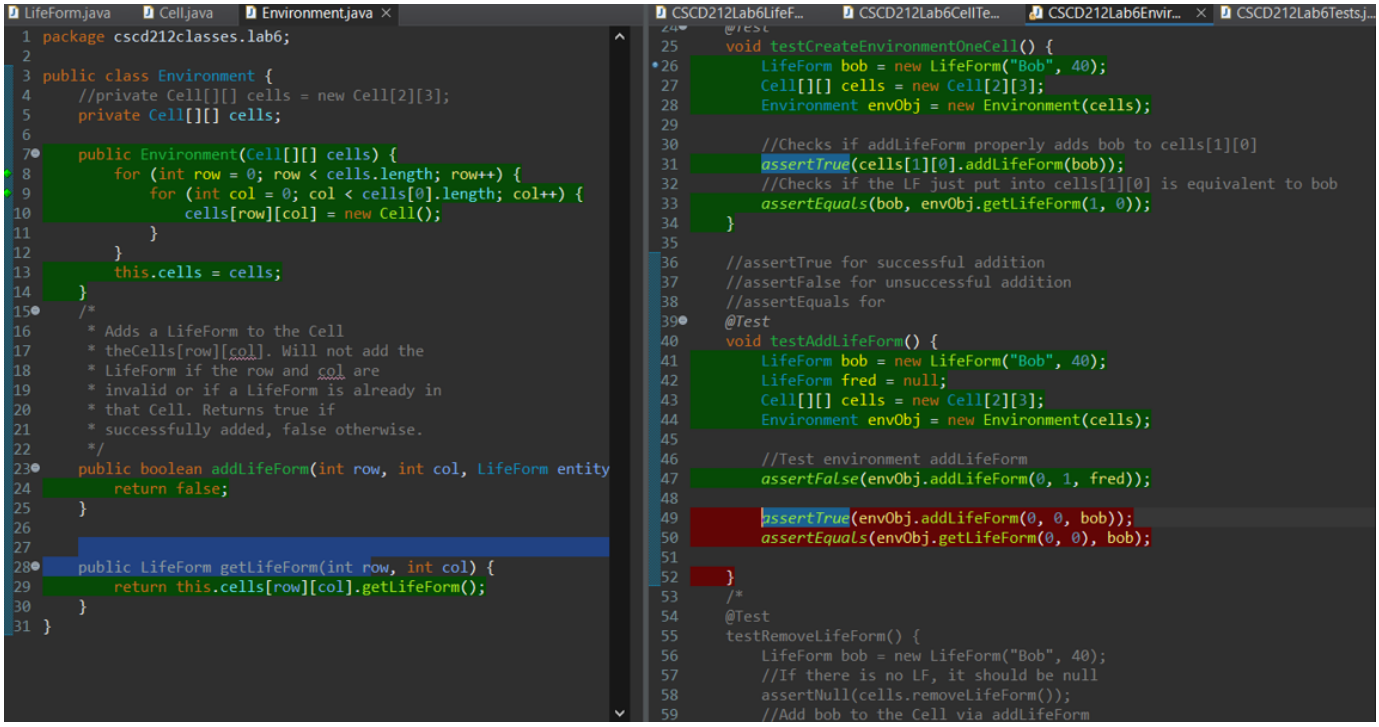
Group 2 Step 5 - Basic Environment Test (Pass) *ignore addLifeForm here*

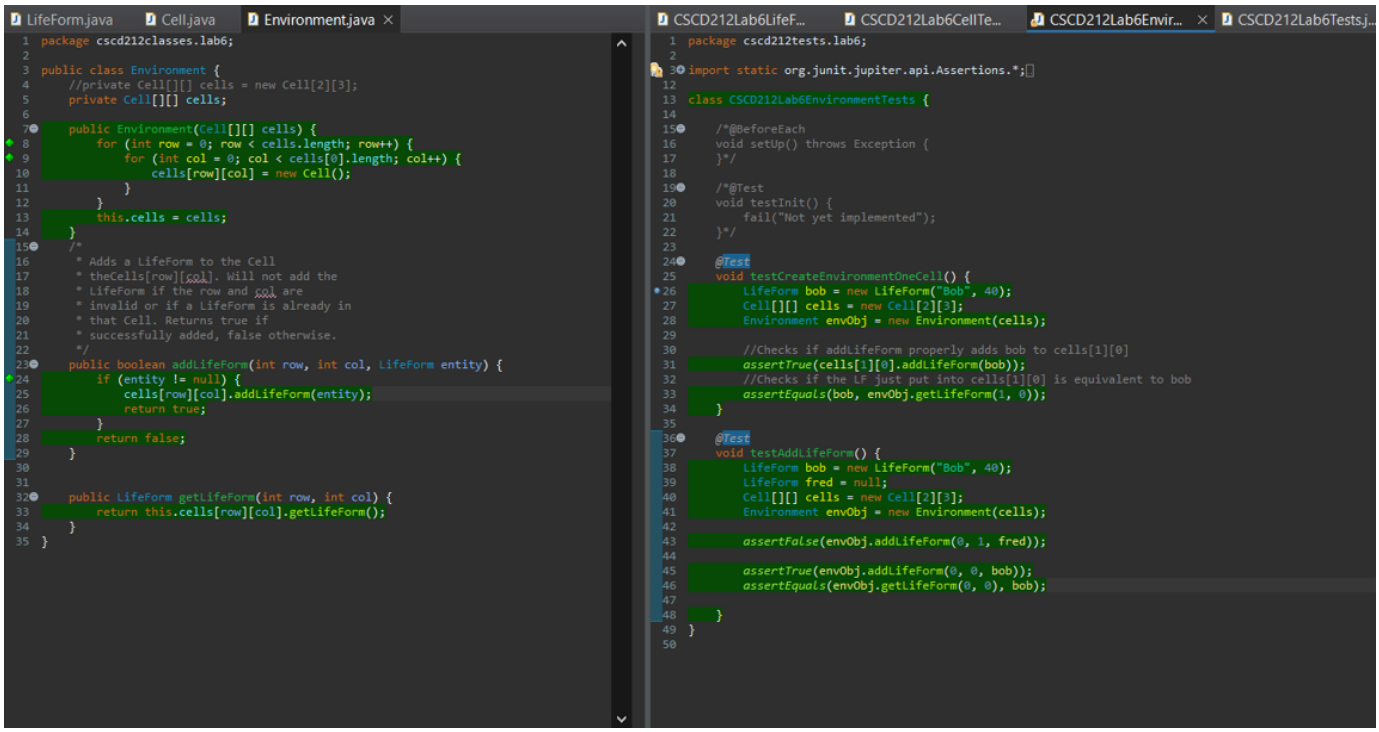


```java
package cscd212classes.lab6;

public class Environment {
    //private Cell[][] cells = new Cell[2][3];
    private Cell[][] cells;

    public Environment(Cell[][] cells) {
        for (int row = 0; row < cells.length; row++) {
            for (int col = 0; col < cells[0].length; col++) {
                cells[row][col] = new Cell();
            }
        }
        this.cells = cells;
    }
    /*
     * Adds a LifeForm to the Cell
     * theCells[row][col]. Will not add the
     * LifeForm if the row and col are
     * invalid or if a LifeForm is already in
     * that Cell. Returns true if
     * successfully added, false otherwise.
     */
    public boolean addLifeForm(int row, int col, LifeForm entity
        return false;
    }


    public LifeForm getLifeForm(int row, int col) {
        return this.cells[row][col].getLifeForm();
    }
}
```

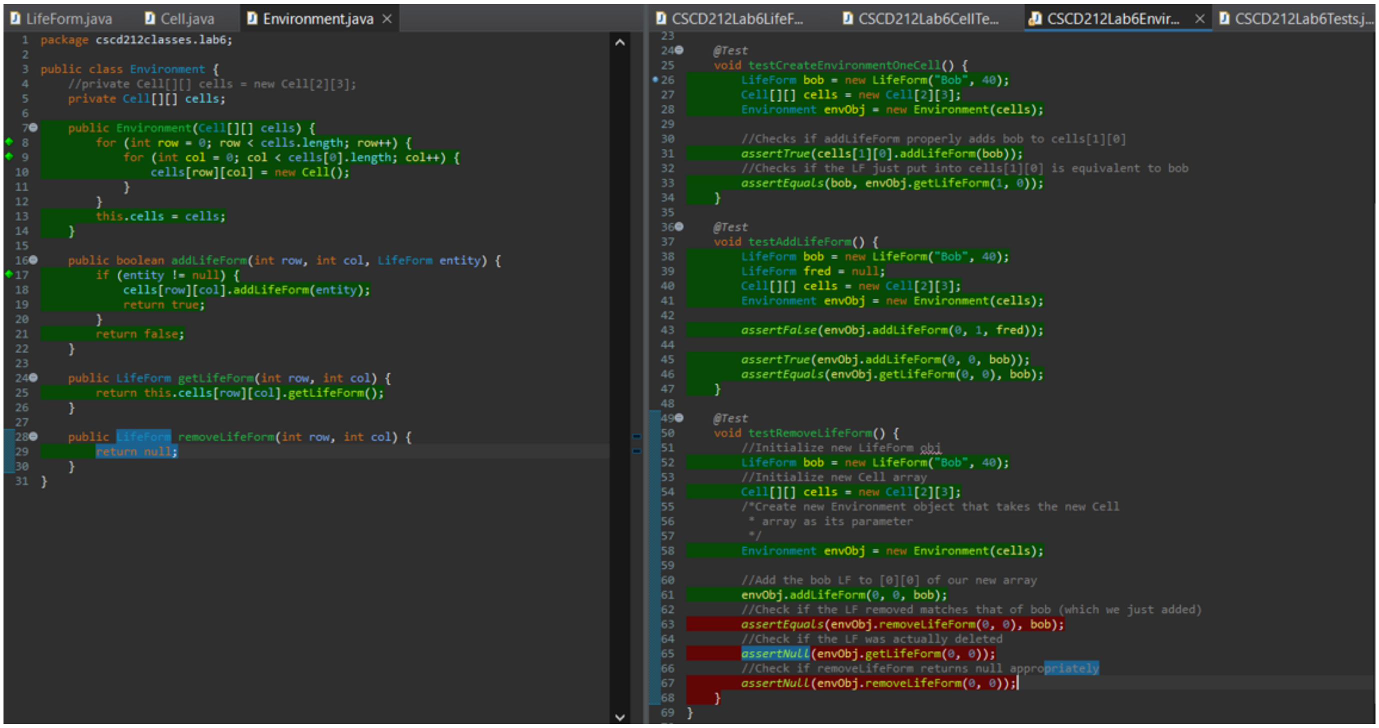
```java
class CSCD212Lab6EnvironmentTests {

    /*@BeforeEach
    void setUp() throws Exception {
    }*/

    /*@Test
    void testInit() {
        fail("Not yet implemented");
    }*/

    @Test
    void testCreateEnvironmentOneCell() {
        LifeForm bob = new LifeForm("Bob", 40);
        Cell[][] cells = new Cell[2][3];
        Environment envObj = new Environment(cells);

        //Checks if addLifeForm properly adds bob to cells[1][0]
        assertTrue(cells[1][0].addLifeForm(bob));
        //Checks if the LF just put into cells[1][0] is equivalent to bob
        assertEquals(bob, envObj.getLifeForm(1, 0));
    }

    //assertTrue for successful addition
    //assertFalse for unsuccessful addition
    //assertEquals for
    @Test
    void testAddLifeForm() {
        LifeForm bob = new LifeForm("Bob", 40);
        LifeForm fred = new LifeForm("Fred", 40);
        //Cell[][] cells = new Cell[2][3];

        //For loop fills cells with Cell() objects
        /*
                for (int row = 0; row < cells.length; row++) {
```

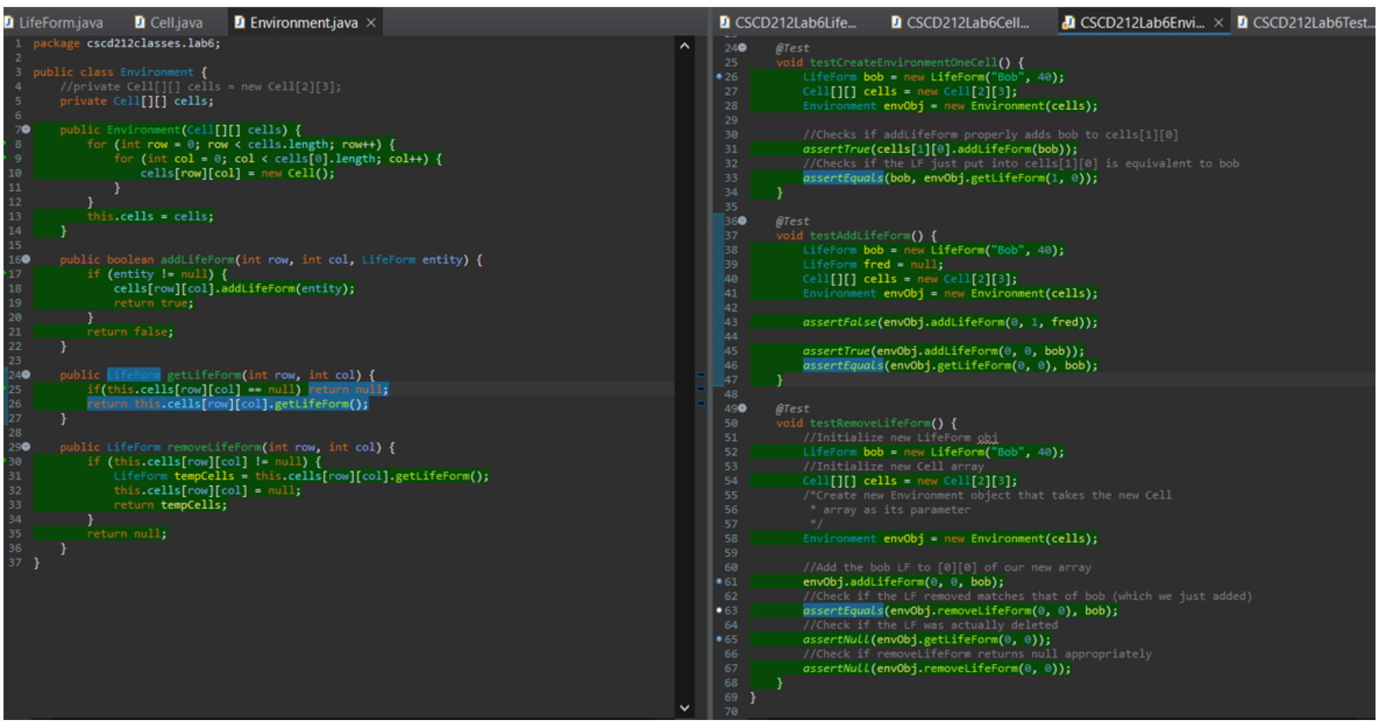
Group 2 Step 6 - testAddLifeForm and testRemoveLifeForm methods
testAddLifeForm:





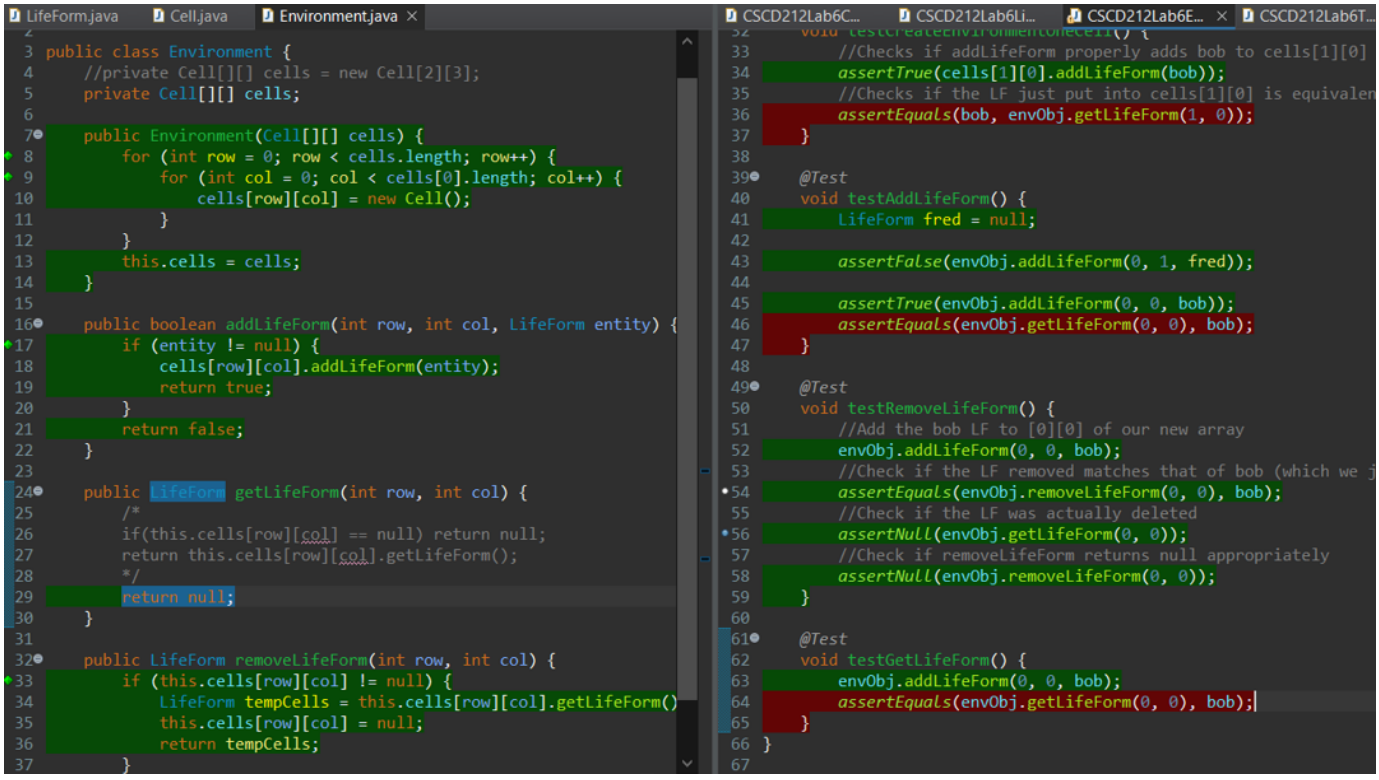
testRemoveLifeForm:



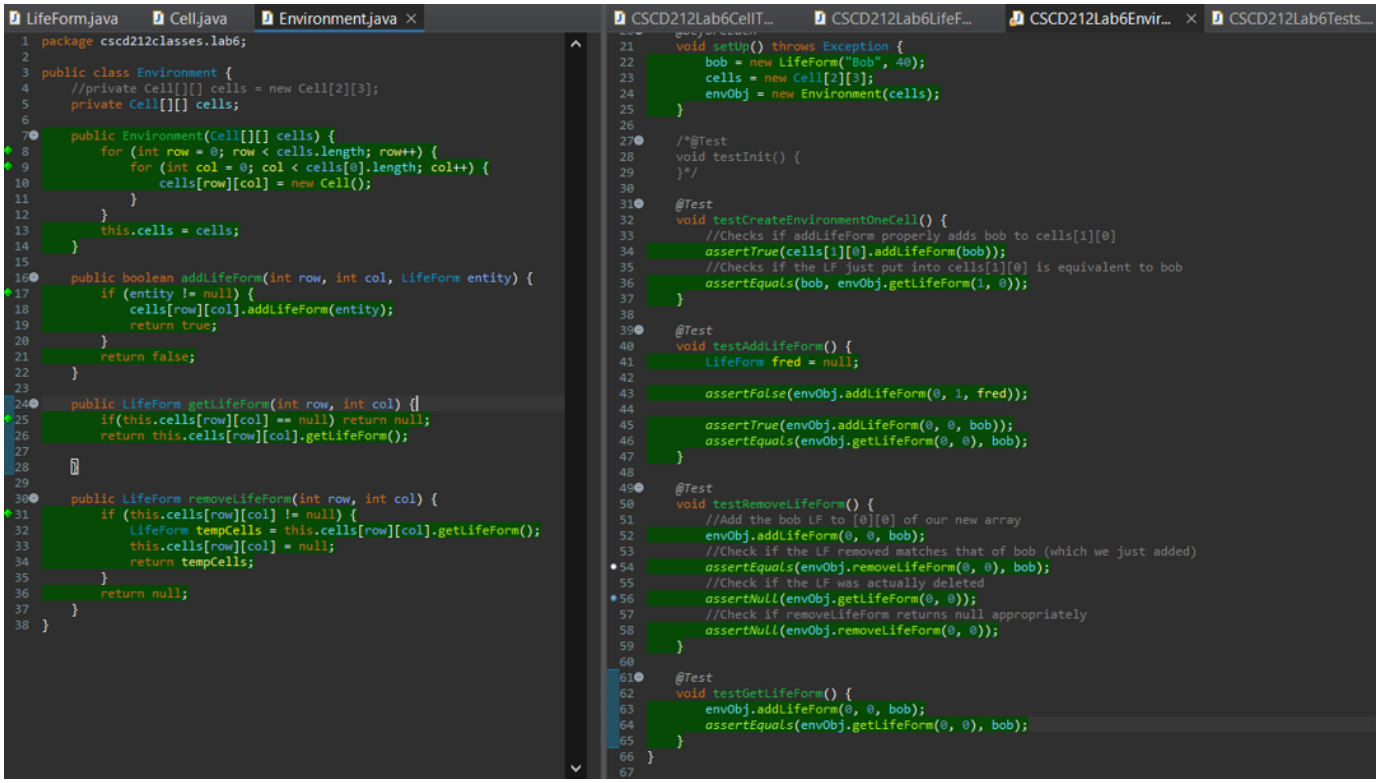


Group 3 Screenshots

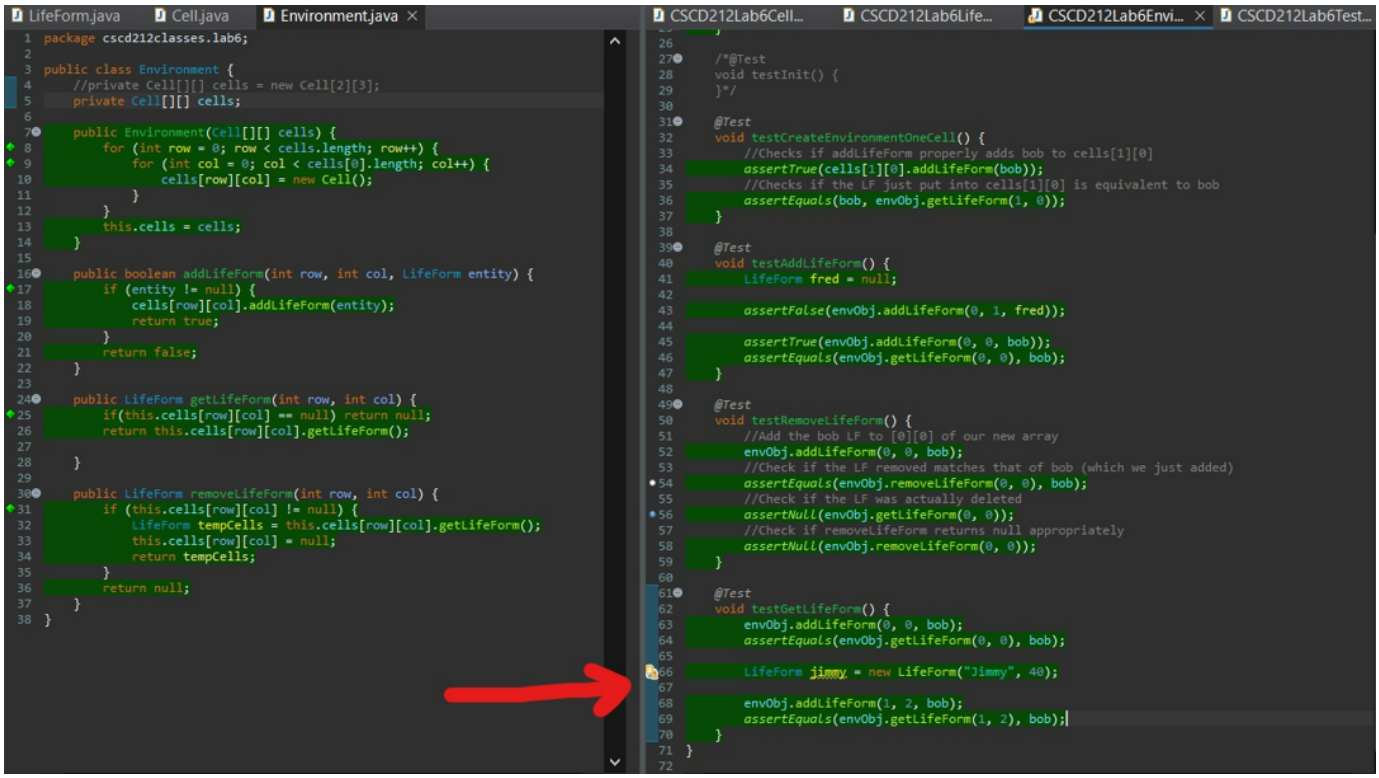
Group 3 Step 1 - Create getLifeForm Test for Environment Class (fail)



