

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

The screenshot shows an IDE interface with two windows. The left window displays the file `CSCD212Lab6LifeFormTests.java`, which contains Java code for unit testing. The right window displays the file `LifeForm.java`, which contains the implementation of the `LifeForm` class. In the `CSCD212Lab6LifeFormTests.java` file, there are several assertion statements that are failing, indicated by red highlights. The failing assertions are:

```
13● @Test
14 void testInit() {
15     LifeForm entity;
16     entity = new LifeForm("Bob", 40);
17     assertEquals("Bob", entity.getName());
18     assertEquals(40, entity.getCurrentLifePoints());
```

In the `LifeForm.java` file, the failing assertions are:

```
16● public String getName() {
17     return this.name;
18 }
```

```
16● public int getCurrentLifePoints() {
17     return this.currentLifePoints;
18 }
```

```
CSCD212Lab6LifeFormTests.java
1 package cscd212tests.lab6;
2
3 //tells the compiler where to find the definition of specific JUnit classes that we need
4 import static org.junit.jupiter.api.Assertions.*;
5
6 import org.junit.jupiter.api.BeforeEach;
7 import org.junit.jupiter.api.Test;
8
9 import cscd212classes.lab6.LifeForm;
10
11 public class CSCD212Lab6LifeFormTests {
12     private LifeForm entity;
13
14     @BeforeEach
15     public void setup() throws Exception {
16         entity = new LifeForm("Bob", 40);
17     }
18
19
20     @Test
21     public void testGetName() {
22         assertEquals("Bob", entity.getName());
23     }
24
25     @Test
26     public void testGetCurrentLifePoints() {
27         assertEquals(40, entity.getCurrentLifePoints());
28     }
29
30 }

```

```
LifeForm.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         this.name = name;
9         this.currentLifePoints = currentLifePoints;
10    }
11
12    public String getName() {
13        return this.name;
14    }
15
16    public int getCurrentLifePoints() {
17        return this.currentLifePoints;
18    }
19
20 }
```

Testing addLifeForm in CellsTests

```
Cell.java
1 package cscd212classes.lab6;
2
3 public class Cell {
4     private LifeForm entity;
5
6     //Currently, this is only returning null
7     //Returns the LifeForm in the Cell.
8     public LifeForm getLifeForm() {
9         return this.entity;
10    }
11
12    /*
13     * Add a LifeForm as a private data member, store
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        return false;
20    }
21 }
```

```
CSCD212Lab6CellTests.java
1 package cscd212tests.lab6;
2
3 import static org.junit.jupiter.api.Assertions.*;
4
5 public class CSCD212Lab6CellTests {
6
7     @Test
8     void testInit() {
9         Cell cell = new Cell();
10        assertNull(cell.getLifeForm());
11    }
12
13    @Test
14    public void testAddLifeForm() {
15        LifeForm bob = new LifeForm("Bob", 40);
16        LifeForm fred = new LifeForm("Fred", 40);
17        Cell cell = new Cell();
18        boolean success = cell.addLifeForm(bob);
19        assertTrue(success);
20        assertEquals(bob, cell.getLifeForm());
21        success = cell.addLifeForm(fred);
22        assertFalse(success);
23        assertEquals(bob, cell.getLifeForm());
24    }
25
26 }
```

```

1 package cscd212classes.lab6;
2
3 public class Cell {
4     private LifeForm entity;
5
6     //Currently, this is only returning null
7     //Returns the LifeForm in the Cell.
8     public LifeForm getLifeForm() {
9         return this.entity;
10    }
11
12    /*
13     * Add a LifeForm as a private data member, store it in class level
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            this.entity = entity;
21            return true;
22        }
23        return false;
24    }
25 }
26

```

```

1 package cscd212tests.lab6;
2
3 import static org.junit.jupiter.api.Assertions.*;
4
5 public class CSDC212Lab6CellTests {
6
7     @Test
8     void testInit() {
9         Cell cell = new Cell();
10        assertNull(cell.getLifeForm());
11    }
12
13    @Test
14    public void testAddLifeForm() {
15        LifeForm bob = new LifeForm("Bob", 40);
16        LifeForm fred = new LifeForm("Fred", 40);
17        Cell cell = new Cell();
18        boolean success = cell.addLifeForm(bob);
19        assertTrue(success);
20        assertEquals(bob, cell.getLifeForm());
21        success = cell.addLifeForm(fred);
22        assertFalse(success);
23        assertEquals(bob, cell.getLifeForm());
24    }
25
26 }
27
28
29
30
31
32
33
34
35
36
37
38
39
40