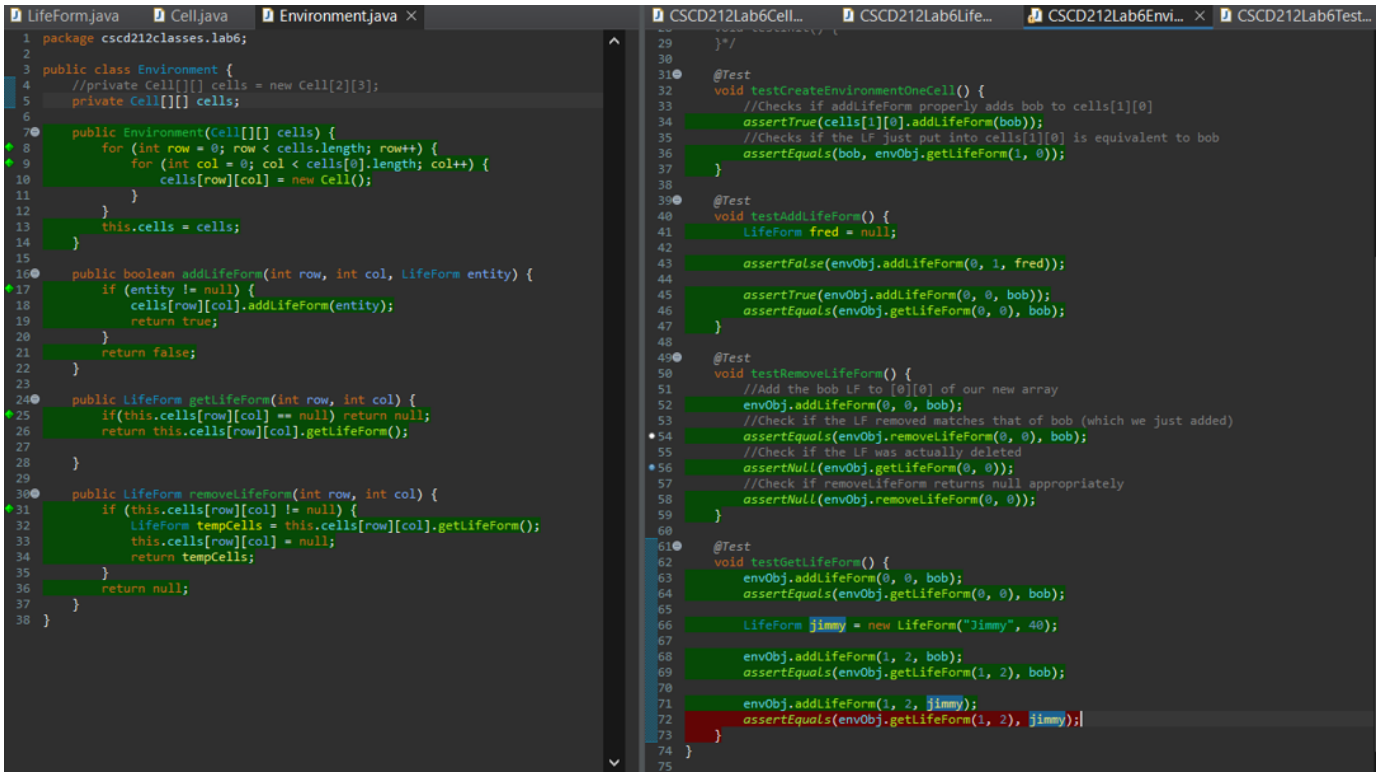
Group 3 Step 2 - getLifeForm Passes



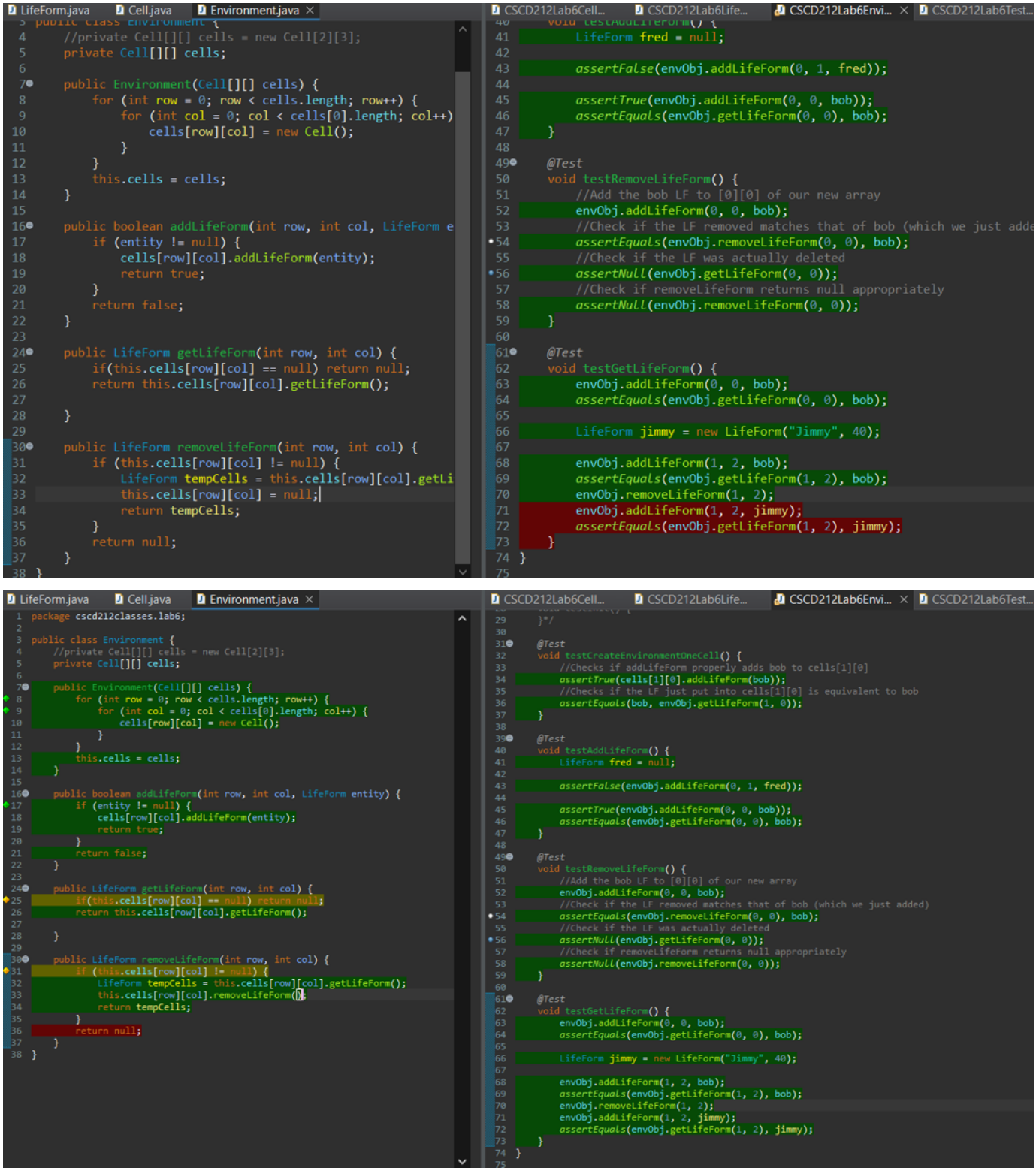
Group 3 Step 7 - Checking Environment to see if it's holding LifeForm



Group 3 Step 8 - Attempting to add second LifeForm to row 1 col 2 (Should fail)



Group 3 Step 9-13 - Removing second LifeForm, checking its removed, then adding it back (Fail -> Pass)



Group 3 Step 15 - Running the Test Suite (All Tests Pass)

LifeForm.java | Cell.java | Environment.java

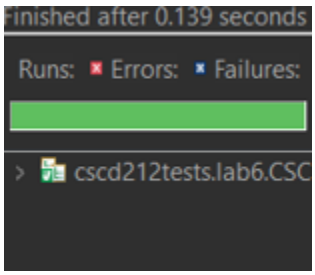
```java
package cscd212classes.lab6;

public class Environment {
    //private Cell[][] cells = new Cell[2][3];
    private Cell[][] cells;

    public Environment(Cell[][] cells) {
        for (int row = 0; row < cells.length; row++)
            for (int col = 0; col < cells[0].length;
                cells[row][col] = new Cell();
            }
        }
        this.cells = cells;
    }

    public boolean addLifeForm(int row, int col, Lif
        if (entity != null) {
            cells[row][col].addLifeForm(entity);
            return true;
        }
        return false;
    }

    public LifeForm getLifeForm(int row, int col) {
        if(this.cells[row][col] == null) return nul
        return this.cells[row][col].getLifeForm();

    }

    public LifeForm removeLifeForm(int row, int col)
        if (this.cells[row][col] != null) {
            LifeForm tempCells = this.cells[row][col
            this.cells[row][col].removeLifeForm();
            return tempCells;
        }
        return null;
    }
}
```

CSCD212Lab6CellT... | CSCD212Lab6LifeF... | CSCD212Lab6Envir...

```java
    }*/

    @Test
    void testCreateEnvironmentOneCell() {
        //Checks if addLifeForm properly adds bob to cells[1][0]
        assertTrue(cells[1][0].addLifeForm(bob));
        //Checks if the LF just put into cells[1][0] is equivalent to
        assertEquals(bob, envObj.getLifeForm(1, 0));
    }

    @Test
    void testAddLifeForm() {
        LifeForm fred = null;

        assertFalse(envObj.addLifeForm(0, 1, fred));

        assertTrue(envObj.addLifeForm(0, 0, bob));
        assertEquals(envObj.getLifeForm(0, 0), bob);
    }

    @Test
    void testRemoveLifeForm() {
        //Add the bob LF to [0][0] of our new array
        envObj.addLifeForm(0, 0, bob);
        //Check if the LF removed matches that of bob (which we just a
        assertEquals(envObj.removeLifeForm(0, 0), bob);
        //Check if the LF was actually deleted
        assertNull(envObj.getLifeForm(0, 0));
        //Check if removeLifeForm returns null appropriately
        assertNull(envObj.removeLifeForm(0, 0));
    }

    @Test
    void testGetLifeForm() {
        envObj.addLifeForm(0, 0, bob);
        assertEquals(envObj.getLifeForm(0, 0), bob);

        LifeForm jimmy = new LifeForm("Jimmy", 40);

        envObj.addLifeForm(1, 2, bob);
        assertEquals(envObj.getLifeForm(1, 2), bob);
        envObj.removeLifeForm(1, 2);
        envObj.addLifeForm(1, 2, jimmy);
        assertEquals(envObj.getLifeForm(1, 2), jimmy);
    }
}
```

CSCD212Lab6Tests.java

```java
package cscd212tests.lab6;

import org.junit.platform.runner.JUnitPlatform;
import org.junit.platform.suite.api.SelectPackages;
import org.junit.runner.RunWith;

import org.junit.platform.suite.api.*;

@RunWith(JUnitPlatform.class)
@SelectPackages("cscd212tests.lab6")
@SelectClasses({CSCD212Lab6EnvironmentTests.class, CSCD212Lab6Cell
public class CSCD212Lab6Tests {}
```

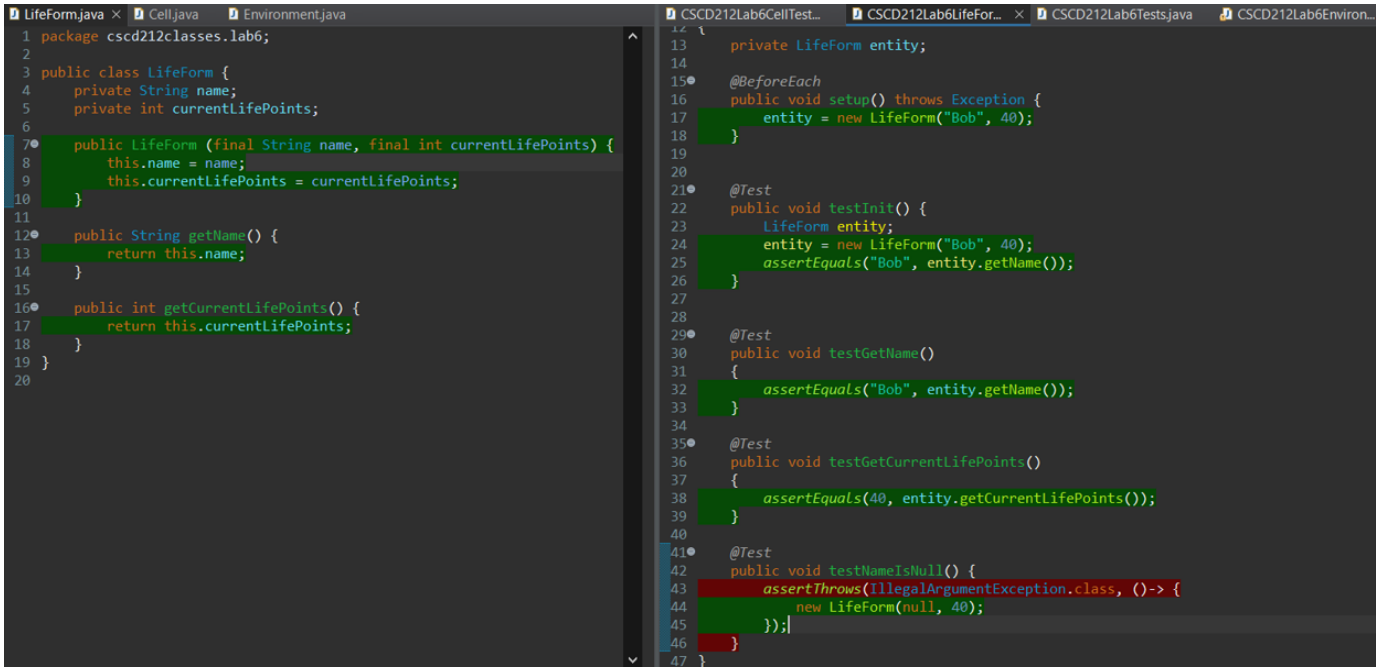


Precondition Screenshots

***NOTE: any red on the right testing side in the passing screenshot is because the assertThrows threw the exception before the inside was executed. My left column of tests say they all passed.**

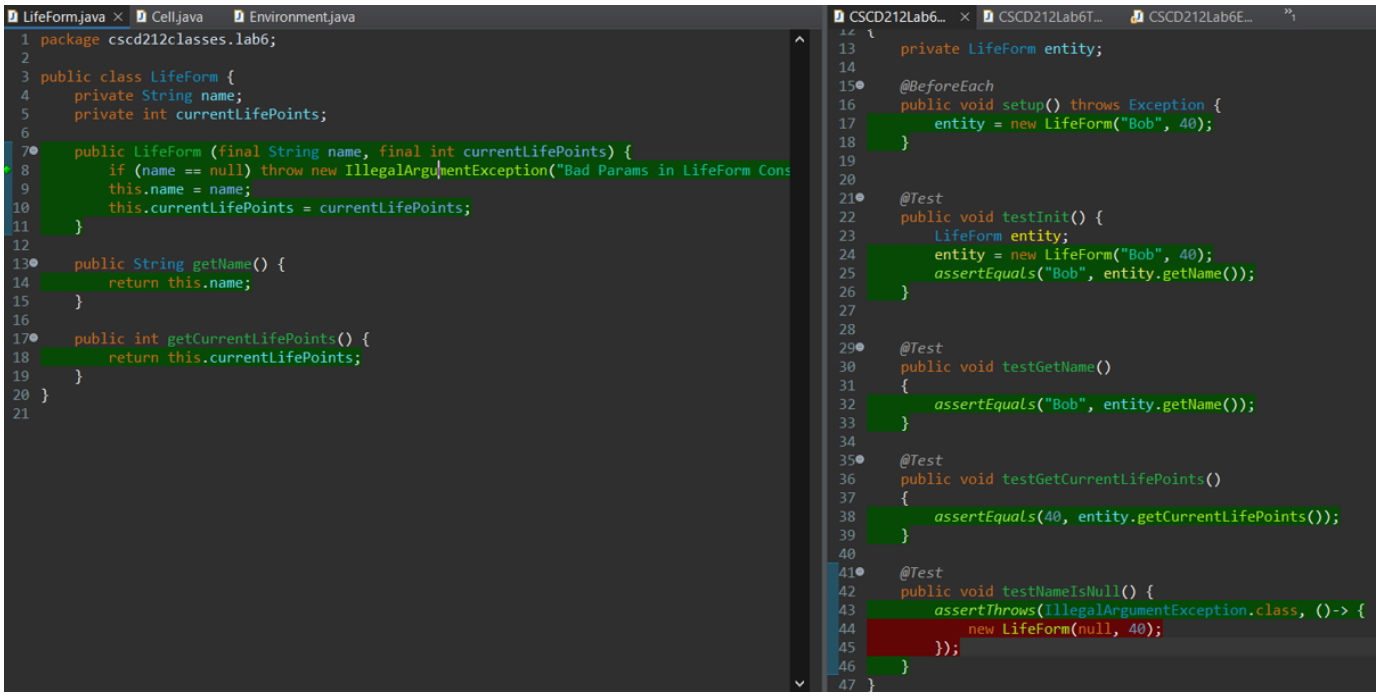
CSCD212LifeFormTests

testNameIsNull:



```
LifeForm.java
package cscd212classes.lab6;

public class LifeForm {
    private String name;
    private int currentLifePoints;

    public LifeForm (final String name, final int currentLifePoints) {
        this.name = name;
        this.currentLifePoints = currentLifePoints;
    }

    public String getName() {
        return this.name;
    }

    public int getCurrentLifePoints() {
        return this.currentLifePoints;
    }
}
```

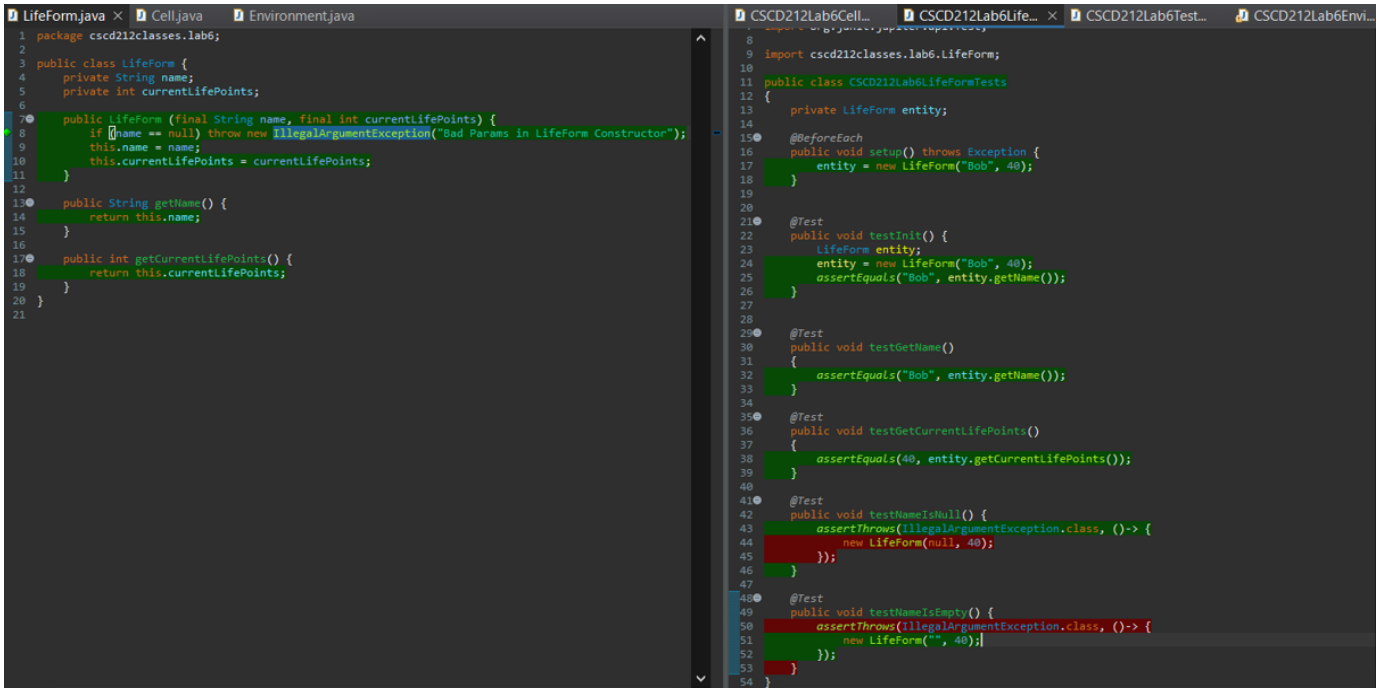
```
CSCD212Lab6LifeFor...
    private LifeForm entity;

    @BeforeEach
    public void setup() throws Exception {
        entity = new LifeForm("Bob", 40);
    }


    @Test
    public void testInit() {
        LifeForm entity;
        entity = new LifeForm("Bob", 40);
        assertEquals("Bob", entity.getName());
    }


    @Test
    public void testGetName()
    {
        assertEquals("Bob", entity.getName());
    }

    @Test
    public void testGetCurrentLifePoints()
    {
        assertEquals(40, entity.getCurrentLifePoints());
    }

    @Test
    public void testNameIsNull() {
        assertThrows(IllegalArgumentException.class, ()-> {
            new LifeForm(null, 40);
        });
    }
}
```