```

Group 1 Screenshots

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

```

1 package cscd212classes.lab6;
2
3 public class Cell {
4     private LifeForm entity;
5
6     //Currently, this is only returning null
7     //Returns the LifeForm in the Cell.
8     public LifeForm getLifeForm() {
9         return this.entity;
10    }
11
12    /*
13     * Add a LifeForm as a private data member, store it in class level
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        //Currently, entity is null
20        if (this.entity == null) {
21            //Set the passed in null entity to the bob entity
22            //into this method
23            /*
24             * this.entity = entity;
25            */
26            return true;
27        }
28        return false;
29    }
30
31    //Passes with just null for some reason
32    /*
33     * - Removes the LifeForm in the Cell.
34     * - Returns the LifeForm removed, null
35     * if no LifeForm in the Cell.
36     */
37    public LifeForm removeLifeForm() {
38        return null;
39    }
40

```

```

19    LifeForm bob = new LifeForm("Bob", 40);
20    LifeForm fred = new LifeForm("Fred", 40);
21    Cell cell = new Cell();
22    boolean success = cell.addLifeForm(bob);
23    assertTrue(success);
24    assertEquals(bob, cell.getLifeForm());
25    success = cell.addLifeForm(fred);
26    assertFalse(success);
27    assertEquals(bob, cell.getLifeForm());
28
29    /* Checks:
30     * -is LifeForm in the Cell
31     * -is LifeForm removed
32     * -is LifeForm returned
33     *
34     * Desc: removeLifeForm should first check if there
35     * is a LifeForm in the Cell to remove. Next, it should
36     * either return null if there is no LifeForm to remove,
37     * or it should return the LifeForm that is removed.
38     */
39    @Test
40    public void testRemoveLifeForm() {
41        LifeForm bob = new LifeForm("Bob", 40);
42        Cell cell = new Cell();
43        //Make sure the LifeForm hasn't been added yet
44        assertNull(cell.getLifeForm());
45        //Add bob to the Cell via addLifeForm
46        cell.addLifeForm(bob);
47        //Make sure the LifeForm just added is equal to bob
48        assertEquals(bob, cell.getLifeForm());
49        //Make sure the LifeForm is removed
50        assertNull(cell.getLifeForm());
51    }
52
53 }
54

```

```

1 CSDC212Lab6LifeFormTests.java x LifeForm.java Cell.java x
2
3     return this.entity;
4 }
5
6 /**
7  * Add a LifeForm as a private data member, store it in class.
8  * Adds a LifeForm to this Cell.
9  * Will not add if LifeForm is already in the Cell.
10 * Returns true if added, false otherwise.
11 */
12 public boolean addLifeForm(final LifeForm entity) {
13     //Currently, entity is null
14     if (this.entity == null) {
15         /*Set the passed in null entity to the bob entity
16          * into this method
17         */
18         this.entity = entity;
19         return true;
20     }
21     return false;
22 }
23
24 //Passes with just null for some reason
25 /**
26  * - Removes the LifeForm in the Cell.
27  * - Returns the LifeForm removed, null
28  * if no LifeForm in the Cell.
29 */
30 public LifeForm removeLifeForm() {
31     if (this.entity != null) {
32         LifeForm removedLF = this.entity;
33         this.entity = null;
34         return removedLF;
35     }
36     return null;
37 }
38
39 }
40
41 }
42
43 }
44 }

1 CSDC212Lab6CellTests.java x
2
3     LifeForm bob = new LifeForm("Bob", 40);
4     LifeForm fred = new LifeForm("Fred", 40);
5     Cell cell = new Cell();
6     boolean success = cell.addLifeForm(bob);
7     assertTrue(success);
8     assertEquals(bob, cell.getLifeForm());
9     success = cell.addLifeForm(fred);
10    assertFalse(success);
11    assertEquals(bob, cell.getLifeForm());
12
13 /**
14  * Checks:
15  * -is LifeForm in the Cell
16  * -is LifeForm removed
17  * -is LifeForm returned
18  *
19  * Desc: removeLifeForm should first check if there
20  * is a LifeForm in the Cell to remove. Next, it should
21  * either return null if there is no LifeForm to remove,
22  * or it should return the LifeForm that is removed.
23  */
24 @Test
25 public void testRemoveLifeForm() {
26     LifeForm bob = new LifeForm("Bob", 40);
27     Cell cell = new Cell();
28     //Make sure the LifeForm hasn't been added yet
29     assertNull(cell.removeLifeForm());
30     //Add bob to the Cell via addLifeForm
31     cell.addLifeForm(bob);
32     //Make sure the LifeForm just added is equal to bob
33     assertEquals(bob, cell.removeLifeForm());
34     //Make sure the LifeForm is removed
35     assertNull(cell.getLifeForm());
36 }
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

```

Group 1 Step 11 - testRemoveLifeForm (Refactored)

```

1 CSDC212Lab6LifeFormTests.java x LifeForm.java Cell.java x
2
3     return this.entity;
4 }
5
6 /**
7  * Add a LifeForm as a private data member, store it in class.
8  * Adds a LifeForm to this Cell.
9  * Will not add if LifeForm is already in the Cell.
10 * Returns true if added, false otherwise.
11 */
12 public boolean addLifeForm(final LifeForm entity) {
13     //Currently, entity is null
14     if (this.entity == null) {
15         /*Set the passed in null entity to the bob entity
16          * into this method
17         */
18         this.entity = entity;
19         return true;
20     }
21     return false;
22 }
23
24 /**
25  * - Removes the LifeForm in the Cell.
26  * - Returns the LifeForm removed, null
27  * if no LifeForm in the Cell.
28 */
29 public LifeForm removeLifeForm() {
30     if (this.entity != null) {
31         LifeForm removedLF = this.entity;
32         this.entity = null;
33         return removedLF;
34     }
35     return null;
36 }
37
38 }
39
40 }
41
42 }
43
44 }

1 CSDC212Lab6CellTests.java x
2
3     assertNotEquals(success),
4     assertEquals(bob, cell.getLifeForm());
5     success = cell.addLifeForm(fred);
6     assertFalse(success);
7     assertEquals(bob, cell.getLifeForm());
8 }
8
9 /**
10  * Checks:
11  * -is LifeForm in the Cell
12  * -is LifeForm removed
13  * -is LifeForm returned
14  *
15  * Desc: removeLifeForm should first check if there
16  * is a LifeForm in the Cell to remove. Next, it should
17  * either return null if there is no LifeForm to remove,
18  * or it should return the LifeForm that is removed.
19  */
20 @Test
21 public void testRemoveLifeFormBeforeAdding() {
22     //If there is no LF, it should be null
23     assertNull(cell.removeLifeForm());
24 }
25
26
27 /**
28  * Test when one cell is empty and one is full
29  */
30 @Test
31 public void testRemoveLifeFormAfterAdding() {
32     //Add bob to the Cell via addLifeForm
33     cell.addLifeForm(bob);
34     //Make sure the LifeForm just added is equal to bob
35     assertEquals(bob, cell.removeLifeForm());
36     //Make sure the LifeForm is removed
37     assertNull(cell.getLifeForm());
38 }
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66

```

Group 2 Screenshots

Group 2 Step 3 - Basic Environment Test (Fail)

```

1 package cscd212classes.lab6;
2
3 public class Environment {
4     //private Cell[][] cells = new Cell[2][3];
5     private Cell[][] cells;
6
7     public Environment(Cell[][] cells) {
8         for (int row = 0; row < cells.length; row++) {
9             for (int col = 0; col < cells[0].length; col++) {
10                 cells[row][col] = new Cell();
11             }
12         }
13     }
14     /*
15      * Adds a LifeForm to the Cell
16      * theCells[row][col]. Will not add the
17      * LifeForm if the row and col are
18      * invalid or if a LifeForm is already in
19      * that Cell. Returns true if
20      * successfully added, false otherwise.
21      */
22     public boolean addLifeForm(int row, int col, LifeForm entity) {
23         return false;
24     }
25
26
27     public LifeForm getLifeForm(int row, int col) {
28         return null;
29     }
30 }

```

```

13 class CSDC212Lab6EnvironmentTests {
14
15     /*@BeforeEach
16     void setUp() throws Exception {
17     }*/
18
19     /*@Test
20     void testInit() {
21         fail("Not yet implemented");
22    }*/
23
24     @Test
25     void testCreateEnvironmentOneCell() {
26         LifeForm bob = new LifeForm("Bob", 40);
27         Cell[][] cells = new Cell[2][3];
28         Environment envObj = new Environment(cells);
29
30         //Checks if addLifeForm properly adds bob to cells[1][0]
31         assertTrue(cells[1][0].addLifeForm(bob));
32         //Checks if the LF just put into cells[1][0] is equivalent to bob
33         assertEquals(bob, envObj.getLifeForm(1, 0));
34     }
35
36     //assertTrue for successful addition
37     //assertFalse for unsuccessful addition
38     //assertEquals for
39     @Test
40     void testAddLifeForm() {
41         LifeForm bob = new LifeForm("Bob", 40);
42         LifeForm fred = new LifeForm("Fred", 40);
43         //Cell[][] cells = new Cell[2][3];
44
45         //For loop fills cells with Cell() objects
46         /*
47             for (int row = 0; row < cells.length; row++) {