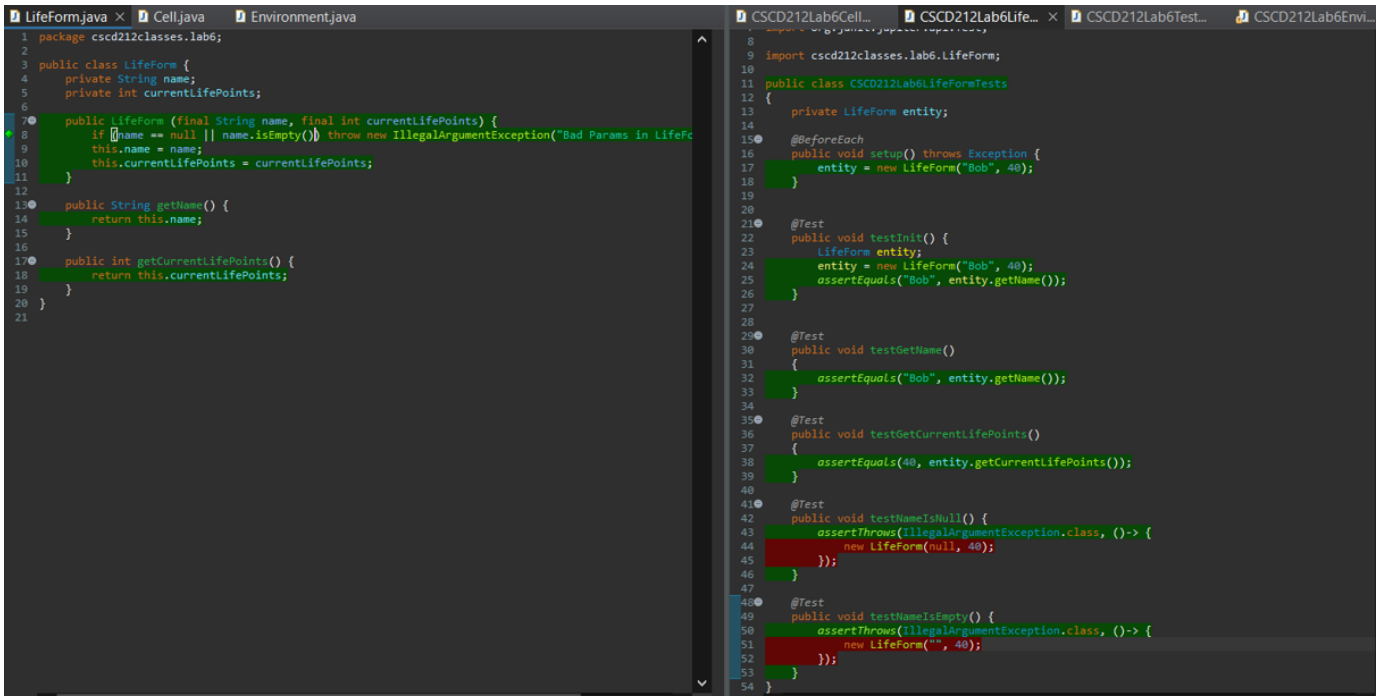


```
LifeForm.java
package cscd212classes.lab6;

public class LifeForm {
    private String name;
    private int currentLifePoints;

    public LifeForm (final String name, final int currentLifePoints) {
        if (name == null) throw new IllegalArgumentException("Bad Params in LifeForm Cons
        this.name = name;
        this.currentLifePoints = currentLifePoints;
    }

    public String getName() {
        return this.name;
    }

    public int getCurrentLifePoints() {
        return this.currentLifePoints;
    }
}
```

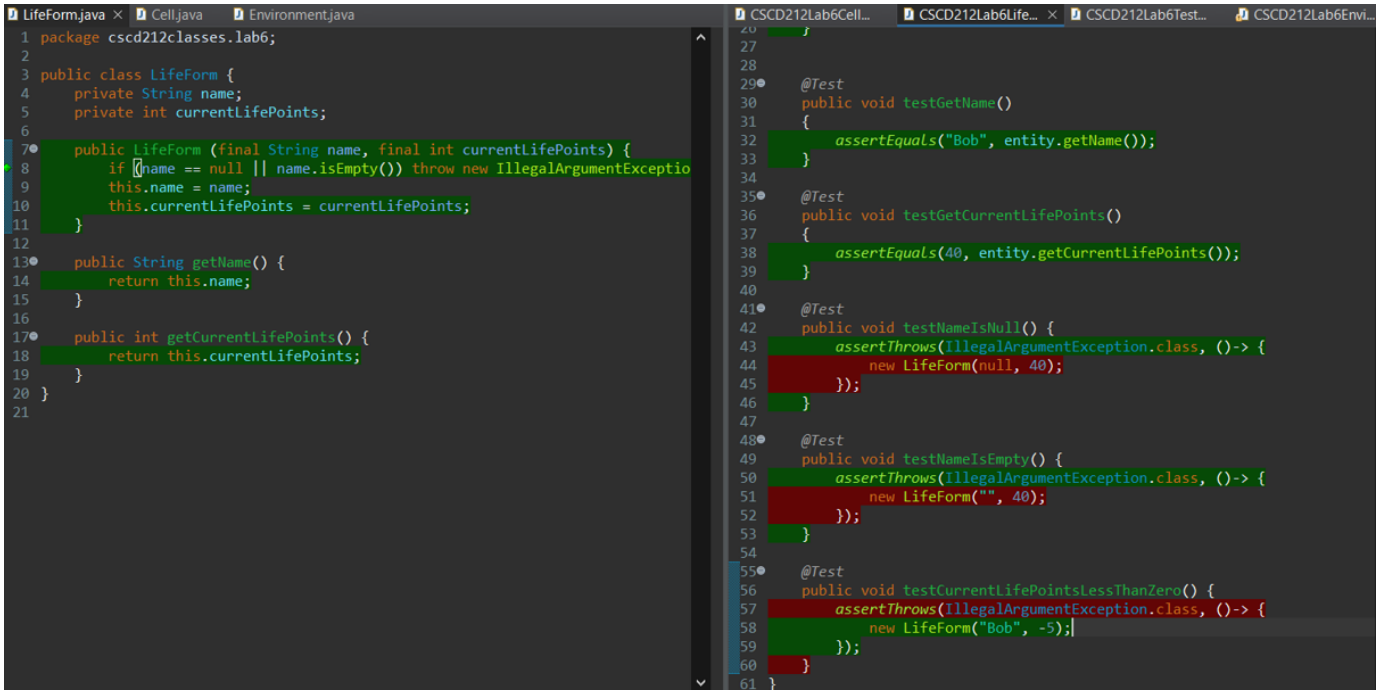
```
CSCD212Lab6...
    private LifeForm entity;

    @BeforeEach
    public void setup() throws Exception {
        entity = new LifeForm("Bob", 40);
    }


    @Test
    public void testInit() {
        LifeForm entity;
        entity = new LifeForm("Bob", 40);
        assertEquals("Bob", entity.getName());
    }


    @Test
    public void testGetName()
    {
        assertEquals("Bob", entity.getName());
    }

    @Test
    public void testGetCurrentLifePoints()
    {
        assertEquals(40, entity.getCurrentLifePoints());
    }

    @Test
    public void testNameIsNull() {
        assertThrows(IllegalArgumentException.class, ()-> {
            new LifeForm(null, 40);
        });
    }
}
```

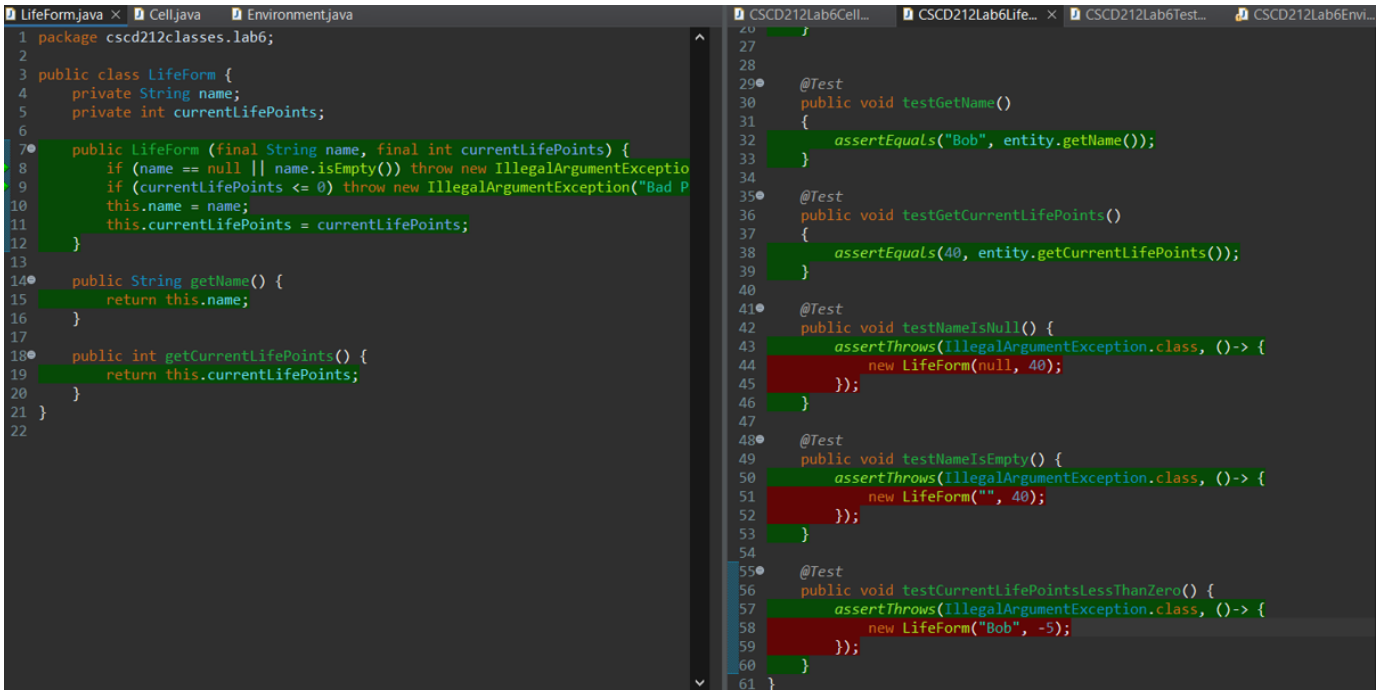
testNameIsEmpty:



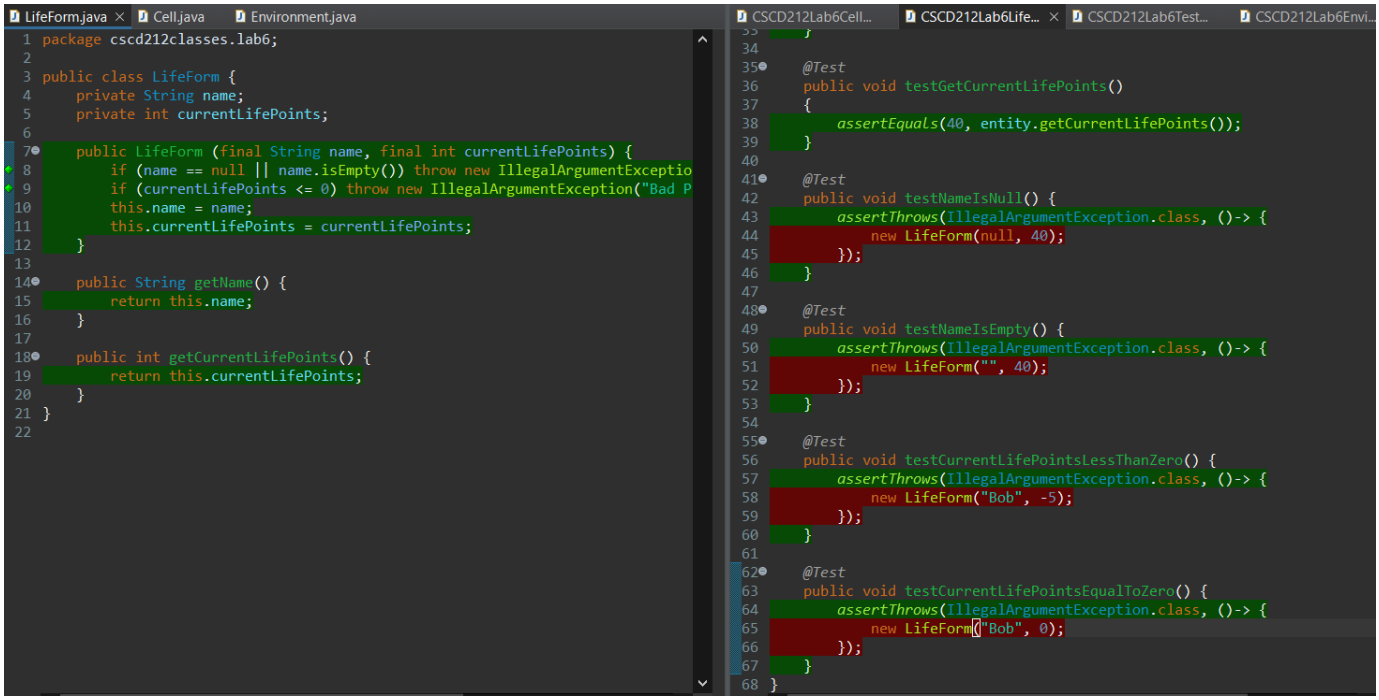


testCurrentLifePoints < 0:





testCurrentLifePoints == 0:



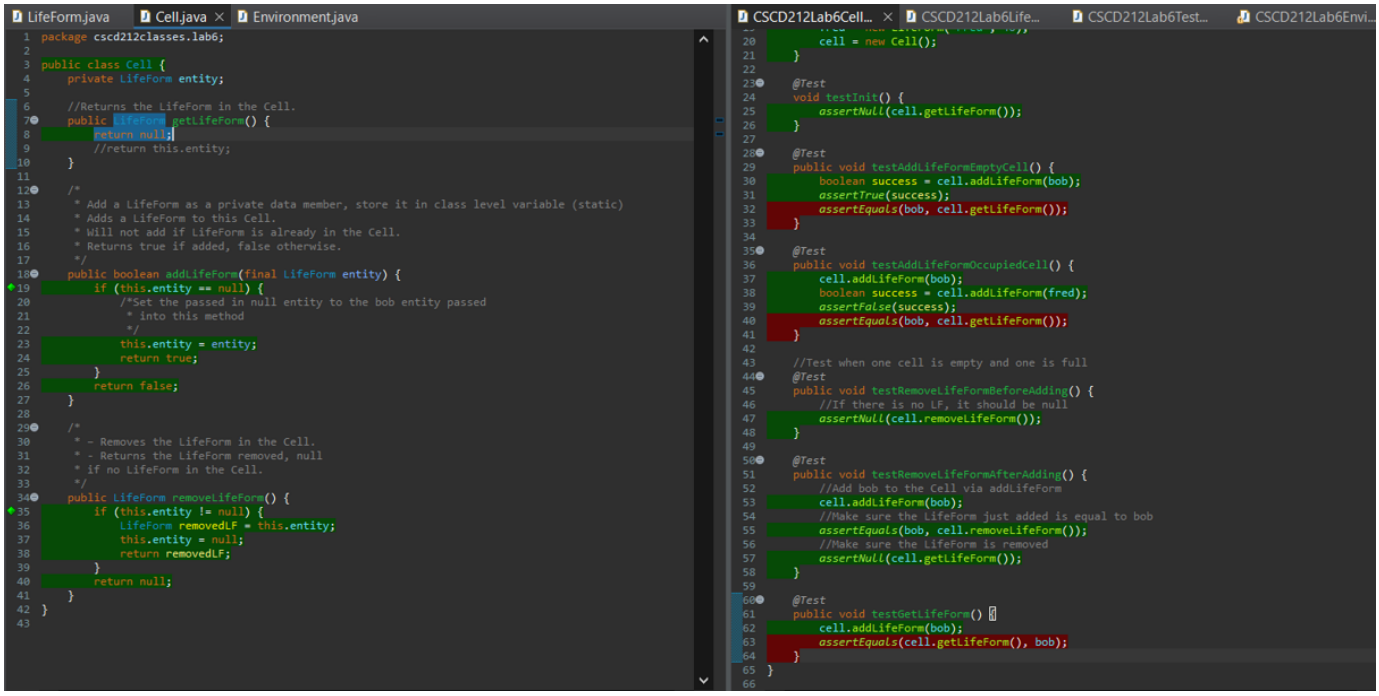
```
LifeForm.java | Cell.java | Environment.java

1  package cscd212classes.lab6;
2
3  public class LifeForm {
4      private String name;
5      private int currentLifePoints;
6
7      public LifeForm (final String name, final int currentLifePoints) {
8          if (name == null || name.isEmpty()) throw new IllegalArgumentExceptio
9          if (currentLifePoints <= 0) throw new IllegalArgumentException("Bad P
10         this.name = name;
11         this.currentLifePoints = currentLifePoints;
12     }
13
14     public String getName() {
15         return this.name;
16     }
17
18     public int getCurrentLifePoints() {
19         return this.currentLifePoints;
20     }
21 }
22
```

```
CSCD212Lab6Cell... | CSCD212Lab6Life... | CSCD212Lab6Test... | CSCD212Lab6Envi...

34
35     @Test
36     public void testGetCurrentLifePoints()
37     {
38         assertEquals(40, entity.getCurrentLifePoints());
39     }
40
41     @Test
42     public void testNameIsNull() {
43         assertThrows(IllegalArgumentException.class, ()-> {
44             new LifeForm(null, 40);
45         });
46     }
47
48     @Test
49     public void testNameIsEmpty() {
50         assertThrows(IllegalArgumentException.class, ()-> {
51             new LifeForm("", 40);
52         });
53     }
54
55     @Test
56     public void testCurrentLifePointsLessThanZero() {
57         assertThrows(IllegalArgumentException.class, ()-> {
58             new LifeForm("Bob", -5);
59         });
60     }
61
62     @Test
63     public void testCurrentLifePointsEqualToZero() {
64         assertThrows(IllegalArgumentException.class, ()-> {
65             new LifeForm("Bob", 0);
66         });
67     }
68 }
```

***NOTE: The screenshots don't show as many for this because I have refactored my old tests, bringing the total amount to 8 total tests.**

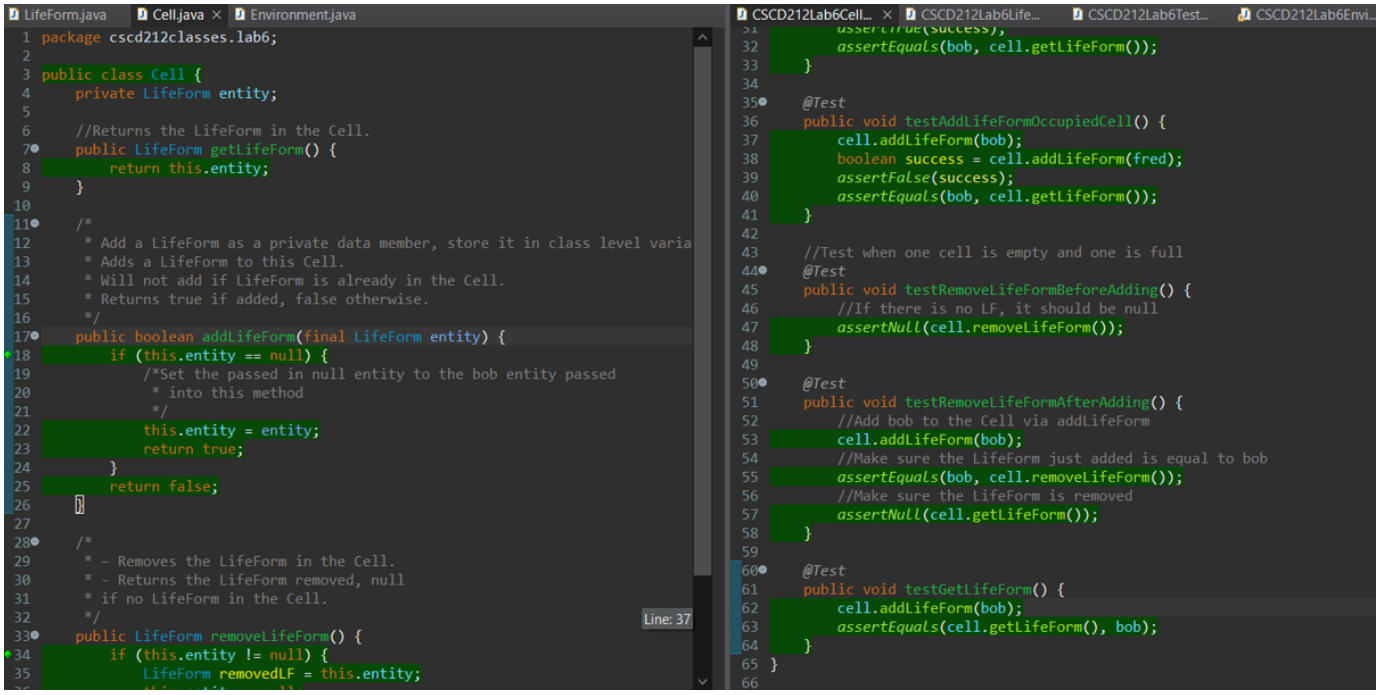
CSCD212CellTests

testGetLifeForm:



```
LifeForm.java | Cell.java | Environment.java

1  package cscd212classes.lab6;
2
3  public class Cell {
4      private LifeForm entity;
5
6      //Returns the LifeForm in the Cell.
7      public LifeForm getLifeForm() {
8          return null;
9          //return this.entity;
10     }
11
12     /*
13      * Add a LifeForm as a private data member, store it in class level variable (static)
14      * Adds a LifeForm to this Cell.
15      * Will not add if LifeForm is already in the Cell.
16      * Returns true if added, false otherwise.
17      */
18     public boolean addLifeForm(final LifeForm entity) {
19         if (this.entity == null) {
20             /*Set the passed in null entity to the bob entity passed
21              * into this method
22              */
23             this.entity = entity;
24             return true;
25         }
26         return false;
27     }
28
29     /*
30      * - Removes the LifeForm in the Cell.
31      * - Returns the LifeForm removed, null
32      * if no LifeForm in the Cell.
33      */
34     public LifeForm removeLifeForm() {
35         if (this.entity != null) {
36             LifeForm removedLF = this.entity;
37             this.entity = null;
38             return removedLF;
39         }
40         return null;
41     }
42 }
43
```

```
CSCD212Lab6Cell... | CSCD212Lab6Life... | CSCD212Lab6Test... | CSCD212Lab6Envi...

20         cell = new Cell();
21     }
22
23     @Test
24     void testInit() {
25         assertNull(cell.getLifeForm());
26     }
27
28     @Test
29     public void testAddLifeFormEmptyCell() {
30         boolean success = cell.addLifeForm(bob);
31         assertTrue(success);
32         assertEquals(bob, cell.getLifeForm());
33     }
34
35     @Test
36     public void testAddLifeFormOccupiedCell() {
37         cell.addLifeForm(bob);
38         boolean success = cell.addLifeForm(fred);
39         assertFalse(success);
40         assertEquals(bob, cell.getLifeForm());
41     }
42
43     //Test when one cell is empty and one is full
44     @Test
45     public void testRemoveLifeFormBeforeAdding() {
46         //If there is no LF, it should be null
47         assertNull(cell.removeLifeForm());
48     }
49
50     @Test
51     public void testRemoveLifeFormAfterAdding() {
52         //Add bob to the Cell via addLifeForm
53         cell.addLifeForm(bob);
54         //Make sure the LifeForm just added is equal to bob
55         assertEquals(bob, cell.removeLifeForm());
56         //Make sure the LifeForm is removed
57         assertNull(cell.getLifeForm());
58     }
59
60     @Test
61     public void testGetLifeForm() {
62         cell.addLifeForm(bob);
63         assertEquals(cell.getLifeForm(), bob);
64     }
65 }
66
```



```
LifeForm.java | Cell.java | Environment.java
```