```

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

```

1 package cscd212classes.lab6;
2
3 public class Environment {
4     //private Cell[][] cells = new Cell[2][3];
5     private Cell[][] cells;
6
7     public Environment(Cell[][] cells) {
8         for (int row = 0; row < cells.length; row++) {
9             for (int col = 0; col < cells[0].length; col++) {
10                 cells[row][col] = new Cell();
11             }
12         }
13     }
14     /*
15      * Adds a LifeForm to the Cell
16      * theCells[row][col]. Will not add the
17      * LifeForm if the row and col are
18      * invalid or if a LifeForm is already in
19      * that Cell. Returns true if
20      * successfully added, false otherwise.
21      */
22     public boolean addLifeForm(int row, int col, LifeForm entity) {
23         return false;
24     }
25
26
27     public LifeForm getLifeForm(int row, int col) {
28         return this.cells[row][col].getLifeForm();
29     }
30 }

```

```

13 class CSDC212Lab6EnvironmentTests {
14
15     /*@BeforeEach
16     void setUp() throws Exception {
17     }*/
18
19     /*@Test
20     void testInit() {
21         fail("Not yet implemented");
22    }*/
23
24     @Test
25     void testCreateEnvironmentOneCell() {
26         LifeForm bob = new LifeForm("Bob", 40);
27         Cell[][] cells = new Cell[2][3];
28         Environment envObj = new Environment(cells);
29
30         //Checks if addLifeForm properly adds bob to cells[1][0]
31         assertTrue(cells[1][0].addLifeForm(bob));
32         //Checks if the LF just put into cells[1][0] is equivalent to bob
33         assertEquals(bob, envObj.getLifeForm(1, 0));
34     }
35
36     //assertTrue for successful addition
37     //assertFalse for unsuccessful addition
38     //assertEquals for
39     @Test
40     void testAddLifeForm() {
41         LifeForm bob = new LifeForm("Bob", 40);
42         LifeForm fred = new LifeForm("Fred", 40);
43         //Cell[][] cells = new Cell[2][3];
44
45         //For loop fills cells with Cell() objects
46         /*
47             for (int row = 0; row < cells.length; row++) {

```

Group 2 Step 6 - testAddLifeForm and testRemoveLifeForm methods

testAddLifeForm:

```
1 package cscd212classes.lab6;
2
3 public class Environment {
4     //private Cell[][][] cells = new Cell[2][3];
5     private Cell[][][] cells;
6
7     public Environment(Cell[][][] cells) {
8         for (int row = 0; row < cells.length; row++) {
9             for (int col = 0; col < cells[0].length; col++) {
10                 cells[row][col] = new Cell();
11             }
12         }
13     this.cells = cells;
14 }
15 /**
16 * Adds a LifeForm to the Cell
17 * theCells[row][col]. Will not add the
18 * LifeForm if the row and col are
19 * invalid or if a LifeForm is already in
20 * that Cell. Returns true if
21 * successfully added, false otherwise.
22 */
23 public boolean addLifeForm(int row, int col, LifeForm entity)
24     return false;
25 }
26
27
28 public LifeForm getLifeForm(int row, int col) {
29     return this.cells[row][col].getLifeForm();
30 }
31 }
```

```
1 package cscd212classes.lab6;
2
3 public class Environment {
4     //public Cell[][] cells = new Cell[2][3];
5     private Cell[][] cells;
6
7     public Environment(Cell[][] cells) {
8         for (int row = 0; row < cells.length; row++) {
9             for (int col = 0; col < cells[0].length; col++) {
10                 cells[row][col] = new Cell();
11             }
12         }
13         this.cells = cells;
14     }
15     /*
16      * Adds a LifeForm to the Cell
17      * theCells[row][col]. Will not add the
18      * Lifeform if the row and col are
19      * invalid or if a Lifeform is already in
20      * that Cell. Returns true if
21      * successfully added, false otherwise.
22      */
23     public boolean addLifeForm(int row, int col, LifeForm entity) {
24         if (entity != null) {
25             cells[row][col].addLifeForm(entity);
26             return true;
27         }
28         return false;
29     }
30
31
32     public LifeForm getLifeForm(int row, int col) {
33         return this.cells[row][col].getLifeForm();
34     }
35 }
```

```
1 package cscd212tests.lab6;
2
3 import static org.junit.jupiter.api.Assertions.*;
4
5 class CSCD212Lab6EnvironmentTests {
6
7     /*@BeforeEach
8     void setUp() throws Exception {
9     }*/
10
11     /*@Test
12     void testInit() {
13         fail("Not yet implemented");
14     }*/
15
16     @Test
17     void testCreateEnvironmentOneCell() {
18         LifeForm bob = new LifeForm("Bob", 40);
19         Cell[][] cells = new Cell[2][3];
20         Environment envObj = new Environment(cells);
21
22         //Checks if addLifeForm properly adds bob to cells[1][0]
23         assertTrue(cells[1][0].addLifeForm(bob));
24         //Checks if the LF just put into cells[1][0] is equivalent to bob
25         assertEquals(bob, envObj.getLifeForm(1, 0));
26     }
27
28     @Test
29     void testAddLifeForm() {
30         LifeForm bob = new LifeForm("Bob", 40);
31         LifeForm fred = null;
32         Cell[][] cells = new Cell[2][3];
33         Environment envObj = new Environment(cells);
34
35         assertFalse(envObj.addLifeForm(0, 1, fred));
36
37         assertTrue(envObj.addLifeForm(0, 0, bob));
38         assertEquals(envObj.getLifeForm(0, 0), bob);
39     }
40 }
```