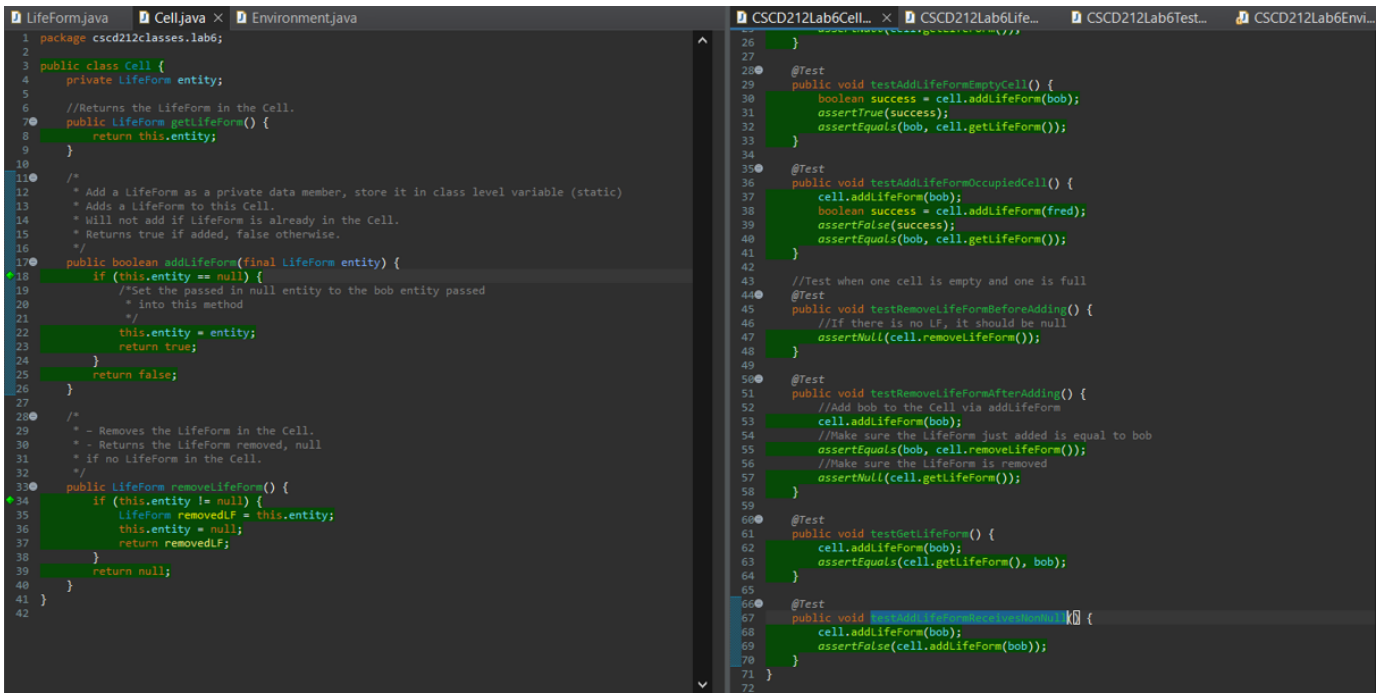
```java
package cscd212classes.lab6;

public class Cell {
    private LifeForm entity;

    //Returns the LifeForm in the Cell.
    public LifeForm getLifeForm() {
        return this.entity;
    }

    /*
     * Add a LifeForm as a private data member, store it in class level varia
     * Adds a LifeForm to this Cell.
     * Will not add if LifeForm is already in the Cell.
     * Returns true if added, false otherwise.
     */
    public boolean addLifeForm(final LifeForm entity) {
        if (this.entity == null) {
            /*Set the passed in null entity to the bob entity passed
             * into this method
             */
            this.entity = entity;
            return true;
        }
        return false;
    }

    /*
     * - Removes the LifeForm in the Cell.
     * - Returns the LifeForm removed, null
     * if no LifeForm in the Cell.
     */
    public LifeForm removeLifeForm() {
        if (this.entity != null) {
            LifeForm removedLF = this.entity;
```

Line: 37

```
CSCD212Lab6Cell... | CSCD212Lab6Life... | CSCD212Lab6Test... | CSCD212Lab6Envi...
```

```java
        assertEquals(bob, cell.getLifeForm());
    }

    @Test
    public void testAddLifeFormOccupiedCell() {
        cell.addLifeForm(bob);
        boolean success = cell.addLifeForm(fred);
        assertFalse(success);
        assertEquals(bob, cell.getLifeForm());
    }

    //Test when one cell is empty and one is full
    @Test
    public void testRemoveLifeFormBeforeAdding() {
        //If there is no LF, it should be null
        assertNull(cell.removeLifeForm());
    }

    @Test
    public void testRemoveLifeFormAfterAdding() {
        //Add bob to the Cell via addLifeForm
        cell.addLifeForm(bob);
        //Make sure the LifeForm just added is equal to bob
        assertEquals(bob, cell.removeLifeForm());
        //Make sure the LifeForm is removed
        assertNull(cell.getLifeForm());
    }

    @Test
    public void testGetLifeForm() {
        cell.addLifeForm(bob);
        assertEquals(cell.getLifeForm(), bob);
    }
}
```

testAddLifeFormReceivesNonNull: *passed right away



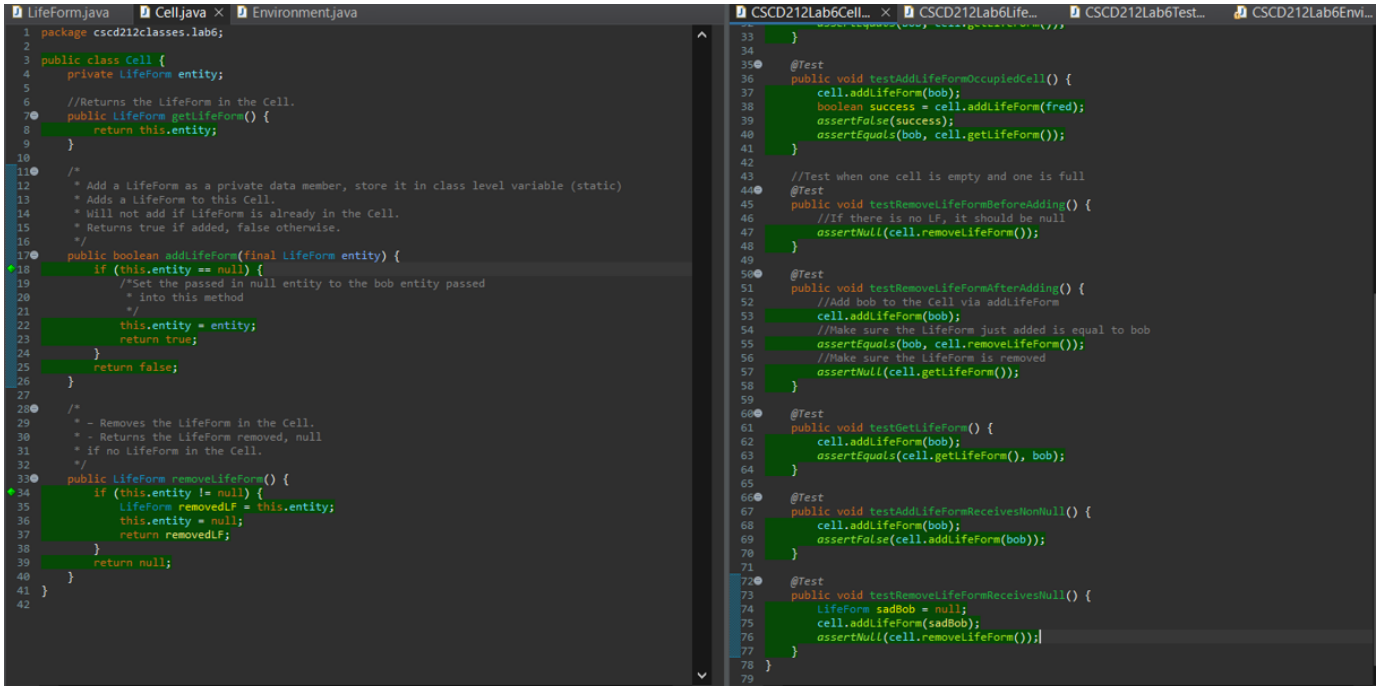
```
LifeForm.java | Cell.java | Environment.java
```

```java
package cscd212classes.lab6;

public class Cell {
    private LifeForm entity;

    //Returns the LifeForm in the Cell.
    public LifeForm getLifeForm() {
        return this.entity;
    }

    /*
     * Add a LifeForm as a private data member, store it in class level variable (static)
     * Adds a LifeForm to this Cell.
     * Will not add if LifeForm is already in the Cell.
     * Returns true if added, false otherwise.
     */
    public boolean addLifeForm(final LifeForm entity) {
        if (this.entity == null) {
            /*Set the passed in null entity to the bob entity passed
             * into this method
             */
            this.entity = entity;
            return true;
        }
        return false;
    }

    /*
     * - Removes the LifeForm in the Cell.
     * - Returns the LifeForm removed, null
     * if no LifeForm in the Cell.
     */
    public LifeForm removeLifeForm() {
        if (this.entity != null) {
            LifeForm removedLF = this.entity;
            this.entity = null;
            return removedLF;
        }
        return null;
    }
}
```

```
CSCD212Lab6Cell... | CSCD212Lab6Life... | CSCD212Lab6Test... | CSCD212Lab6Envi...
```

```java
    }

    @Test
    public void testAddLifeFormEmptyCell() {
        boolean success = cell.addLifeForm(bob);
        assertTrue(success);
        assertEquals(bob, cell.getLifeForm());
    }

    @Test
    public void testAddLifeFormOccupiedCell() {
        cell.addLifeForm(bob);
        boolean success = cell.addLifeForm(fred);
        assertFalse(success);
        assertEquals(bob, cell.getLifeForm());
    }

    //Test when one cell is empty and one is full
    @Test
    public void testRemoveLifeFormBeforeAdding() {
        //If there is no LF, it should be null
        assertNull(cell.removeLifeForm());
    }

    @Test
    public void testRemoveLifeFormAfterAdding() {
        //Add bob to the Cell via addLifeForm
        cell.addLifeForm(bob);
        //Make sure the LifeForm just added is equal to bob
        assertEquals(bob, cell.removeLifeForm());
        //Make sure the LifeForm is removed
        assertNull(cell.getLifeForm());
    }

    @Test
    public void testGetLifeForm() {
        cell.addLifeForm(bob);
        assertEquals(cell.getLifeForm(), bob);
    }

    @Test
    public void testAddLifeFormReceivesNonNull() {
        cell.addLifeForm(bob);
        assertFalse(cell.addLifeForm(bob));
    }
}
```

testRemoveLifeFormReceivesNull: *passed right away



```
LifeForm.java | Cell.java × | Environment.java
```

```java
package cscd212classes.lab6;

public class Cell {
    private LifeForm entity;

    //Returns the LifeForm in the Cell.
    public LifeForm getLifeForm() {
        return this.entity;
    }

    /*
     * Add a LifeForm as a private data member, store it in class level variable (static)
     * Adds a LifeForm to this Cell.
     * Will not add if LifeForm is already in the Cell.
     * Returns true if added, false otherwise.
     */
    public boolean addLifeForm(final LifeForm entity) {
        if (this.entity == null) {
            /*Set the passed in null entity to the bob entity passed
             * into this method
             */
            this.entity = entity;
            return true;
        }
        return false;
    }

    /*
     * - Removes the LifeForm in the Cell.
     * - Returns the LifeForm removed, null
     * if no LifeForm in the Cell.
     */
    public LifeForm removeLifeForm() {
        if (this.entity != null) {
            LifeForm removedLF = this.entity;
            this.entity = null;
            return removedLF;
        }
        return null;
    }
}
```

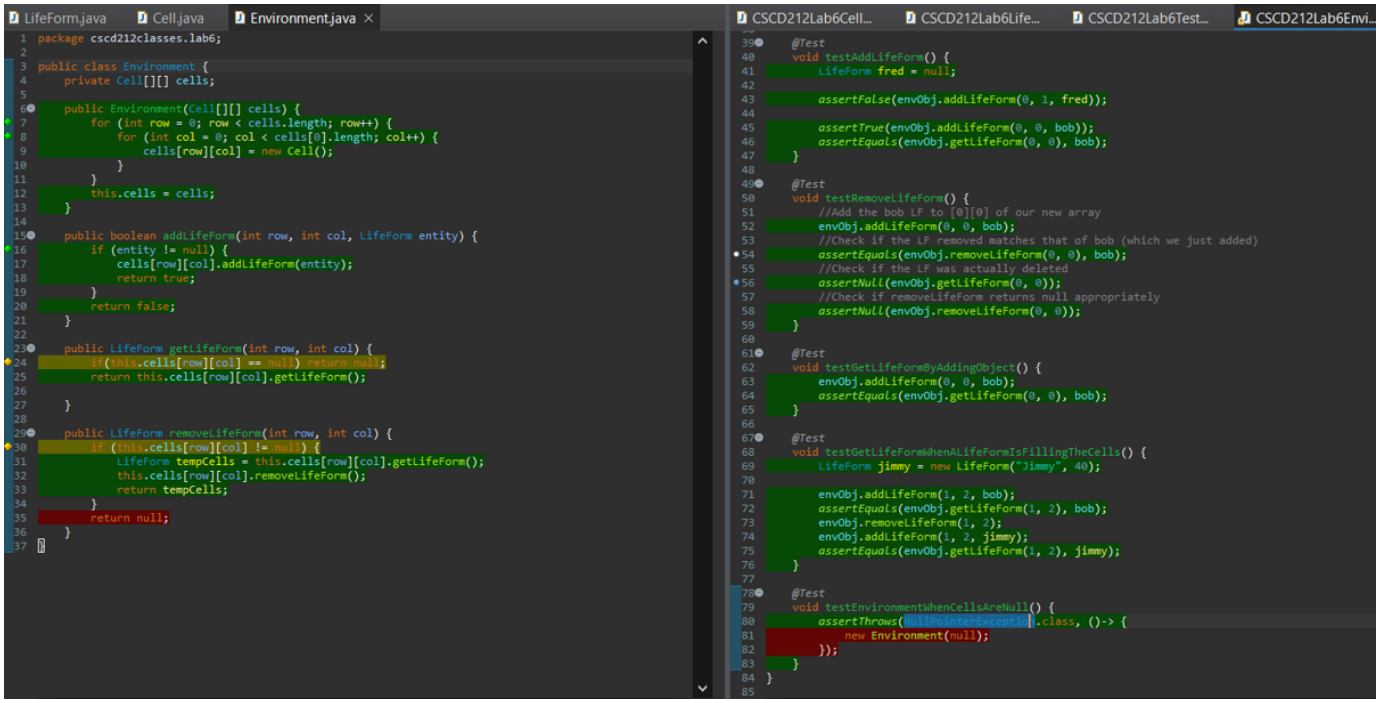
```
CSCD212Lab6Cell... × | CSCD212Lab6Life... | CSCD212Lab6Test... | CSCD212Lab6Envi...
```

```java
    }

    @Test
    public void testAddLifeFormOccupiedCell() {
        cell.addLifeForm(bob);
        boolean success = cell.addLifeForm(fred);
        assertFalse(success);
        assertEquals(bob, cell.getLifeForm());
    }

    //Test when one cell is empty and one is full
    @Test
    public void testRemoveLifeFormBeforeAdding() {
        //If there is no LF, it should be null
        assertNull(cell.removeLifeForm());
    }

    @Test
    public void testRemoveLifeFormAfterAdding() {
        //Add bob to the Cell via addLifeForm
        cell.addLifeForm(bob);
        //Make sure the LifeForm just added is equal to bob
        assertEquals(bob, cell.removeLifeForm());
        //Make sure the LifeForm is removed
        assertNull(cell.getLifeForm());
    }

    @Test
    public void testGetLifeForm() {
        cell.addLifeForm(bob);
        assertEquals(cell.getLifeForm(), bob);
    }

    @Test
    public void testAddLifeFormReceivesNonNull() {
        cell.addLifeForm(bob);
        assertFalse(cell.addLifeForm(bob));
    }

    @Test
    public void testRemoveLifeFormReceivesNull() {
        LifeForm sadBob = null;
        cell.addLifeForm(sadBob);
        assertNull(cell.removeLifeForm());
    }
}
```

***NOTE: any red on the right testing side in the passing screenshot is because the assertThrows threw the exception before the inside was executed. My left column of tests say they all passed. Also, refactoring old tests brings the total to 19 tests.**

CSCD212EnvironmentTests

testEnvironmentWhenCellsAreNull: *passed right away



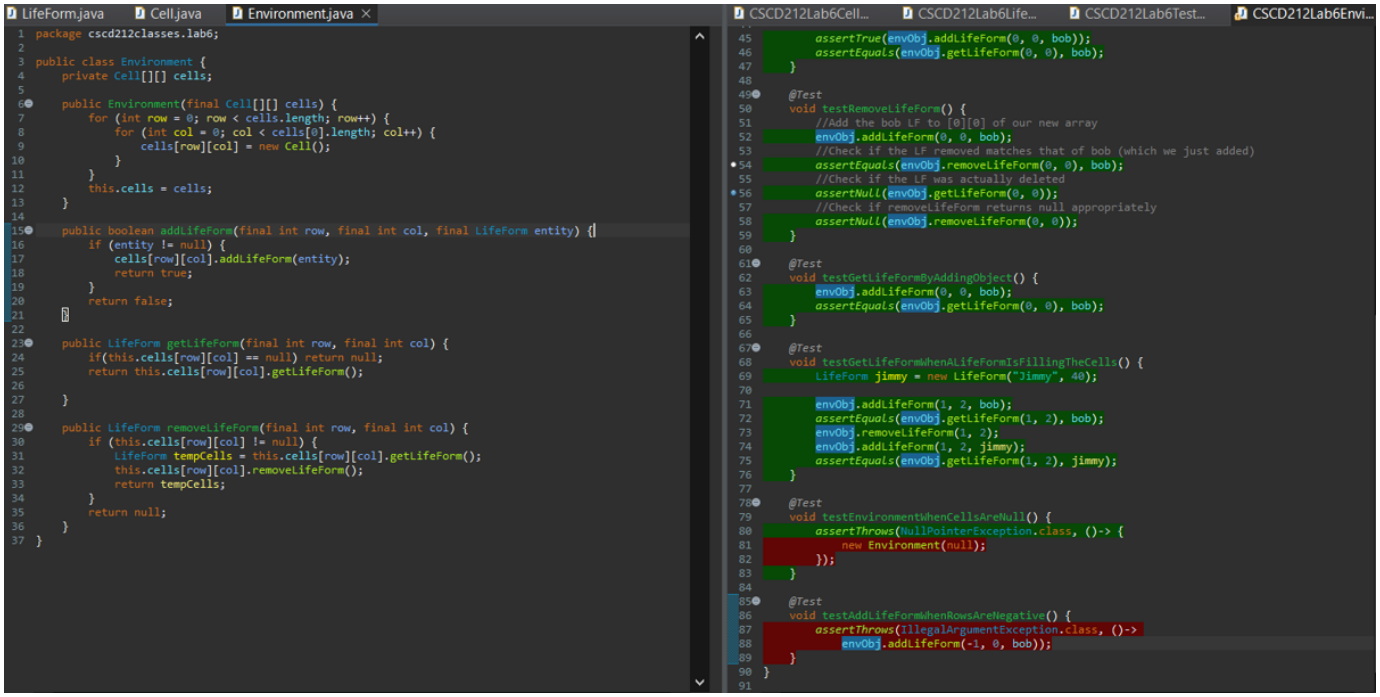
```
LifeForm.java | Cell.java | Environment.java ×
```

```java
package cscd212classes.lab6;

public class Environment {
    private Cell[][] cells;

    public Environment(Cell[][] cells) {
        for (int row = 0; row < cells.length; row++) {
            for (int col = 0; col < cells[0].length; col++) {
                cells[row][col] = new Cell();
            }
        }
        this.cells = cells;
    }

    public boolean addLifeForm(int row, int col, LifeForm entity) {
        if (entity != null) {
            cells[row][col].addLifeForm(entity);
            return true;
        }
        return false;
    }

    public LifeForm getLifeForm(int row, int col) {
        if(this.cells[row][col] == null) return null;
        return this.cells[row][col].getLifeForm();

    }

    public LifeForm removeLifeForm(int row, int col) {
        if (this.cells[row][col] != null) {
            LifeForm tempCells = this.cells[row][col].getLifeForm();
            this.cells[row][col].removeLifeForm();
            return tempCells;
        }
        return null;
    }
}
```

```
CSCD212Lab6Cell... | CSCD212Lab6Life... | CSCD212Lab6Test... | CSCD212Lab6Envi...
```

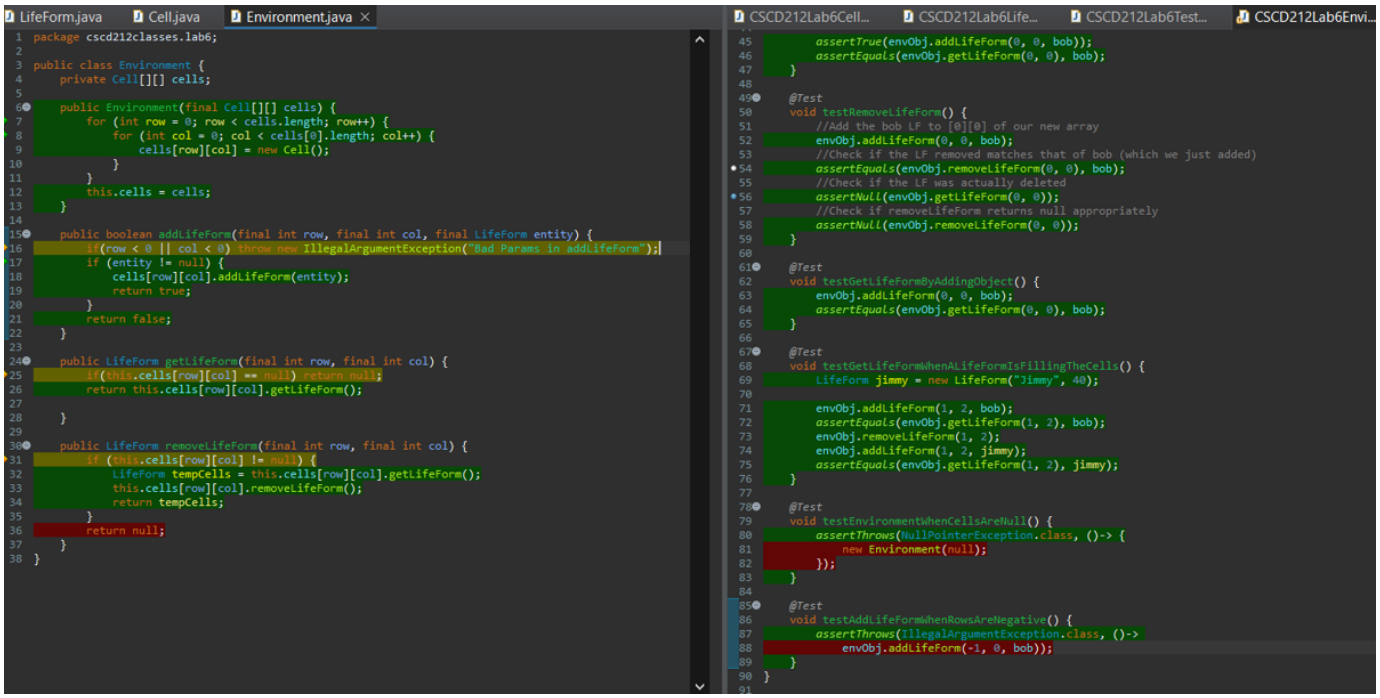
```java
    @Test
    void testAddLifeForm() {
        LifeForm fred = null;

        assertFalse(envObj.addLifeForm(0, 1, fred));

        assertTrue(envObj.addLifeForm(0, 0, bob));
        assertEquals(envObj.getLifeForm(0, 0), bob);
    }

    @Test
    void testRemoveLifeForm() {
        //Add the bob LF to [0][0] of our new array
        envObj.addLifeForm(0, 0, bob);
        //Check if the LF removed matches that of bob (which we just added)
        assertEquals(envObj.removeLifeForm(0, 0), bob);
        //Check if the LF was actually deleted
        assertNull(envObj.getLifeForm(0, 0));
        //Check if removeLifeForm returns null appropriately
        assertNull(envObj.removeLifeForm(0, 0));
    }

    @Test
    void testGetLifeFormByAddingObject() {
        envObj.addLifeForm(0, 0, bob);
        assertEquals(envObj.getLifeForm(0, 0), bob);
    }

    @Test
    void testGetLifeFormWhenALifeFormIsFillingTheCells() {
        LifeForm jimmy = new LifeForm("Jimmy", 40);

        envObj.addLifeForm(1, 2, bob);
        assertEquals(envObj.getLifeForm(1, 2), bob);
        envObj.removeLifeForm(1, 2);
        envObj.addLifeForm(1, 2, jimmy);
        assertEquals(envObj.getLifeForm(1, 2), jimmy);
    }

    @Test
    void testEnvironmentWhenCellsAreNull() {
        assertThrows(NullPointerException.class, ()-> {
            new Environment(null);
        });
    }
}
```

testAddLifeFormWhenRowsAreNegative:



```
LifeForm.java | Cell.java | Environment.java

package cscd212classes.lab6;

public class Environment {
    private Cell[][] cells;

    public Environment(final Cell[][] cells) {
        for (int row = 0; row < cells.length; row++) {
            for (int col = 0; col < cells[0].length; col++) {
                cells[row][col] = new Cell();
            }
        }
        this.cells = cells;
    }

    public boolean addLifeForm(final int row, final int col, final LifeForm entity) {
        if (entity != null) {
            cells[row][col].addLifeForm(entity);
            return true;
        }
        return false;
    }

    public LifeForm getLifeForm(final int row, final int col) {
        if(this.cells[row][col] == null) return null;
        return this.cells[row][col].getLifeForm();

    }

    public LifeForm removeLifeForm(final int row, final int col) {
        if (this.cells[row][col] != null) {
            LifeForm tempCells = this.cells[row][col].getLifeForm();
            this.cells[row][col].removeLifeForm();
            return tempCells;
        }
        return null;
    }
}
```