testRemoveLifeForm:

The image shows two side-by-side Java code editors. The left editor contains the `Environment.java` class, which defines a 2D array of `Cell` objects. It includes methods for adding and removing `LifeForm` entities from specific cells. The right editor contains the `CSDC212Lab6LifeFormTest.java` test class, which contains three test methods: `testCreateEnvironmentOneCell()`, `testAddLifeForm()`, and `testRemoveLifeForm()`. These tests verify the functionality of the `Environment` class by creating a new environment, adding a life form, and then removing it.

```

1 package cscd212classes.lab6;
2
3 public class Environment {
4     //private Cell[][] cells = new Cell[2][3];
5     private Cell[][] cells;
6
7     public Environment(Cell[][] cells) {
8         for (int row = 0; row < cells.length; row++) {
9             for (int col = 0; col < cells[0].length; col++) {
10                 cells[row][col] = new Cell();
11             }
12         }
13         this.cells = cells;
14     }
15
16     public boolean addLifeForm(int row, int col, LifeForm entity) {
17         if (entity != null) {
18             cells[row][col].addLifeForm(entity);
19             return true;
20         }
21         return false;
22     }
23
24     public LifeForm getLifeForm(int row, int col) {
25         return this.cells[row][col].getLifeForm();
26     }
27
28     public LifeForm removeLifeForm(int row, int col) {
29         return null;
30     }
31 }

```

```

1 @Test
2 void testCreateEnvironmentOneCell() {
3     LifeForm bob = new LifeForm("Bob", 40);
4     Cell[][] cells = new Cell[2][3];
5     Environment envObj = new Environment(cells);
6
7     //Checks if addLifeForm properly adds bob to cells[1][0]
8     assertTrue(cells[1][0].addLifeForm(bob));
9     //Checks if the LF just put into cells[1][0] is equivalent to bob
10    assertEquals(bob, envObj.getLifeForm(1, 0));
11 }
12
13 @Test
14 void testAddLifeForm() {
15     LifeForm bob = new LifeForm("Bob", 40);
16     LifeForm fred = null;
17     Cell[][] cells = new Cell[2][3];
18     Environment envObj = new Environment(cells);
19
20     assertFalse(envObj.addLifeForm(0, 1, fred));
21
22     assertTrue(envObj.addLifeForm(0, 0, bob));
23     assertEquals(envObj.getLifeForm(0, 0), bob);
24 }
25
26 @Test
27 void testRemoveLifeForm() {
28     //Initialize new LifeForm obj
29     LifeForm bob = new LifeForm("Bob", 40);
30     //Initialize new Cell array
31     Cell[][] cells = new Cell[2][3];
32     /*Create new Environment object that takes the new Cell
33      * array as its parameter
34      */
35     Environment envObj = new Environment(cells);
36
37     //Add the bob LF to [0][0] of our new array
38     envObj.addLifeForm(0, 0, bob);
39     //Check if the LF removed matches that of bob (which we just added)
40     assertEquals(envObj.removeLifeForm(0, 0), bob);
41     //Check if the LF was actually deleted
42     assertNull(envObj.getLifeForm(0, 0));
43     //Check if removeLifeForm returns null appropriately
44     assertNull(envObj.removeLifeForm(0, 0));
45 }
46
47 }