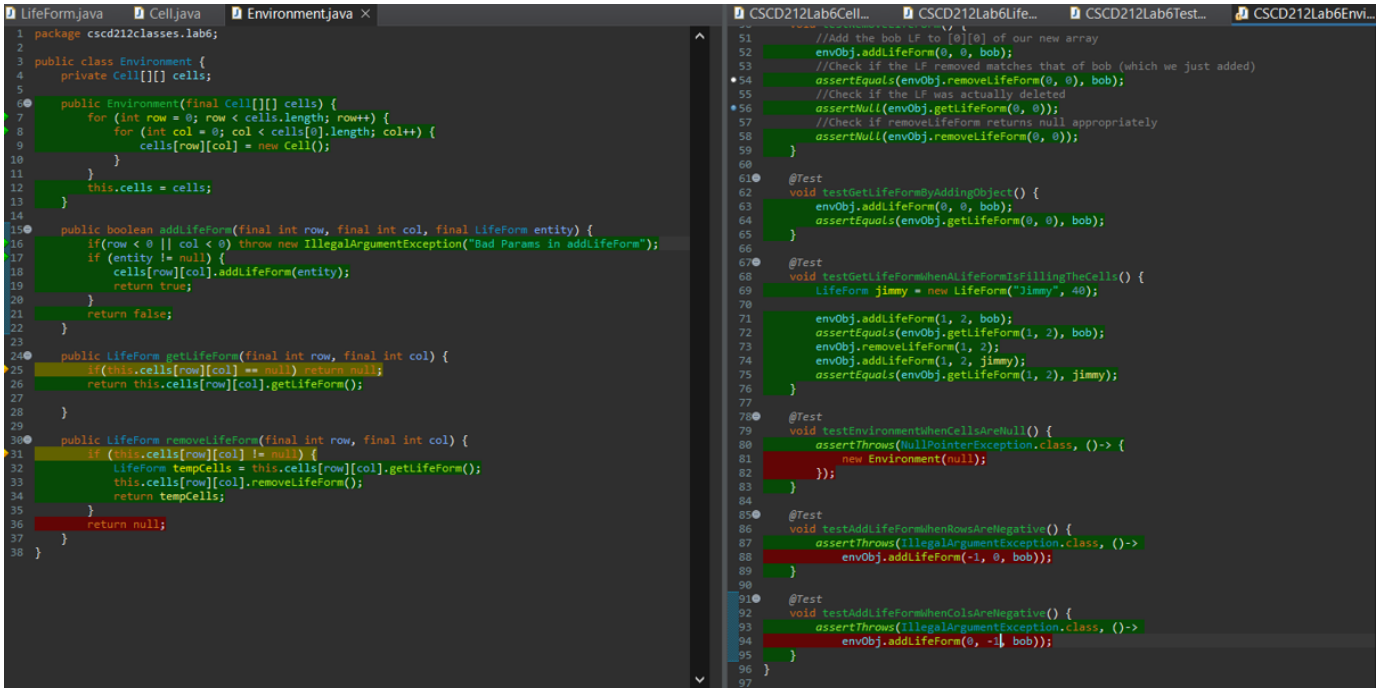
```
CSCD212Lab6Cell... | CSCD212Lab6Life... | CSCD212Lab6Test... | CSCD212Lab6Envi...

        assertTrue(envObj.addLifeForm(0, 0, bob));
        assertEquals(envObj.getLifeForm(0, 0), bob);
    }

    @Test
    void testRemoveLifeForm() {
        //Add the bob LF to [0][0] of our new array
        envObj.addLifeForm(0, 0, bob);
        //Check if the LF removed matches that of bob (which we just added)
        assertEquals(envObj.removeLifeForm(0, 0), bob);
        //Check if the LF was actually deleted
        assertNull(envObj.getLifeForm(0, 0));
        //Check if removeLifeForm returns null appropriately
        assertNull(envObj.removeLifeForm(0, 0));
    }

    @Test
    void testGetLifeFormByAddingObject() {
        envObj.addLifeForm(0, 0, bob);
        assertEquals(envObj.getLifeForm(0, 0), bob);
    }

    @Test
    void testGetLifeFormWhenALifeFormIsFillingTheCells() {
        LifeForm jimmy = new LifeForm("Jimmy", 40);

        envObj.addLifeForm(1, 2, bob);
        assertEquals(envObj.getLifeForm(1, 2), bob);
        envObj.removeLifeForm(1, 2);
        envObj.addLifeForm(1, 2, jimmy);
        assertEquals(envObj.getLifeForm(1, 2), jimmy);
    }

    @Test
    void testEnvironmentWhenCellsAreNull() {
        assertThrows(NullPointerException.class, ()-> {
            new Environment(null);
        });
    }

    @Test
    void testAddLifeFormWhenRowsAreNegative() {
        assertThrows(IllegalArgumentException.class, ()->
            envObj.addLifeForm(-1, 0, bob));
    }
}
```



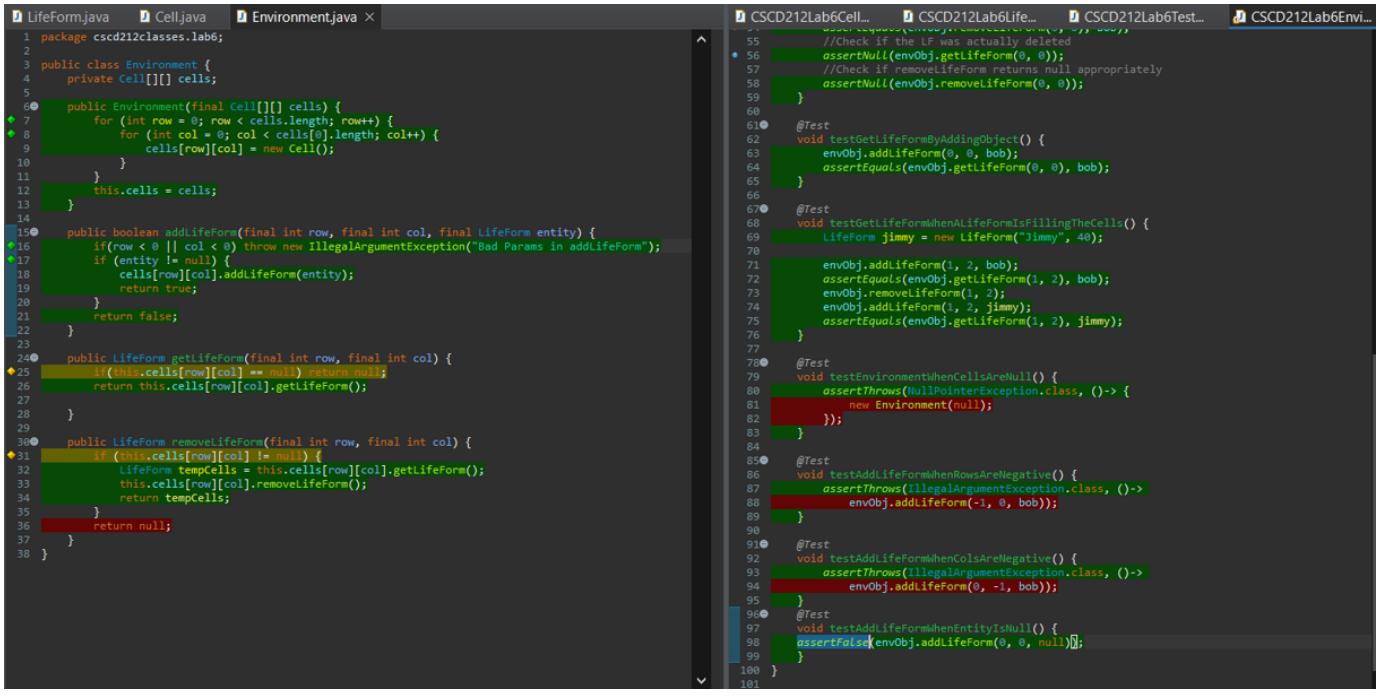
```
LifeForm.java | Cell.java | Environment.java

package cscd212classes.lab6;

public class Environment {
    private Cell[][] cells;

    public Environment(final Cell[][] cells) {
        for (int row = 0; row < cells.length; row++) {
            for (int col = 0; col < cells[0].length; col++) {
                cells[row][col] = new Cell();
            }
        }
        this.cells = cells;
    }

    public boolean addLifeForm(final int row, final int col, final LifeForm entity) {
        if(row < 0 || col < 0) throw new IllegalArgumentException("Bad Params in addLifeForm");
        if (entity != null) {
            cells[row][col].addLifeForm(entity);
            return true;
        }
        return false;
    }

    public LifeForm getLifeForm(final int row, final int col) {
        if(this.cells[row][col] == null) return null;
        return this.cells[row][col].getLifeForm();

    }

    public LifeForm removeLifeForm(final int row, final int col) {
        if (this.cells[row][col] != null) {
            LifeForm tempCells = this.cells[row][col].getLifeForm();
            this.cells[row][col].removeLifeForm();
            return tempCells;
        }
        return null;
    }
}
```

```
CSCD212Lab6Cell... | CSCD212Lab6Life... | CSCD212Lab6Test... | CSCD212Lab6Envi...

        assertTrue(envObj.addLifeForm(0, 0, bob));
        assertEquals(envObj.getLifeForm(0, 0), bob);
    }

    @Test
    void testRemoveLifeForm() {
        //Add the bob LF to [0][0] of our new array
        envObj.addLifeForm(0, 0, bob);
        //Check if the LF removed matches that of bob (which we just added)
        assertEquals(envObj.removeLifeForm(0, 0), bob);
        //Check if the LF was actually deleted
        assertNull(envObj.getLifeForm(0, 0));
        //Check if removeLifeForm returns null appropriately
        assertNull(envObj.removeLifeForm(0, 0));
    }

    @Test
    void testGetLifeFormByAddingObject() {
        envObj.addLifeForm(0, 0, bob);
        assertEquals(envObj.getLifeForm(0, 0), bob);
    }

    @Test
    void testGetLifeFormWhenALifeFormIsFillingTheCells() {
        LifeForm jimmy = new LifeForm("Jimmy", 40);

        envObj.addLifeForm(1, 2, bob);
        assertEquals(envObj.getLifeForm(1, 2), bob);
        envObj.removeLifeForm(1, 2);
        envObj.addLifeForm(1, 2, jimmy);
        assertEquals(envObj.getLifeForm(1, 2), jimmy);
    }

    @Test
    void testEnvironmentWhenCellsAreNull() {
        assertThrows(NullPointerException.class, ()-> {
            new Environment(null);
        });
    }

    @Test
    void testAddLifeFormWhenRowsAreNegative() {
        assertThrows(IllegalArgumentException.class, ()->
            envObj.addLifeForm(-1, 0, bob));
    }
}
```

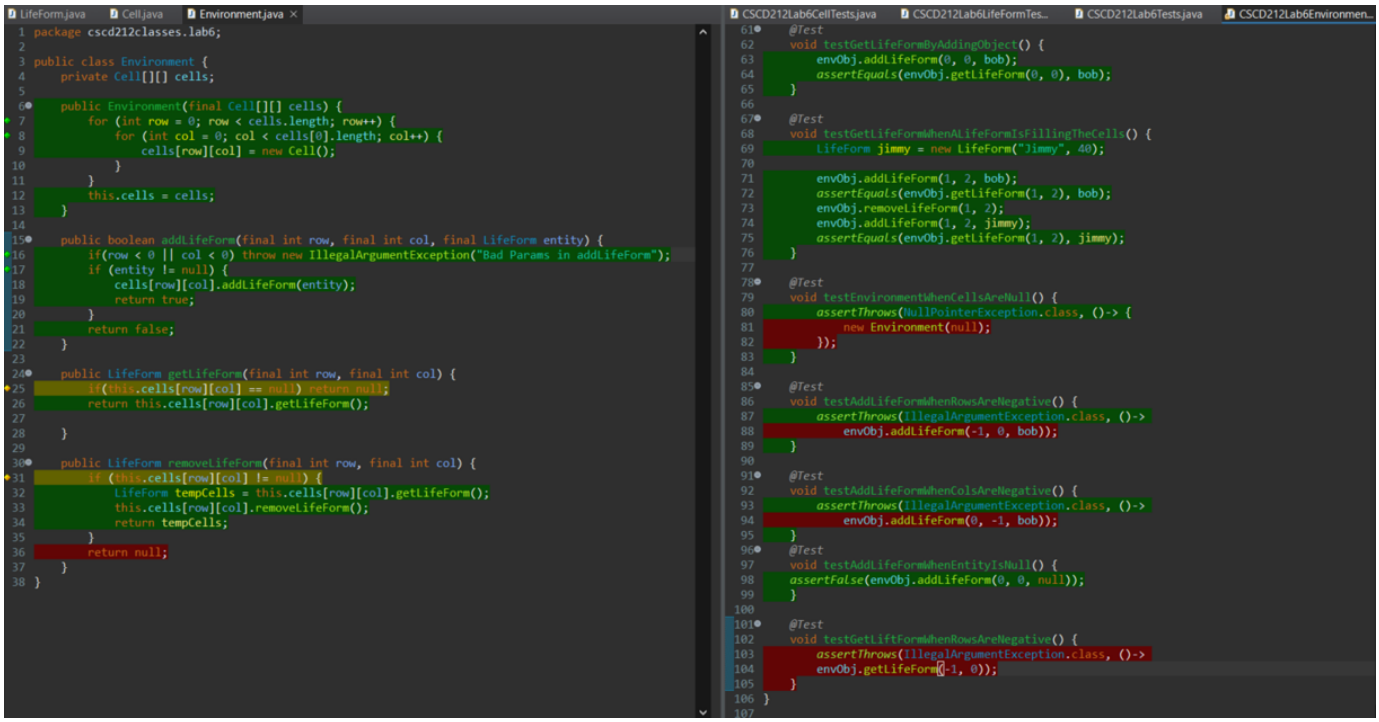
testAddLifeFormWhenColsAreNegative: *passed because I fixed it out of habit on accident just above

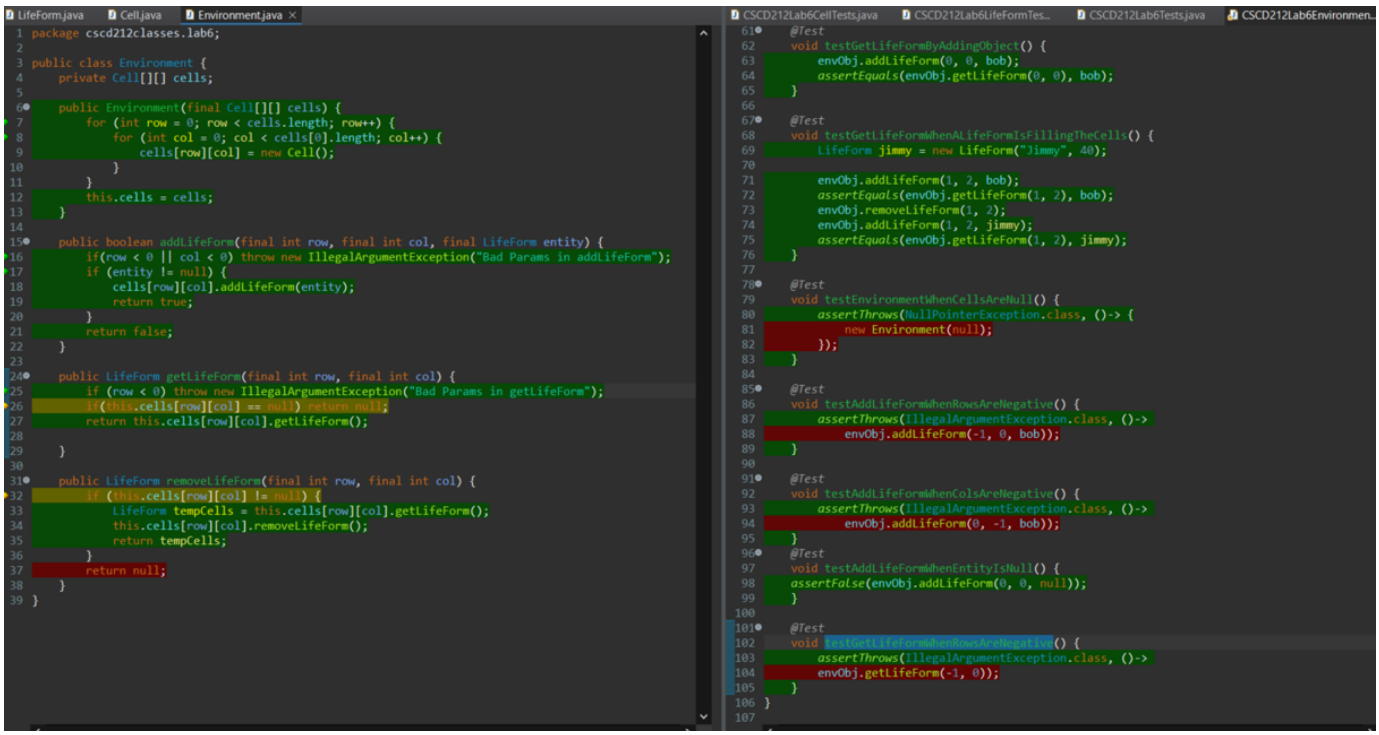


testAddLifeFormWhenEntityIsNull: *passed right away

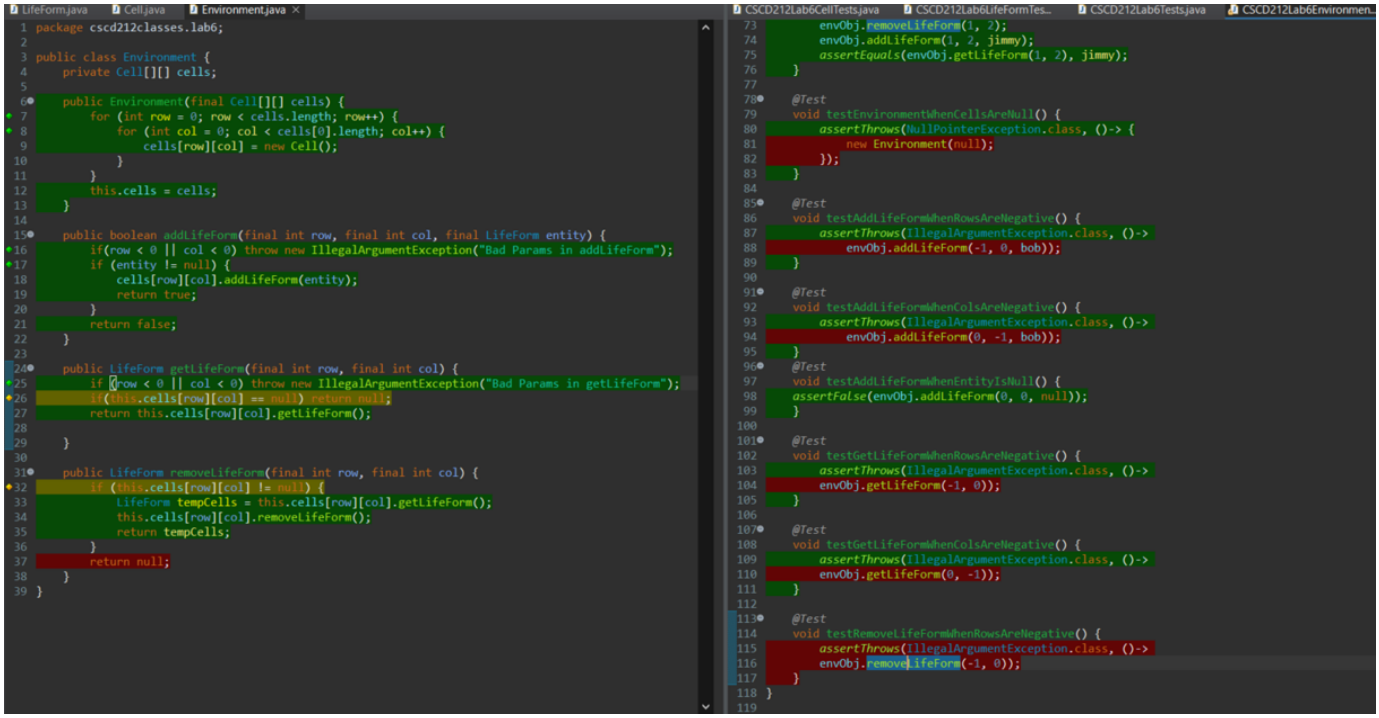


testGetLifeFormWhenRowsAreNegative:





testRemoveLifeFormWhenRowsAreNegative:

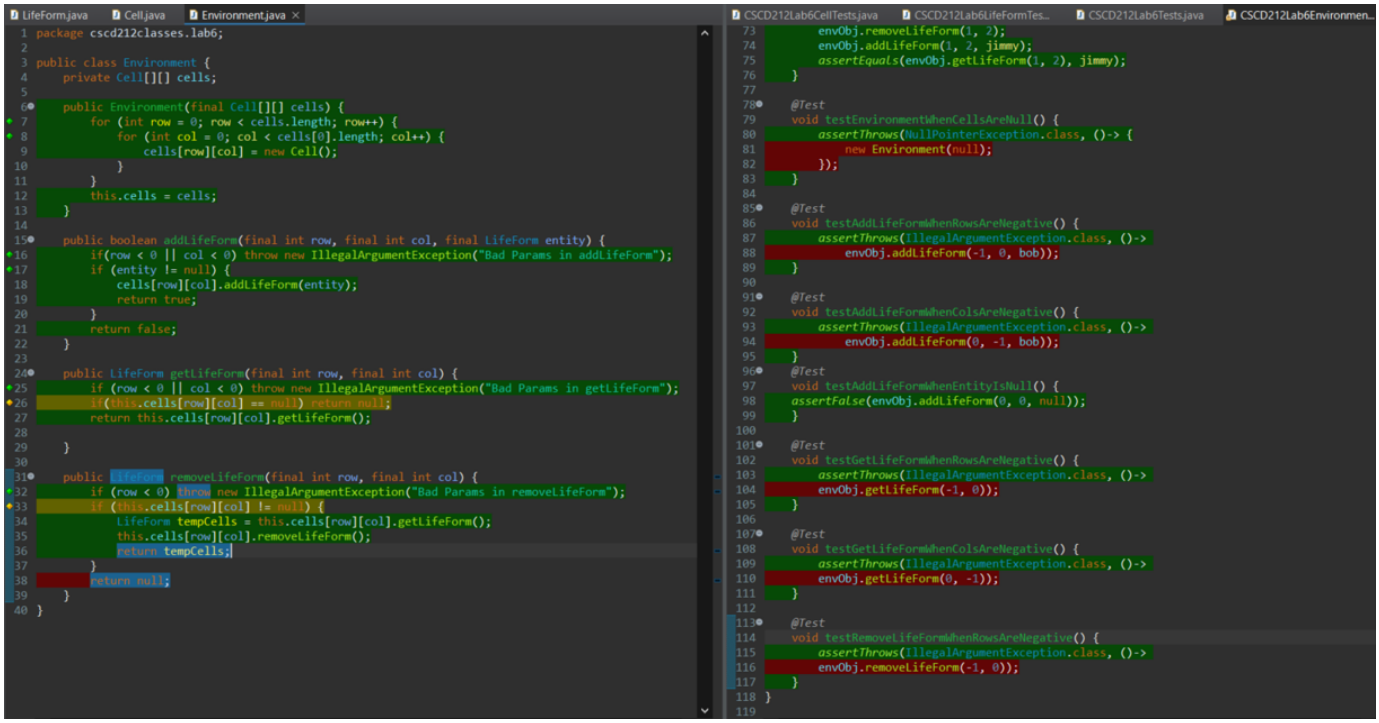


```
LifeForm.java  Cell.java  Environment.java ×
```

```java
package cscd212classes.lab6;

public class Environment {
    private Cell[][] cells;

    public Environment(final Cell[][] cells) {
        for (int row = 0; row < cells.length; row++) {
            for (int col = 0; col < cells[0].length; col++) {
                cells[row][col] = new Cell();
            }
        }
        this.cells = cells;
    }

    public boolean addLifeForm(final int row, final int col, final LifeForm entity) {
        if(row < 0 || col < 0) throw new IllegalArgumentException("Bad Params in addLifeForm");
        if (entity != null) {
            cells[row][col].addLifeForm(entity);
            return true;
        }
        return false;
    }

    public LifeForm getLifeForm(final int row, final int col) {
        if (row < 0 || col < 0) throw new IllegalArgumentException("Bad Params in getLifeForm");
        if(this.cells[row][col] == null) return null;
        return this.cells[row][col].getLifeForm();

    }

    public LifeForm removeLifeForm(final int row, final int col) {
        if (this.cells[row][col] != null) {
            LifeForm tempCells = this.cells[row][col].getLifeForm();
            this.cells[row][col].removeLifeForm();
            return tempCells;
        }
        return null;
    }
}
```

```
CSCD212Lab6CellTests.java  CSCD212Lab6LifeFormTes...  CSCD212Lab6Tests.java  CSCD212Lab6Environmen...
```

```java
        envObj.removeLifeForm(1, 2);
        envObj.addLifeForm(1, 2, jimmy);
        assertEquals(envObj.getLifeForm(1, 2), jimmy);
    }

    @Test
    void testEnvironmentWhenCellsAreNull() {
        assertThrows(NullPointerException.class, ()-> {
            new Environment(null);
        });
    }

    @Test
    void testAddLifeFormWhenRowsAreNegative() {
        assertThrows(IllegalArgumentException.class, ()->
            envObj.addLifeForm(-1, 0, bob));
    }

    @Test
    void testAddLifeFormWhenColsAreNegative() {
        assertThrows(IllegalArgumentException.class, ()->
            envObj.addLifeForm(0, -1, bob));
    }
    @Test
    void testAddLifeFormWhenEntityIsNull() {
    assertFalse(envObj.addLifeForm(0, 0, null));
    }

    @Test
    void testGetLifeFormWhenRowsAreNegative() {
        assertThrows(IllegalArgumentException.class, ()->
        envObj.getLifeForm(-1, 0));
    }

    @Test
    void testGetLifeFormWhenColsAreNegative() {
        assertThrows(IllegalArgumentException.class, ()->
        envObj.getLifeForm(0, -1));
    }

    @Test
    void testRemoveLifeFormWhenRowsAreNegative() {
        assertThrows(IllegalArgumentException.class, ()->
        envObj.removeLifeForm(-1, 0));
    }
}
```



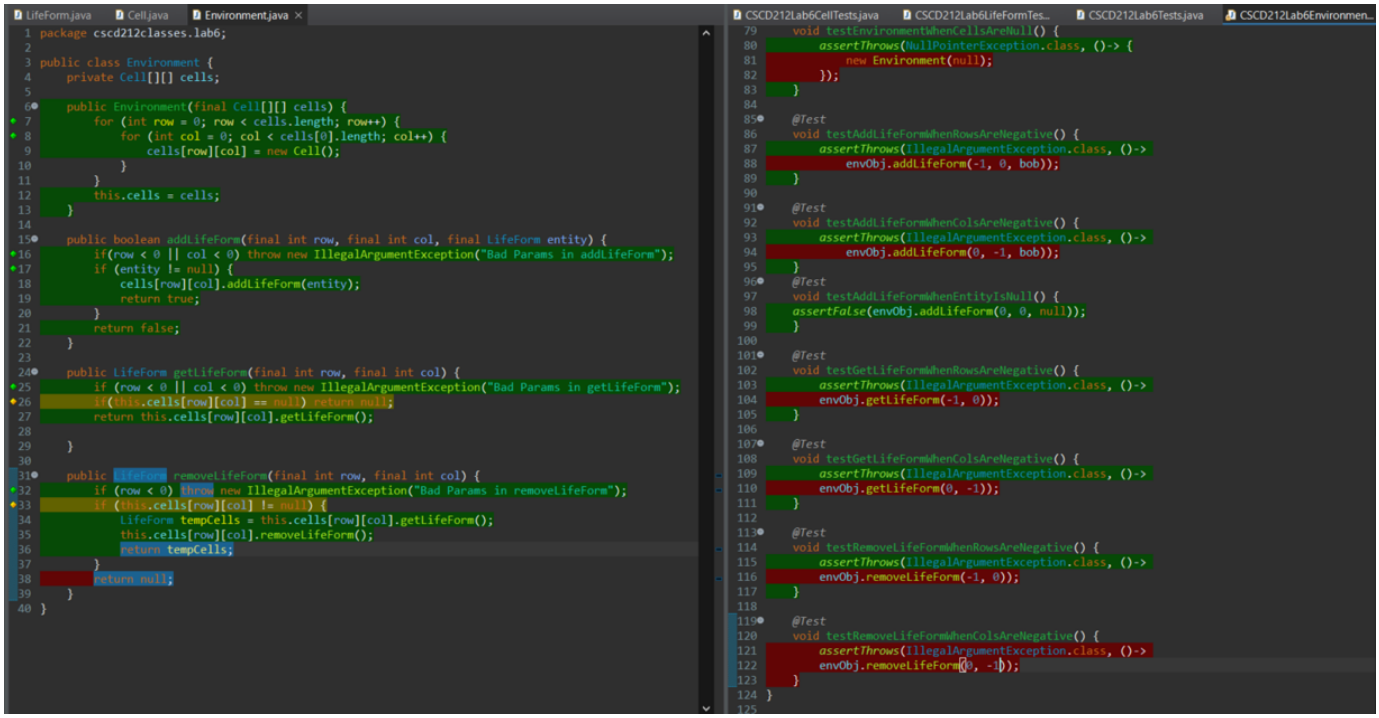
```
LifeForm.java  Cell.java  Environment.java ×
```

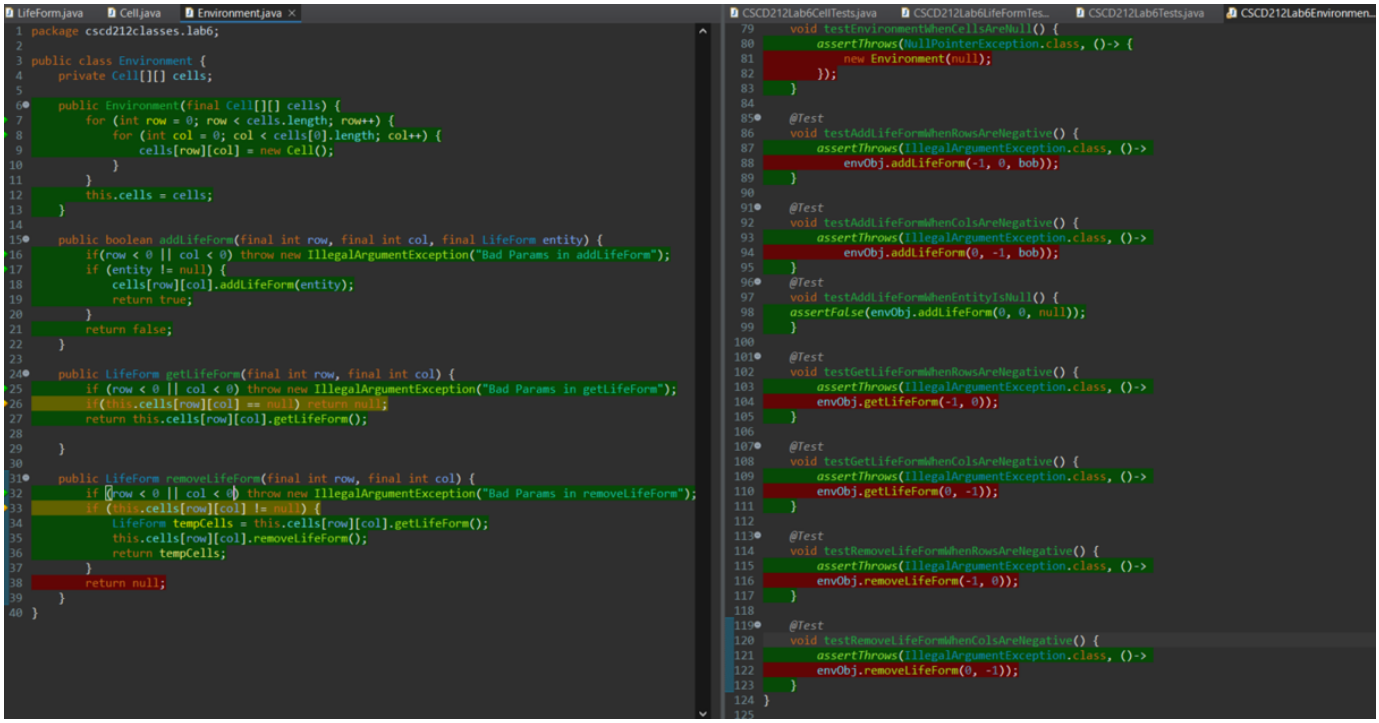
```java
package cscd212classes.lab6;

public class Environment {
    private Cell[][] cells;

    public Environment(final Cell[][] cells) {
        for (int row = 0; row < cells.length; row++) {
            for (int col = 0; col < cells[0].length; col++) {
                cells[row][col] = new Cell();
            }
        }
        this.cells = cells;
    }

    public boolean addLifeForm(final int row, final int col, final LifeForm entity) {
        if(row < 0 || col < 0) throw new IllegalArgumentException("Bad Params in addLifeForm");
        if (entity != null) {
            cells[row][col].addLifeForm(entity);
            return true;
        }
        return false;
    }

    public LifeForm getLifeForm(final int row, final int col) {
        if (row < 0 || col < 0) throw new IllegalArgumentException("Bad Params in getLifeForm");
        if(this.cells[row][col] == null) return null;
        return this.cells[row][col].getLifeForm();

    }

    public LifeForm removeLifeForm(final int row, final int col) {
        if (row < 0) throw new IllegalArgumentException("Bad Params in removeLifeForm");
        if (this.cells[row][col] != null) {
            LifeForm tempCells = this.cells[row][col].getLifeForm();
            this.cells[row][col].removeLifeForm();
            return tempCells;
        }
        return null;
    }
}
```

```
CSCD212Lab6CellTests.java  CSCD212Lab6LifeFormTes...  CSCD212Lab6Tests.java  CSCD212Lab6Environmen...
```

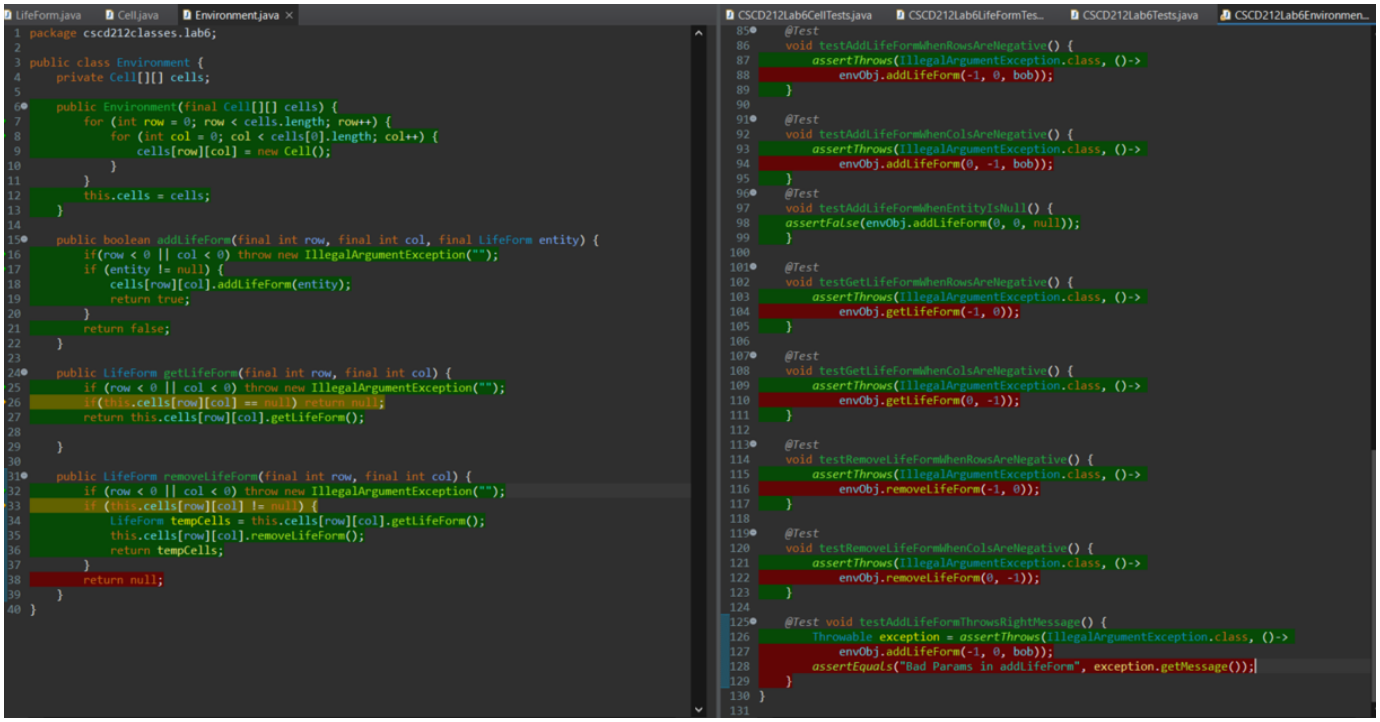
```java
        envObj.removeLifeForm(1, 2);
        envObj.addLifeForm(1, 2, jimmy);
        assertEquals(envObj.getLifeForm(1, 2), jimmy);
    }

    @Test
    void testEnvironmentWhenCellsAreNull() {
        assertThrows(NullPointerException.class, ()-> {
            new Environment(null);
        });
    }

    @Test
    void testAddLifeFormWhenRowsAreNegative() {
        assertThrows(IllegalArgumentException.class, ()->
            envObj.addLifeForm(-1, 0, bob));
    }

    @Test
    void testAddLifeFormWhenColsAreNegative() {
        assertThrows(IllegalArgumentException.class, ()->
            envObj.addLifeForm(0, -1, bob));
    }
    @Test
    void testAddLifeFormWhenEntityIsNull() {
    assertFalse(envObj.addLifeForm(0, 0, null));
    }

    @Test
    void testGetLifeFormWhenRowsAreNegative() {
        assertThrows(IllegalArgumentException.class, ()->
        envObj.getLifeForm(-1, 0));
    }

    @Test
    void testGetLifeFormWhenColsAreNegative() {
        assertThrows(IllegalArgumentException.class, ()->
        envObj.getLifeForm(0, -1));
    }

    @Test
    void testRemoveLifeFormWhenRowsAreNegative() {
        assertThrows(IllegalArgumentException.class, ()->
        envObj.removeLifeForm(-1, 0));
    }
}
```

testRemoveLifeFormWhenColsAreNegative:



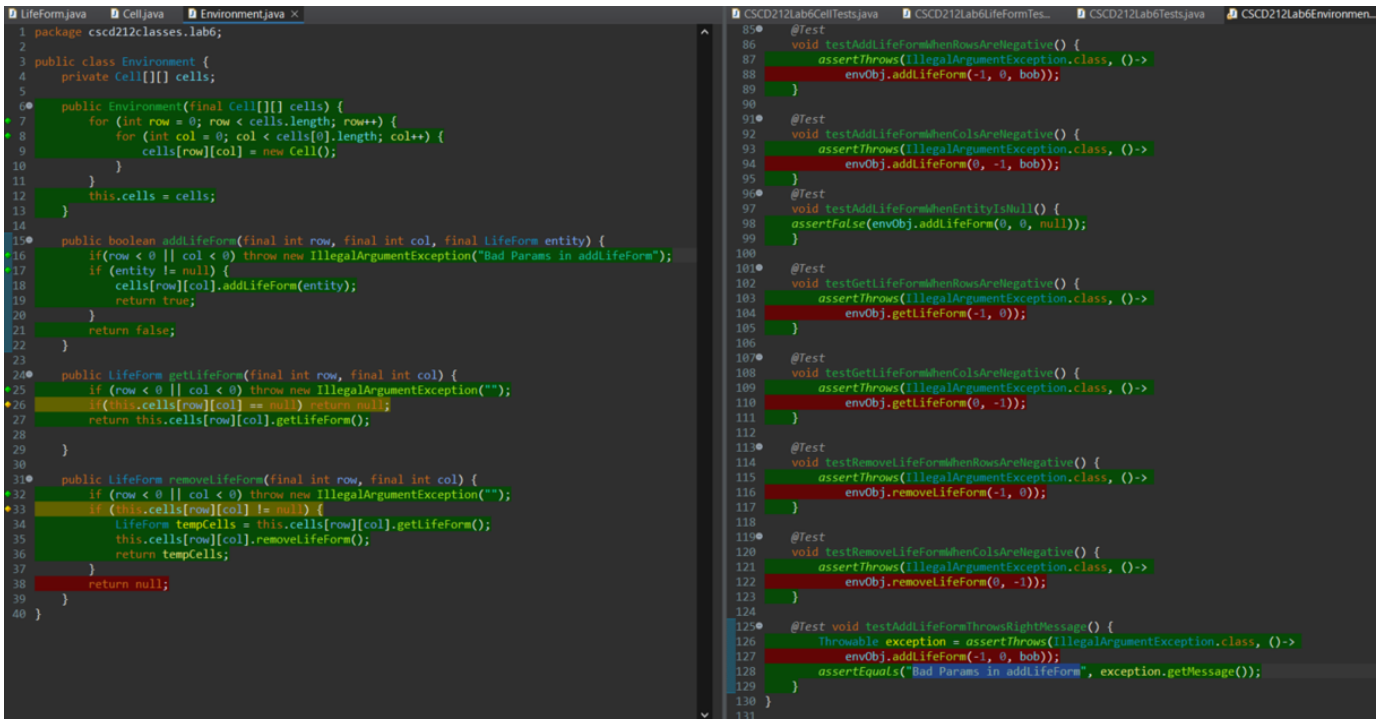


testAddLifeFormThrowsRightMessage:



```
LifeForm.java  Cell.java  Environment.java ×
1  package cscd212classes.lab6;
2
3  public class Environment {
4      private Cell[][] cells;
5
6      public Environment(final Cell[][] cells) {
7          for (int row = 0; row < cells.length; row++) {
8              for (int col = 0; col < cells[0].length; col++) {
9                  cells[row][col] = new Cell();
10             }
11         }
12         this.cells = cells;
13     }
14
15     public boolean addLifeForm(final int row, final int col, final LifeForm entity) {
16         if(row < 0 || col < 0) throw new IllegalArgumentException("");
17         if (entity != null) {
18             cells[row][col].addLifeForm(entity);
19             return true;
20         }
21         return false;
22     }
23
24     public LifeForm getLifeForm(final int row, final int col) {
25         if (row < 0 || col < 0) throw new IllegalArgumentException("");
26         if(this.cells[row][col] == null) return null;
27         return this.cells[row][col].getLifeForm();
28
29     }
30
31     public LifeForm removeLifeForm(final int row, final int col) {
32         if (row < 0 || col < 0) throw new IllegalArgumentException("");
33         if (this.cells[row][col] != null) {
34             LifeForm tempCells = this.cells[row][col].getLifeForm();
35             this.cells[row][col].removeLifeForm();
36             return tempCells;
37         }
38         return null;
39     }
40 }
```