```



```

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

```

```

1 @Test
2 void testCreateEnvironmentOneCell() {
3     LifeForm bob = new LifeForm("Bob", 40);
4     Cell[][] cells = new Cell[2][3];
5     Environment envObj = new Environment(cells);
6
7     //Checks if addLifeForm properly adds bob to cells[1][0]
8     assertTrue(cells[1][0].addLifeForm(bob));
9     //Checks if the LF just put into cells[1][0] is equivalent to bob
10    assertEquals(bob, envObj.getLifeForm(1, 0));
11 }
12
13 @Test
14 void testAddLifeForm() {
15     LifeForm bob = new LifeForm("Bob", 40);
16     LifeForm fred = null;
17     Cell[][] cells = new Cell[2][3];
18     Environment envObj = new Environment(cells);
19
20     assertFalse(envObj.addLifeForm(0, 1, fred));
21
22     assertTrue(envObj.addLifeForm(0, 0, bob));
23     assertEquals(envObj.getLifeForm(0, 0), bob);
24 }
25
26 @Test
27 void testRemoveLifeForm() {
28     //Initialize new LifeForm obj
29     LifeForm bob = new LifeForm("Bob", 40);
30     //Initialize new Cell array
31     Cell[][] cells = new Cell[2][3];
32     /*Create new Environment object that takes the new Cell
33      * array as its parameter
34      */
35     Environment envObj = new Environment(cells);
36
37     //Add the bob LF to [0][0] of our new array
38     envObj.addLifeForm(0, 0, bob);
39     //Check if the LF removed matches that of bob (which we just added)
40     assertEquals(envObj.removeLifeForm(0, 0), bob);
41     //Check if the LF was actually deleted
42     assertNull(envObj.getLifeForm(0, 0));
43     //Check if removeLifeForm returns null appropriately
44     assertNull(envObj.removeLifeForm(0, 0));
45 }
46
47 }

```

Group 3 Screenshots

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

```

1 LifeForm.java   2 Cell.java    3 Environment.java
4
5 public class Environment {
6     //private Cell[][] cells = new Cell[2][3];
7     private Cell[][] cells;
8
9     public Environment(Cell[][] cells) {
10         for (int row = 0; row < cells.length; row++) {
11             for (int col = 0; col < cells[0].length; col++) {
12                 cells[row][col] = new Cell();
13             }
14         }
15     }
16     public boolean addLifeForm(int row, int col, LifeForm entity) {
17         if (entity != null) {
18             cells[row][col].addLifeForm(entity);
19             return true;
20         }
21         return false;
22     }
23
24     public LifeForm getLifeForm(int row, int col) {
25         /*
26         if(this.cells[row][col] == null) return null;
27         return this.cells[row][col].getLifeForm();
28         */
29         return null;
30     }
31
32     public LifeForm removeLifeForm(int row, int col) {
33         if (this.cells[row][col] != null) {
34             LifeForm tempCells = this.cells[row][col].getLifeForm();
35             this.cells[row][col] = null;
36             return tempCells;
37         }
38     }
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67

```

```

1 CSDC212Lab6C... 2 CSDC212Lab6Li... 3 CSDC212Lab6E... 4 CSDC212Lab6...
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67

```

Group 3 Step 2 - getLifeForm Passes

```

1 LifeForm.java   2 Cell.java    3 Environment.java
4
5 package cscd212classes.lab6;
6
7 public class Environment {
8     //private Cell[][] cells = new Cell[2][3];
9     private Cell[][] cells;
10
11    public Environment(Cell[][] cells) {
12        for (int row = 0; row < cells.length; row++) {
13            for (int col = 0; col < cells[0].length; col++) {
14                cells[row][col] = new Cell();
15            }
16        }
17        this.cells = cells;
18    }
19
20    public boolean addLifeForm(int row, int col, LifeForm entity) {
21        if (entity != null) {
22            cells[row][col].addLifeForm(entity);
23            return true;
24        }
25        return false;
26    }
27
28    public LifeForm getLifeForm(int row, int col) {
29        if(this.cells[row][col] == null) return null;
30        return this.cells[row][col].getLifeForm();
31    }
32
33    public LifeForm removeLifeForm(int row, int col) {
34        if (this.cells[row][col] != null) {
35            LifeForm tempCells = this.cells[row][col].getLifeForm();
36            this.cells[row][col] = null;
37            return tempCells;
38        }
39    }
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67