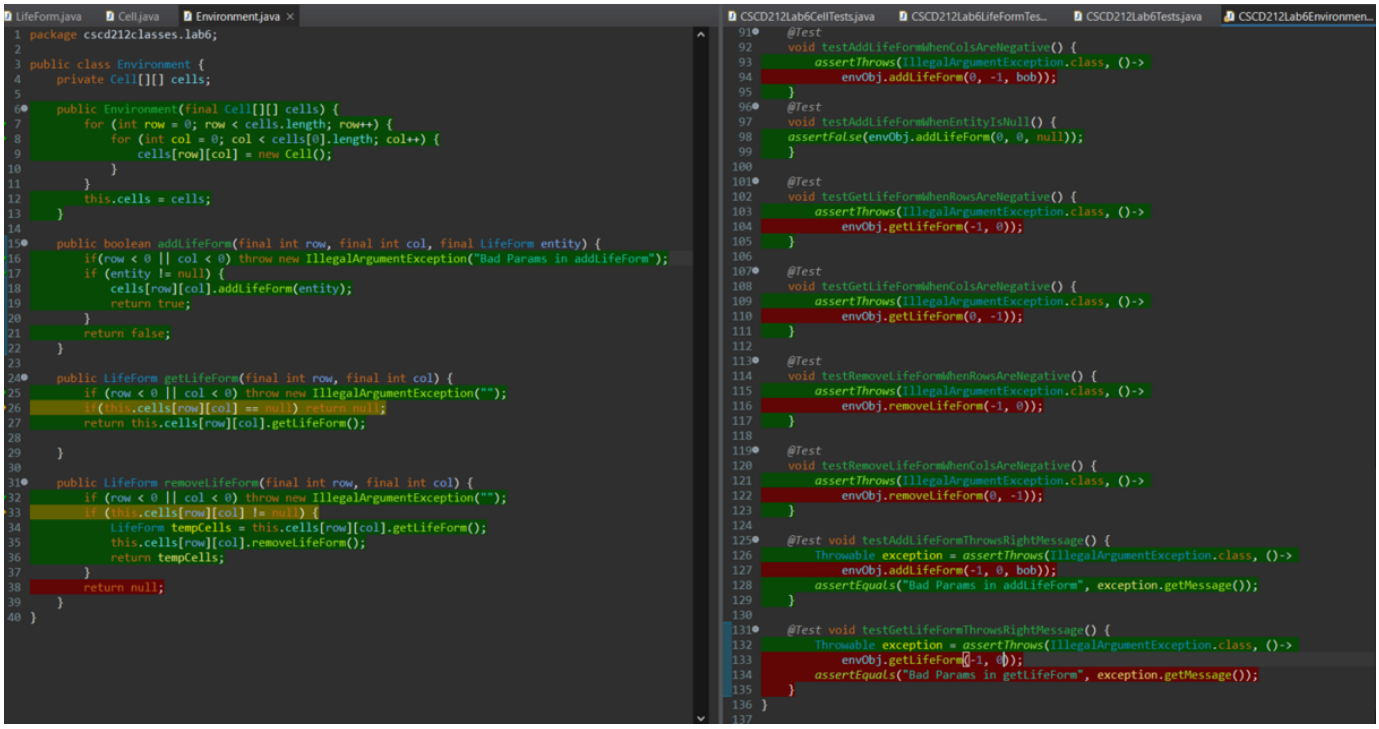
```
CSCD212Lab6CellTests.java  CSCD212Lab6LifeFormTes...  CSCD212Lab6Tests.java  CSCD212Lab6Environmen...
85     @Test
86     void testAddLifeFormWhenRowsAreNegative() {
87         assertThrows(IllegalArgumentException.class, ()->
88             envObj.addLifeForm(-1, 0, bob));
89     }
90
91     @Test
92     void testAddLifeFormWhenColsAreNegative() {
93         assertThrows(IllegalArgumentException.class, ()->
94             envObj.addLifeForm(0, -1, bob));
95     }
96     @Test
97     void testAddLifeFormWhenEntityIsNull() {
98     assertFalse(envObj.addLifeForm(0, 0, null));
99     }
100
101    @Test
102    void testGetLifeFormWhenRowsAreNegative() {
103        assertThrows(IllegalArgumentException.class, ()->
104            envObj.getLifeForm(-1, 0));
105    }
106
107    @Test
108    void testGetLifeFormWhenColsAreNegative() {
109        assertThrows(IllegalArgumentException.class, ()->
110            envObj.getLifeForm(0, -1));
111    }
112
113    @Test
114    void testRemoveLifeFormWhenRowsAreNegative() {
115        assertThrows(IllegalArgumentException.class, ()->
116            envObj.removeLifeForm(-1, 0));
117    }
118
119    @Test
120    void testRemoveLifeFormWhenColsAreNegative() {
121        assertThrows(IllegalArgumentException.class, ()->
122            envObj.removeLifeForm(0, -1));
123    }
124
125    @Test void testAddLifeFormThrowsRightMessage() {
126        Throwable exception = assertThrows(IllegalArgumentException.class, ()->
127            envObj.addLifeForm(-1, 0, bob));
128        assertEquals("Bad Params in addLifeForm", exception.getMessage());
129    }
130 }
131
```



```
LifeForm.java  Cell.java  Environment.java ×
1  package cscd212classes.lab6;
2
3  public class Environment {
4      private Cell[][] cells;
5
6      public Environment(final Cell[][] cells) {
7          for (int row = 0; row < cells.length; row++) {
8              for (int col = 0; col < cells[0].length; col++) {
9                  cells[row][col] = new Cell();
10             }
11         }
12         this.cells = cells;
13     }
14
15     public boolean addLifeForm(final int row, final int col, final LifeForm entity) {
16         if(row < 0 || col < 0) throw new IllegalArgumentException("Bad Params in addLifeForm");
17         if (entity != null) {
18             cells[row][col].addLifeForm(entity);
19             return true;
20         }
21         return false;
22     }
23
24     public LifeForm getLifeForm(final int row, final int col) {
25         if (row < 0 || col < 0) throw new IllegalArgumentException("");
26         if(this.cells[row][col] == null) return null;
27         return this.cells[row][col].getLifeForm();
28
29     }
30
31     public LifeForm removeLifeForm(final int row, final int col) {
32         if (row < 0 || col < 0) throw new IllegalArgumentException("");
33         if (this.cells[row][col] != null) {
34             LifeForm tempCells = this.cells[row][col].getLifeForm();
35             this.cells[row][col].removeLifeForm();
36             return tempCells;
37         }
38         return null;
39     }
40 }
```

```
CSCD212Lab6CellTests.java  CSCD212Lab6LifeFormTes...  CSCD212Lab6Tests.java  CSCD212Lab6Environmen...
85     @Test
86     void testAddLifeFormWhenRowsAreNegative() {
87         assertThrows(IllegalArgumentException.class, ()->
88             envObj.addLifeForm(-1, 0, bob));
89     }
90
91     @Test
92     void testAddLifeFormWhenColsAreNegative() {
93         assertThrows(IllegalArgumentException.class, ()->
94             envObj.addLifeForm(0, -1, bob));
95     }
96     @Test
97     void testAddLifeFormWhenEntityIsNull() {
98     assertFalse(envObj.addLifeForm(0, 0, null));
99     }
100
101    @Test
102    void testGetLifeFormWhenRowsAreNegative() {
103        assertThrows(IllegalArgumentException.class, ()->
104            envObj.getLifeForm(-1, 0));
105    }
106
107    @Test
108    void testGetLifeFormWhenColsAreNegative() {
109        assertThrows(IllegalArgumentException.class, ()->
110            envObj.getLifeForm(0, -1));
111    }
112
113    @Test
114    void testRemoveLifeFormWhenRowsAreNegative() {
115        assertThrows(IllegalArgumentException.class, ()->
116            envObj.removeLifeForm(-1, 0));
117    }
118
119    @Test
120    void testRemoveLifeFormWhenColsAreNegative() {
121        assertThrows(IllegalArgumentException.class, ()->
122            envObj.removeLifeForm(0, -1));
123    }
124
125    @Test void testAddLifeFormThrowsRightMessage() {
126        Throwable exception = assertThrows(IllegalArgumentException.class, ()->
127            envObj.addLifeForm(-1, 0, bob));
128        assertEquals("Bad Params in addLifeForm", exception.getMessage());
129    }
130 }
131
```

testGetLifeFormThrowsRightMessage:

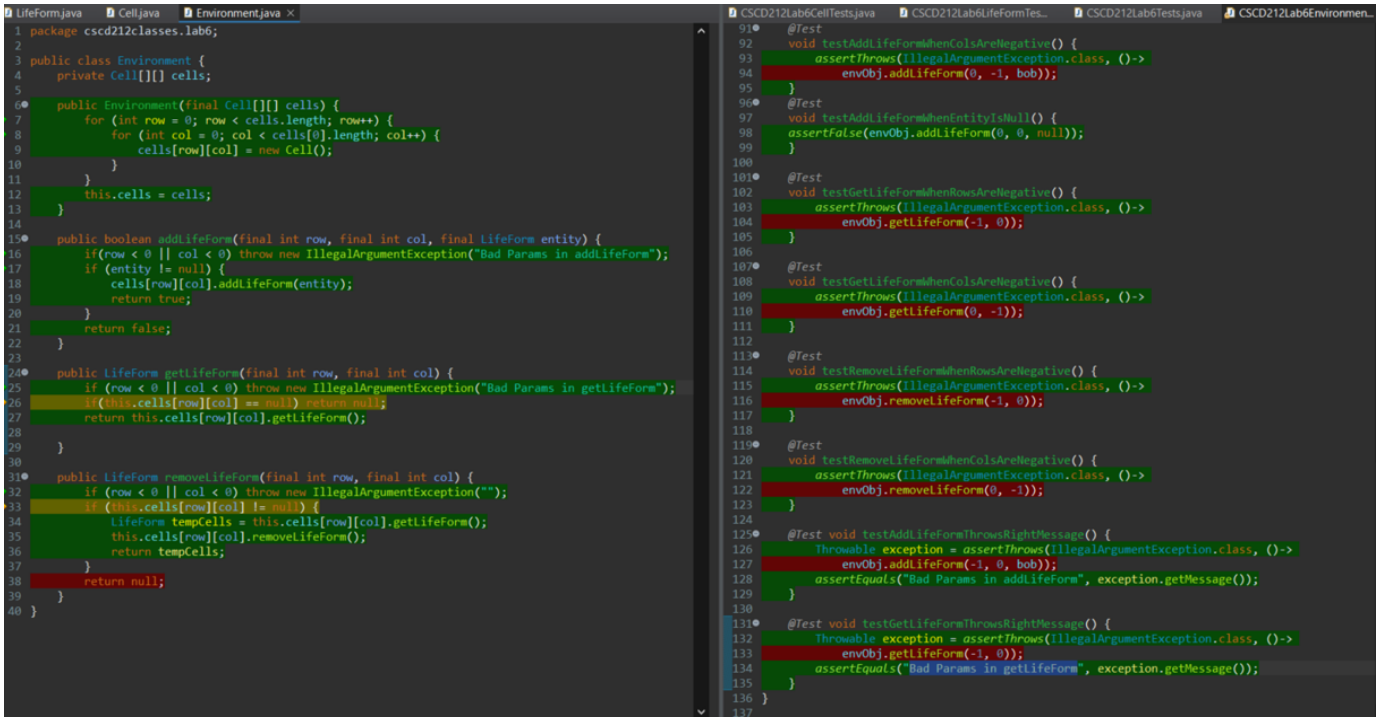


```
LifeForm.java  Cell.java  Environment.java ×
```

```java
package cscd212classes.lab6;

public class Environment {
    private Cell[][] cells;

    public Environment(final Cell[][] cells) {
        for (int row = 0; row < cells.length; row++) {
            for (int col = 0; col < cells[0].length; col++) {
                cells[row][col] = new Cell();
            }
        }
        this.cells = cells;
    }

    public boolean addLifeForm(final int row, final int col, final LifeForm entity) {
        if(row < 0 || col < 0) throw new IllegalArgumentException("Bad Params in addLifeForm");
        if (entity != null) {
            cells[row][col].addLifeForm(entity);
            return true;
        }
        return false;
    }

    public LifeForm getLifeForm(final int row, final int col) {
        if (row < 0 || col < 0) throw new IllegalArgumentException("");
        if(this.cells[row][col] == null) return null;
        return this.cells[row][col].getLifeForm();

    }

    public LifeForm removeLifeForm(final int row, final int col) {
        if (row < 0 || col < 0) throw new IllegalArgumentException("");
        if (this.cells[row][col] != null) {
            LifeForm tempCells = this.cells[row][col].getLifeForm();
            this.cells[row][col].removeLifeForm();
            return tempCells;
        }
        return null;
    }
}
```

```
CSCD212Lab6CellTests.java  CSCD212Lab6LifeFormTes...  CSCD212Lab6Tests.java  CSCD212Lab6Environmen...
```

```java
    @Test
    void testAddLifeFormWhenColsAreNegative() {
        assertThrows(IllegalArgumentException.class, ()->
            envObj.addLifeForm(0, -1, bob));
    }
    @Test
    void testAddLifeFormWhenEntityIsNull() {
    assertFalse(envObj.addLifeForm(0, 0, null));
    }

    @Test
    void testGetLifeFormWhenRowsAreNegative() {
        assertThrows(IllegalArgumentException.class, ()->
            envObj.getLifeForm(-1, 0));
    }

    @Test
    void testGetLifeFormWhenColsAreNegative() {
        assertThrows(IllegalArgumentException.class, ()->
            envObj.getLifeForm(0, -1));
    }

    @Test
    void testRemoveLifeFormWhenRowsAreNegative() {
        assertThrows(IllegalArgumentException.class, ()->
            envObj.removeLifeForm(-1, 0));
    }

    @Test
    void testRemoveLifeFormWhenColsAreNegative() {
        assertThrows(IllegalArgumentException.class, ()->
            envObj.removeLifeForm(0, -1));
    }

    @Test void testAddLifeFormThrowsRightMessage() {
        Throwable exception = assertThrows(IllegalArgumentException.class, ()->
            envObj.addLifeForm(-1, 0, bob));
        assertEquals("Bad Params in addLifeForm", exception.getMessage());
    }

    @Test void testGetLifeFormThrowsRightMessage() {
        Throwable exception = assertThrows(IllegalArgumentException.class, ()->
            envObj.getLifeForm(-1, 0));
        assertEquals("Bad Params in getLifeForm", exception.getMessage());
    }
}
```



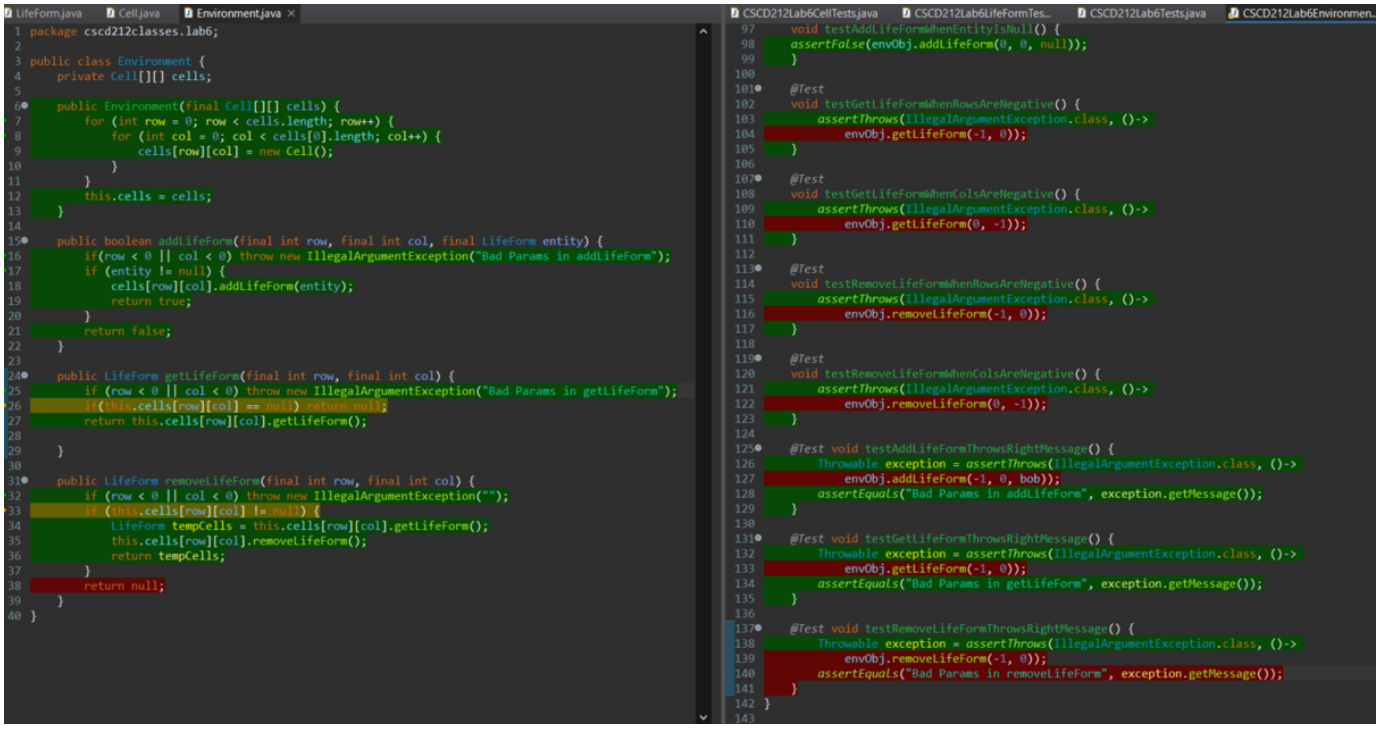
```
LifeForm.java  Cell.java  Environment.java ×
```

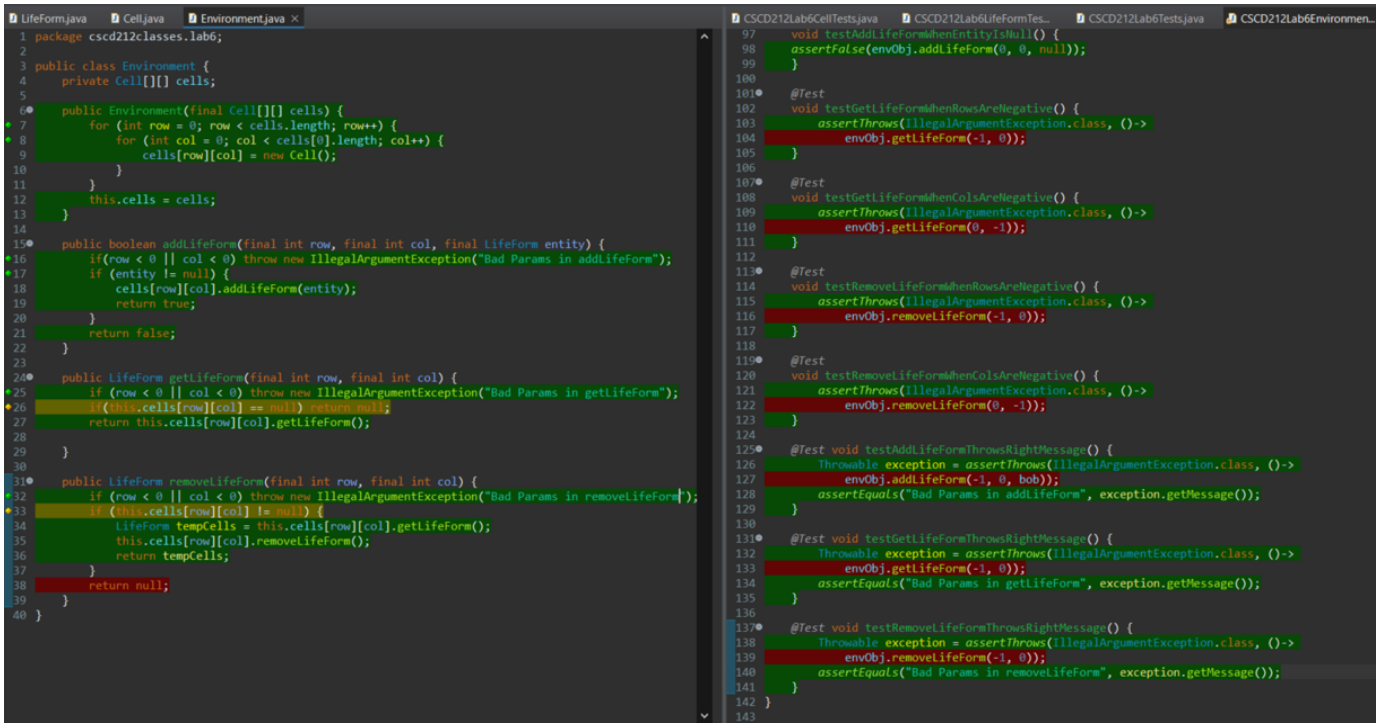
```java
package cscd212classes.lab6;

public class Environment {
    private Cell[][] cells;

    public Environment(final Cell[][] cells) {
        for (int row = 0; row < cells.length; row++) {
            for (int col = 0; col < cells[0].length; col++) {
                cells[row][col] = new Cell();
            }
        }
        this.cells = cells;
    }

    public boolean addLifeForm(final int row, final int col, final LifeForm entity) {
        if(row < 0 || col < 0) throw new IllegalArgumentException("Bad Params in addLifeForm");
        if (entity != null) {
            cells[row][col].addLifeForm(entity);
            return true;
        }
        return false;
    }

    public LifeForm getLifeForm(final int row, final int col) {
        if (row < 0 || col < 0) throw new IllegalArgumentException("Bad Params in getLifeForm");
        if(this.cells[row][col] == null) return null;
        return this.cells[row][col].getLifeForm();

    }

    public LifeForm removeLifeForm(final int row, final int col) {
        if (row < 0 || col < 0) throw new IllegalArgumentException("");
        if (this.cells[row][col] != null) {
            LifeForm tempCells = this.cells[row][col].getLifeForm();
            this.cells[row][col].removeLifeForm();
            return tempCells;
        }
        return null;
    }
}
```

```
CSCD212Lab6CellTests.java  CSCD212Lab6LifeFormTes...  CSCD212Lab6Tests.java  CSCD212Lab6Environmen...
```

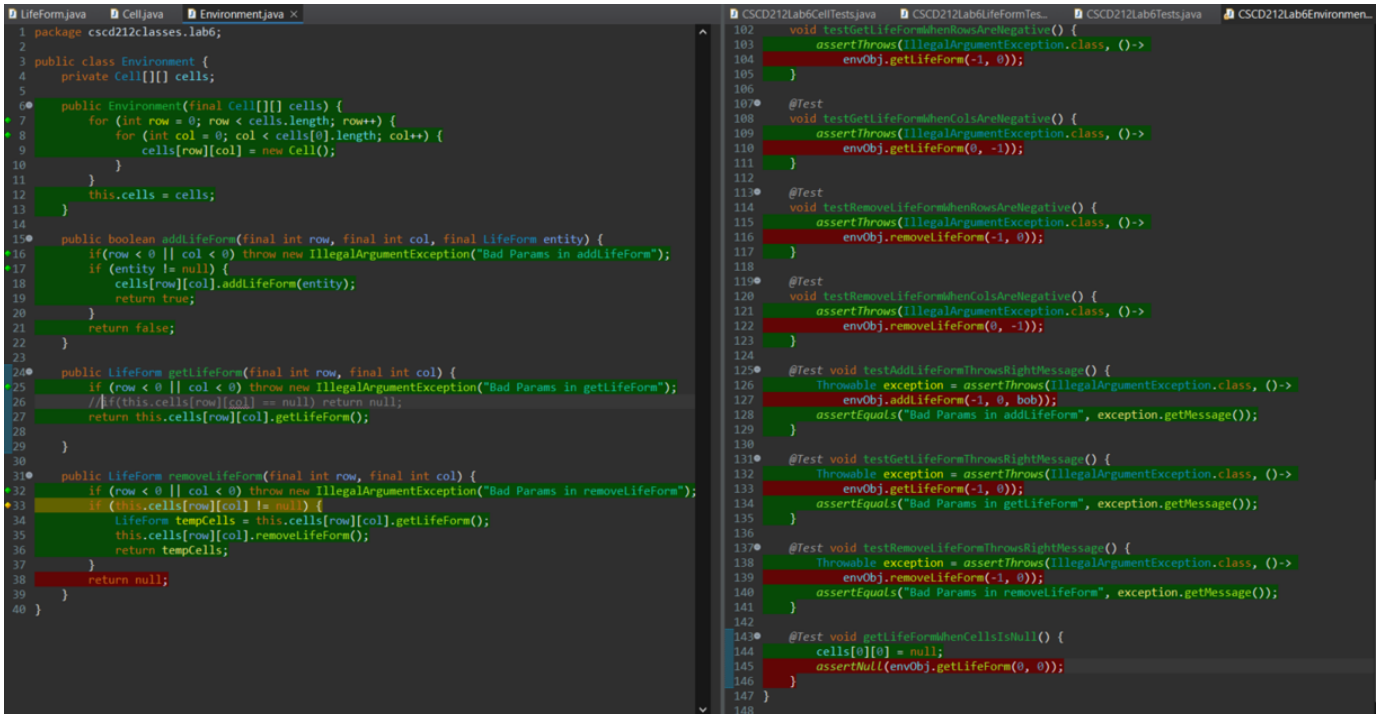
```java
    @Test
    void testAddLifeFormWhenColsAreNegative() {
        assertThrows(IllegalArgumentException.class, ()->
            envObj.addLifeForm(0, -1, bob));
    }
    @Test
    void testAddLifeFormWhenEntityIsNull() {
    assertFalse(envObj.addLifeForm(0, 0, null));
    }

    @Test
    void testGetLifeFormWhenRowsAreNegative() {
        assertThrows(IllegalArgumentException.class, ()->
            envObj.getLifeForm(-1, 0));
    }

    @Test
    void testGetLifeFormWhenColsAreNegative() {
        assertThrows(IllegalArgumentException.class, ()->
            envObj.getLifeForm(0, -1));
    }

    @Test
    void testRemoveLifeFormWhenRowsAreNegative() {
        assertThrows(IllegalArgumentException.class, ()->
            envObj.removeLifeForm(-1, 0));
    }

    @Test
    void testRemoveLifeFormWhenColsAreNegative() {
        assertThrows(IllegalArgumentException.class, ()->
            envObj.removeLifeForm(0, -1));
    }

    @Test void testAddLifeFormThrowsRightMessage() {
        Throwable exception = assertThrows(IllegalArgumentException.class, ()->
            envObj.addLifeForm(-1, 0, bob));
        assertEquals("Bad Params in addLifeForm", exception.getMessage());
    }

    @Test void testGetLifeFormThrowsRightMessage() {
        Throwable exception = assertThrows(IllegalArgumentException.class, ()->
            envObj.getLifeForm(-1, 0));
        assertEquals("Bad Params in getLifeForm", exception.getMessage());
    }
}
```

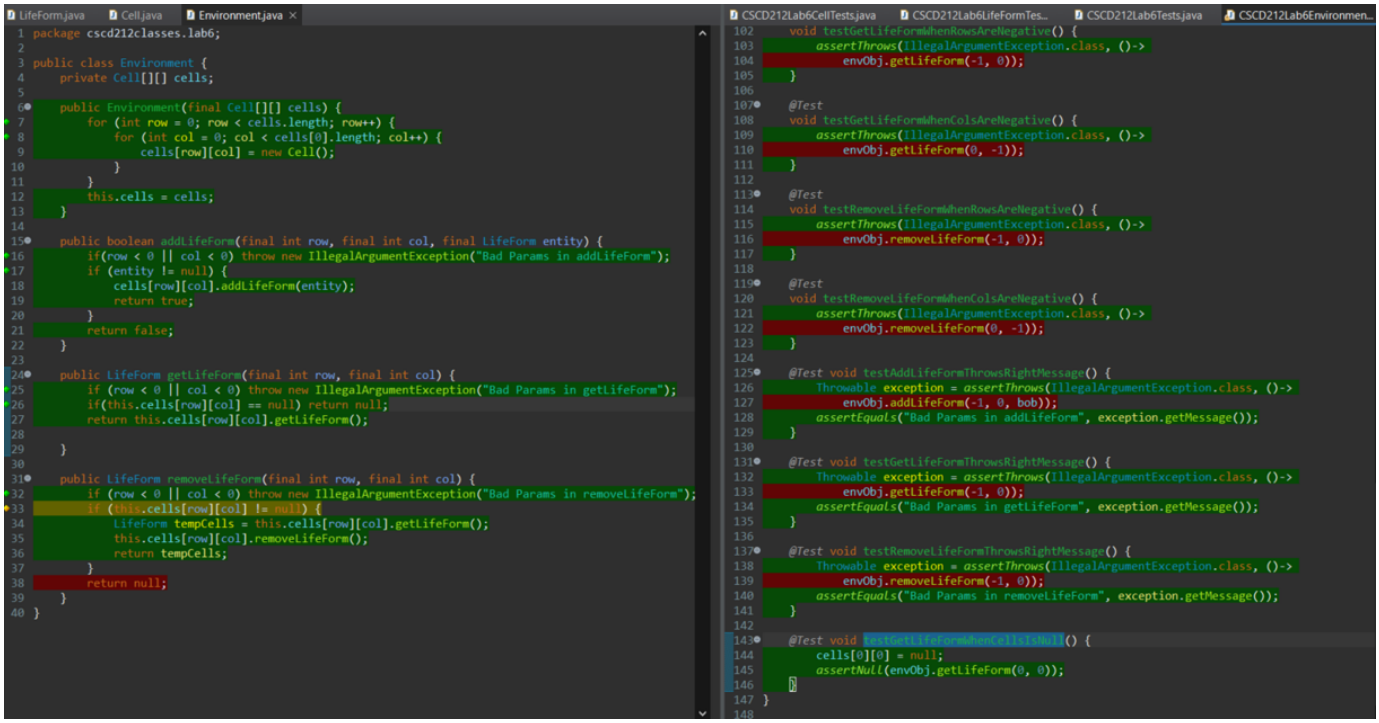
testRemoveLifeFormThrowsRightMessage:



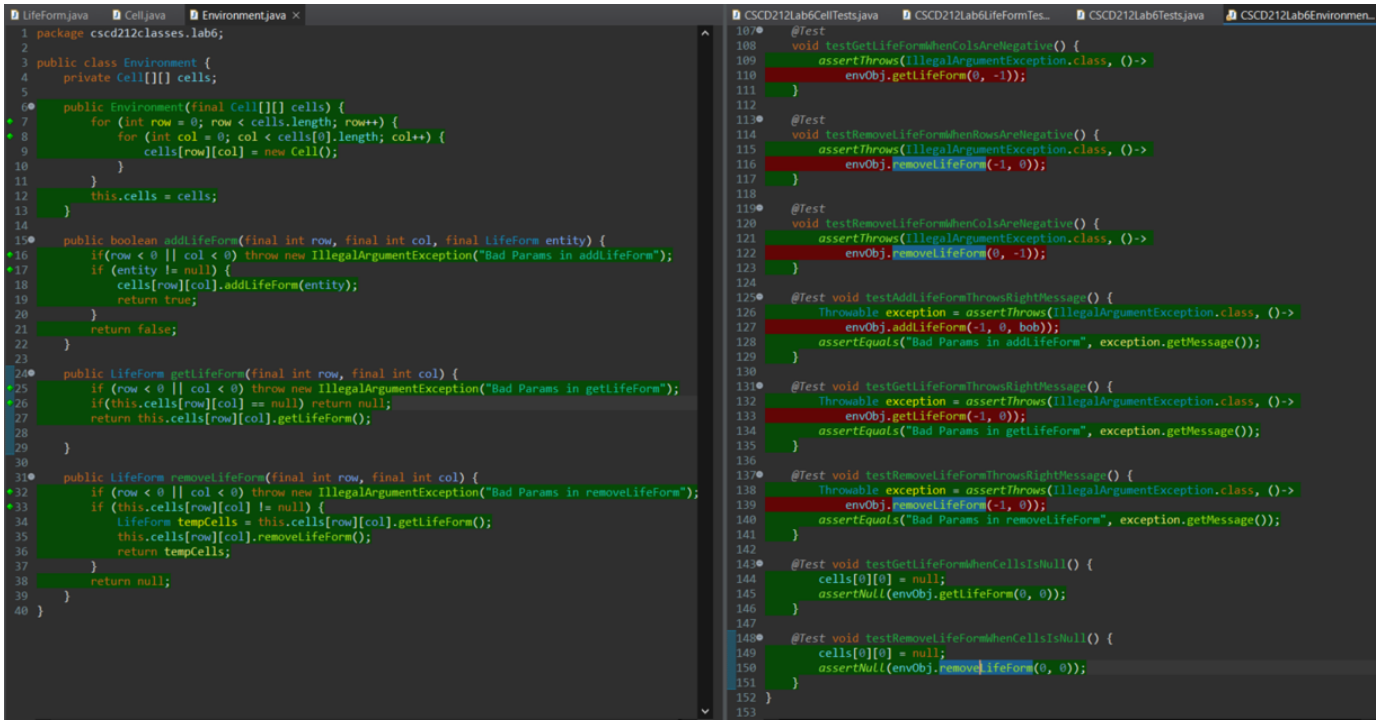


testGetLifeFormWhenCellsIsNull:





testRemoveLifeFormWhenCellsIsNull: *passed right away



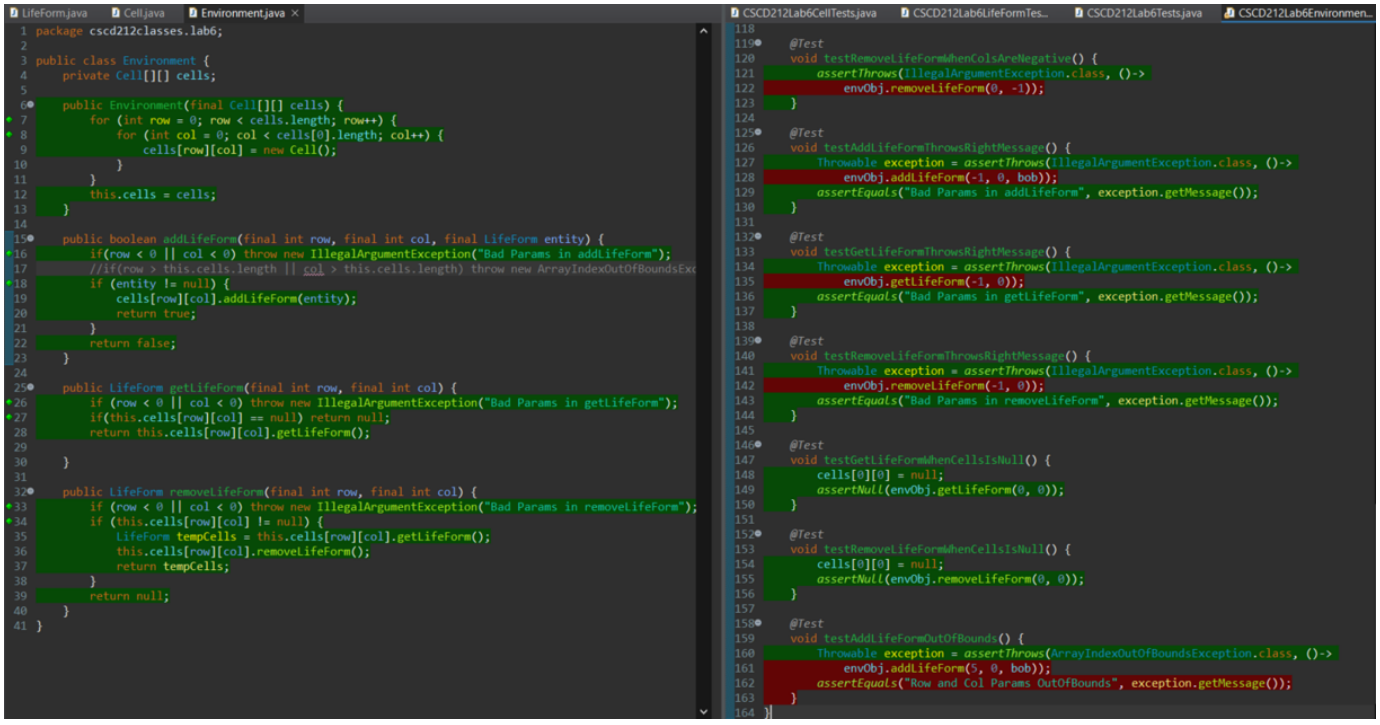
```
LifeForm.java  Cell.java  Environment.java ×
```

```java
package cscd212classes.lab6;

public class Environment {
    private Cell[][] cells;

    public Environment(final Cell[][] cells) {
        for (int row = 0; row < cells.length; row++) {
            for (int col = 0; col < cells[0].length; col++) {
                cells[row][col] = new Cell();
            }
        }
        this.cells = cells;
    }

    public boolean addLifeForm(final int row, final int col, final LifeForm entity) {
        if(row < 0 || col < 0) throw new IllegalArgumentException("Bad Params in addLifeForm");
        if (entity != null) {
            cells[row][col].addLifeForm(entity);
            return true;
        }
        return false;
    }

    public LifeForm getLifeForm(final int row, final int col) {
        if (row < 0 || col < 0) throw new IllegalArgumentException("Bad Params in getLifeForm");
        if(this.cells[row][col] == null) return null;
        return this.cells[row][col].getLifeForm();

    }

    public LifeForm removeLifeForm(final int row, final int col) {
        if (row < 0 || col < 0) throw new IllegalArgumentException("Bad Params in removeLifeForm");
        if (this.cells[row][col] != null) {
            LifeForm tempCells = this.cells[row][col].getLifeForm();
            this.cells[row][col].removeLifeForm();
            return tempCells;
        }
        return null;
    }
}
```

```
CSCD212Lab6CellTests.java  CSCD212Lab6LifeFormTes...  CSCD212Lab6Tests.java  CSCD212Lab6Environmen...
```

```java
    @Test
    void testGetLifeFormWhenColsAreNegative() {
        assertThrows(IllegalArgumentException.class, ()->
            envObj.getLifeForm(0, -1));
    }

    @Test
    void testRemoveLifeFormWhenRowsAreNegative() {
        assertThrows(IllegalArgumentException.class, ()->
            envObj.removeLifeForm(-1, 0));
    }

    @Test
    void testRemoveLifeFormWhenColsAreNegative() {
        assertThrows(IllegalArgumentException.class, ()->
            envObj.removeLifeForm(0, -1));
    }

    @Test void testAddLifeFormThrowsRightMessage() {
        Throwable exception = assertThrows(IllegalArgumentException.class, ()->
            envObj.addLifeForm(-1, 0, bob));
        assertEquals("Bad Params in addLifeForm", exception.getMessage());
    }

    @Test void testGetLifeFormThrowsRightMessage() {
        Throwable exception = assertThrows(IllegalArgumentException.class, ()->
            envObj.getLifeForm(-1, 0));
        assertEquals("Bad Params in getLifeForm", exception.getMessage());
    }

    @Test void testRemoveLifeFormThrowsRightMessage() {
        Throwable exception = assertThrows(IllegalArgumentException.class, ()->
            envObj.removeLifeForm(-1, 0));
        assertEquals("Bad Params in removeLifeForm", exception.getMessage());
    }

    @Test void testGetLifeFormWhenCellsIsNull() {
        cells[0][0] = null;
        assertNull(envObj.getLifeForm(0, 0));
    }

    @Test void testRemoveLifeFormWhenCellsIsNull() {
        cells[0][0] = null;
        assertNull(envObj.removeLifeForm(0, 0));
    }
}
```

testAddLifeFormOutOfBoundsProperMessage:

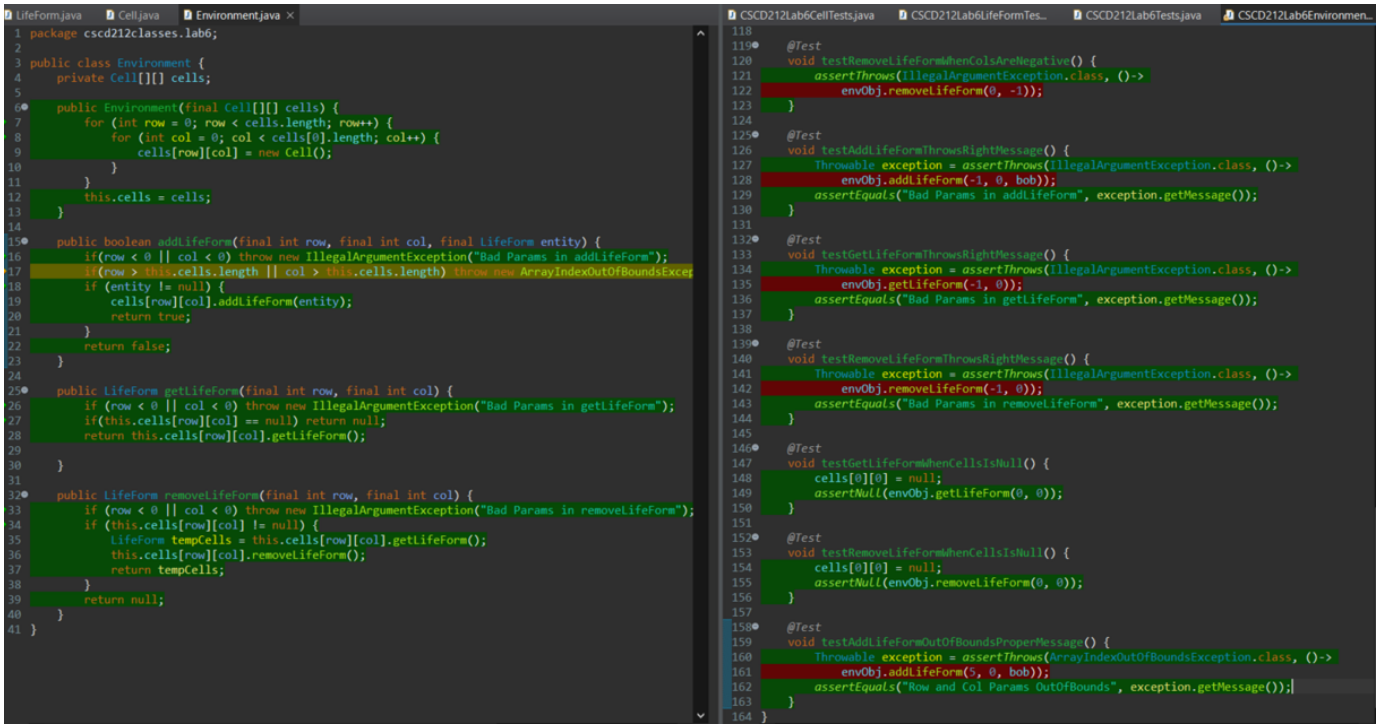


```
LifeForm.java  Cell.java  Environment.java ×
```

```java
package cscd212classes.lab6;

public class Environment {
    private Cell[][] cells;

    public Environment(final Cell[][] cells) {
        for (int row = 0; row < cells.length; row++) {
            for (int col = 0; col < cells[0].length; col++) {
                cells[row][col] = new Cell();
            }
        }
        this.cells = cells;
    }

    public boolean addLifeForm(final int row, final int col, final LifeForm entity) {
        if(row < 0 || col < 0) throw new IllegalArgumentException("Bad Params in addLifeForm");
        //if(row > this.cells.length || col > this.cells.length) throw new ArrayIndexOutOfBoundsExc
        if (entity != null) {
            cells[row][col].addLifeForm(entity);
            return true;
        }
        return false;
    }

    public LifeForm getLifeForm(final int row, final int col) {
        if (row < 0 || col < 0) throw new IllegalArgumentException("Bad Params in getLifeForm");
        if(this.cells[row][col] == null) return null;
        return this.cells[row][col].getLifeForm();

    }

    public LifeForm removeLifeForm(final int row, final int col) {
        if (row < 0 || col < 0) throw new IllegalArgumentException("Bad Params in removeLifeForm");
        if (this.cells[row][col] != null) {
            LifeForm tempCells = this.cells[row][col].getLifeForm();
            this.cells[row][col].removeLifeForm();
            return tempCells;
        }
        return null;
    }
}
```

```
CSCD212Lab6CellTests.java  CSCD212Lab6LifeFormTes...  CSCD212Lab6Tests.java  CSCD212Lab6Environmen...
```

```java

    @Test
    void testRemoveLifeFormWhenColsAreNegative() {
        assertThrows(IllegalArgumentException.class, ()->
            envObj.removeLifeForm(0, -1));
    }

    @Test
    void testAddLifeFormThrowsRightMessage() {
        Throwable exception = assertThrows(IllegalArgumentException.class, ()->
            envObj.addLifeForm(-1, 0, bob));
        assertEquals("Bad Params in addLifeForm", exception.getMessage());
    }

    @Test
    void testGetLifeFormThrowsRightMessage() {
        Throwable exception = assertThrows(IllegalArgumentException.class, ()->
            envObj.getLifeForm(-1, 0));
        assertEquals("Bad Params in getLifeForm", exception.getMessage());
    }

    @Test
    void testRemoveLifeFormThrowsRightMessage() {
        Throwable exception = assertThrows(IllegalArgumentException.class, ()->
            envObj.removeLifeForm(-1, 0));
        assertEquals("Bad Params in removeLifeForm", exception.getMessage());
    }

    @Test
    void testGetLifeFormWhenCellsIsNull() {
        cells[0][0] = null;
        assertNull(envObj.getLifeForm(0, 0));
    }

    @Test
    void testRemoveLifeFormWhenCellsIsNull() {
        cells[0][0] = null;
        assertNull(envObj.removeLifeForm(0, 0));
    }

    @Test
    void testAddLifeFormOutOfBounds() {
        Throwable exception = assertThrows(ArrayIndexOutOfBoundsException.class, ()->
            envObj.addLifeForm(5, 0, bob));
        assertEquals("Row and Col Params OutOfBounds", exception.getMessage());
    }
}
```



```
LifeForm.java  Cell.java  Environment.java ×
```

```java
package cscd212classes.lab6;

public class Environment {
    private Cell[][] cells;

    public Environment(final Cell[][] cells) {
        for (int row = 0; row < cells.length; row++) {
            for (int col = 0; col < cells[0].length; col++) {
                cells[row][col] = new Cell();
            }
        }
        this.cells = cells;
    }

    public boolean addLifeForm(final int row, final int col, final LifeForm entity) {
        if(row < 0 || col < 0) throw new IllegalArgumentException("Bad Params in addLifeForm");
        if(row > this.cells.length || col > this.cells.length) throw new ArrayIndexOutOfBoundsExcep
        if (entity != null) {
            cells[row][col].addLifeForm(entity);
            return true;
        }
        return false;
    }

    public LifeForm getLifeForm(final int row, final int col) {
        if (row < 0 || col < 0) throw new IllegalArgumentException("Bad Params in getLifeForm");
        if(this.cells[row][col] == null) return null;
        return this.cells[row][col].getLifeForm();

    }

    public LifeForm removeLifeForm(final int row, final int col) {
        if (row < 0 || col < 0) throw new IllegalArgumentException("Bad Params in removeLifeForm");
        if (this.cells[row][col] != null) {
            LifeForm tempCells = this.cells[row][col].getLifeForm();
            this.cells[row][col].removeLifeForm();
            return tempCells;
        }
        return null;
    }
}
```

```
CSCD212Lab6CellTests.java  CSCD212Lab6LifeFormTes...  CSCD212Lab6Tests.java  CSCD212Lab6Environmen...
```

```java
    @Test
    void testRemoveLifeFormWhenColsAreNegative() {
        assertThrows(IllegalArgumentException.class, ()->
            envObj.removeLifeForm(0, -1));
    }

    @Test
    void testAddLifeFormThrowsRightMessage() {
        Throwable exception = assertThrows(IllegalArgumentException.class, ()->
            envObj.addLifeForm(-1, 0, bob));
        assertEquals("Bad Params in addLifeForm", exception.getMessage());
    }

    @Test
    void testGetLifeFormThrowsRightMessage() {
        Throwable exception = assertThrows(IllegalArgumentException.class, ()->
            envObj.getLifeForm(-1, 0));
        assertEquals("Bad Params in getLifeForm", exception.getMessage());
    }

    @Test
    void testRemoveLifeFormThrowsRightMessage() {
        Throwable exception = assertThrows(IllegalArgumentException.class, ()->
            envObj.removeLifeForm(-1, 0));
        assertEquals("Bad Params in removeLifeForm", exception.getMessage());
    }

    @Test
    void testGetLifeFormWhenCellsIsNull() {
        cells[0][0] = null;
        assertNull(envObj.getLifeForm(0, 0));
    }

    @Test
    void testRemoveLifeFormWhenCellsIsNull() {
        cells[0][0] = null;
        assertNull(envObj.removeLifeForm(0, 0));
    }

    @Test
    void testAddLifeFormOutOfBoundsProperMessage() {
        Throwable exception = assertThrows(ArrayIndexOutOfBoundsException.class, ()->
            envObj.addLifeForm(5, 0, bob));
        assertEquals("Row and Col Params OutOfBounds", exception.getMessage());
    }
}
```

This assignment was hard, I really can't guarantee I did it right. I ran out of precondition ideas to check.