```

```

1 CSDC212Lab6C... 2 CSDC212Lab6Li... 3 CSDC212Lab6E... 4 CSDC212Lab6...
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67

```

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



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



```
1 CSDC212Lab6Cell...
2
3     /**
4      * @Test
5      * void testInit() {
6      * }
6
7     @Test
8     void testCreateEnvironmentOneCell() {
9         //Checks if addLifeForm properly adds bob to cells[1][0]
10        assertEquals(cells[1][0].addLifeForm(bob));
11        //Checks if the LF just put into cells[1][0] is equivalent to bob
12        assertEquals(bob, envObj.getLifeForm(1, 0));
13    }
14
15    @Test
16    void testAddLifeForm() {
17        LifeForm fred = null;
18
19        assertFalse(envObj.addLifeForm(0, 1, fred));
20
21        assertTrue(envObj.addLifeForm(0, 0, bob));
22        assertEquals(envObj.getLifeForm(0, 0), bob);
23    }
24
25    @Test
26    void testRemoveLifeForm() {
27        //Add the bob LF to [0][0] of our new array
28        envObj.addLifeForm(0, 0, bob);
29        //Check if the LF removed matches that of bob (which we just added)
30        assertEquals(envObj.removeLifeForm(0, 0), bob);
31        //Check if the LF was actually deleted
32        assertNull(envObj.getLifeForm(0, 0));
33        //Check if removeLifeForm returns null appropriately
34        assertNull(envObj.removeLifeForm(0, 0));
35    }
36
37    @Test
38    void testGetLifeForm() {
39        envObj.addLifeForm(0, 0, bob);
40        assertEquals(envObj.getLifeForm(0, 0), bob);
41
42        LifeForm jimmy = new LifeForm("Jimmy", 40);
43
44        envObj.addLifeForm(1, 2, bob);
45        assertEquals(envObj.getLifeForm(1, 2), bob);
46    }
47
48 }
```

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)

The screenshot shows a Java development environment with two tabs open:

- Environment.java** (Left Tab):


```

1 package cscd212classes.lab6;
2
3 public class Environment {
4     //private Cell[][] cells = new Cell[2][3];
5     private Cell[][] cells;
6
7     public Environment(Cell[][] cells) {
8         for (int row = 0; row < cells.length; row++) {
9             for (int col = 0; col < cells[0].length; col++) {
10                 cells[row][col] = new Cell();
11             }
12         }
13         this.cells = cells;
14     }
15
16     public boolean addLifeForm(int row, int col, LifeForm entity) {
17         if (entity != null) {
18             cells[row][col].addLifeForm(entity);
19             return true;
20         }
21         return false;
22     }
23
24     public LifeForm getLifeForm(int row, int col) {
25         if(this.cells[row][col] == null) return null;
26         return this.cells[row][col].getLifeForm();
27     }
28
29     public LifeForm removeLifeForm(int row, int col) {
30         if (this.cells[row][col] != null) {
31             LifeForm tempCells = this.cells[row][col].getLifeForm();
32             this.cells[row][col] = null;
33             return tempCells;
34         }
35         return null;
36     }
37 }
38 
```
- CSDC212Lab6Test.java** (Right Tab):


```

1 CSDC212Lab6Cell... 2 CSDC212Lab6Life... 3 CSDC212Lab6Envi... 4 CSDC212Lab6Test...
40     void testAddLifeForm() {
41         LifeForm fred = null;
42
43         assertFalse(envObj.addLifeForm(0, 1, fred));
44
45         assertTrue(envObj.addLifeForm(0, 0, bob));
46         assertEquals(envObj.getLifeForm(0, 0), bob);
47     }
48
49     @Test
50     void testRemoveLifeForm() {
51         //Add the bob LF to [0][0] of our new array
52         envObj.addLifeForm(0, 0, bob);
53         //Check if the LF removed matches that of bob (which we just added)
54         assertEquals(envObj.removeLifeForm(0, 0), bob);
55         //Check if the LF was actually deleted
56         assertNull(envObj.getLifeForm(0, 0));
57         //Check if removeLifeForm returns null appropriately
58         assertNull(envObj.removeLifeForm(0, 0));
59     }
60
61     @Test
62     void testGetLifeForm() {
63         envObj.addLifeForm(0, 0, bob);
64         assertEquals(envObj.getLifeForm(0, 0), bob);
65
66         LifeForm jimmy = new LifeForm("Jimmy", 40);
67
68         envObj.addLifeForm(1, 2, bob);
69         assertEquals(envObj.getLifeForm(1, 2), bob);
70         envObj.removeLifeForm(1, 2);
71         envObj.addLifeForm(1, 2, jimmy);
72         assertEquals(envObj.getLifeForm(1, 2), jimmy);
73     }
74 }
75 
```

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

```

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

1 CSDC212Lab6CellT...   2 CSDC212Lab6LifeF...   3 CSDC212Lab6Envir...
1 void testCreateEnvi...
2 */
3
4 @Test
5 void testCreateEnvironmentOneCell() {
6     //Checks if addLifeForm properly adds bob to cells[1][0]
7     assertTrue(cells[1][0].addLifeForm(bob));
8     //Checks if the LF just put into cells[1][0] is equivalent to
9     assertEquals(bob, envObj.getLifeForm(1, 0));
10 }
11
12 @Test
13 void testAddLifeForm() {
14     LifeForm fred = null;
15
16     assertFalse(envObj.addLifeForm(0, 1, fred));
17
18     assertTrue(envObj.addLifeForm(0, 0, bob));
19     assertEquals(envObj.getLifeForm(0, 0), bob);
20 }
21
22 @Test
23 void testRemoveLifeForm() {
24     //Add the bob LF to [0][0] of our new array
25     envObj.addLifeForm(0, 0, bob);
26     //Check if the LF removed matches that of bob (which we just a
27     assertEquals(envObj.removeLifeForm(0, 0), bob);
28     //Check if the LF was actually deleted
29     assertNull(envObj.getLifeForm(0, 0));
30     //Check if removeLifeForm returns null appropriately
31     assertNull(envObj.removeLifeForm(0, 0));
32 }
33
34 @Test
35 void testGetLifeForm() {
36     envObj.addLifeForm(0, 0, bob);
37     assertEquals(envObj.getLifeForm(0, 0), bob);
38
39     LifeForm jimmy = new LifeForm("Jimmy", 40);
40
41     envObj.addLifeForm(1, 2, bob);
42     assertEquals(envObj.getLifeForm(1, 2), bob);
43     envObj.removeLifeForm(1, 2);
44     envObj.addLifeForm(1, 2, jimmy);
45     assertEquals(envObj.getLifeForm(1, 2), jimmy);
46 }
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75

```

```

1 CSDC212Lab6Tests.java ×
1 package cscd212tests.lab6;
2
3 import org.junit.platform.runner.JUnitPlatform;
4 import org.junit.platform.suite.api.SelectPackages;
5 import org.junit.runner.RunWith;
6
7 import org.junit.platform.suite.api.*;
8
9 @RunWith(JUnitPlatform.class)
10 @SelectPackages("cscd212tests.lab6")
11 @SelectClasses({CSDC212Lab6EnvironmentTests.class, CSDC212Lab6CellT
12 public class CSDC212Lab6Tests {}
```

Finished after 0.139 seconds

Runs: ✘ Errors: ✘ Failures:

> cscd212tests.lab6.CSC

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:

```
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         this.name = name;
9         this.currentLifePoints = currentLifePoints;
10    }
11
12    public String getName() {
13        return this.name;
14    }
15
16    public int getCurrentLifePoints() {
17        return this.currentLifePoints;
18    }
19 }
20
```

```
13     private LifeForm entity;
14
15     @BeforeEach
16     public void setup() throws Exception {
17         entity = new LifeForm("Bob", 40);
18     }
19
20
21     @Test
22     public void testInit() {
23         LifeForm entity;
24         entity = new LifeForm("Bob", 40);
25         assertEquals("Bob", entity.getName());
26     }
27
28
29     @Test
30     public void testName() {
31     }
32         assertEquals("Bob", entity.getName());
33     }
34
35     @Test
36     public void testGetCurrentLifePoints() {
37     }
38         assertEquals(40, entity.getCurrentLifePoints());
39     }
40
41     @Test
42     public void testNameIsNotNull() {
43         assertThrows(IllegalArgumentException.class, ()-> {
44             new LifeForm(null, 40);
45         });
46     }
47 }
```

```
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) throw new IllegalArgumentException("Bad Params in LifeForm Cons");
9         this.name = name;
10        this.currentLifePoints = currentLifePoints;
11    }
12
13    public String getName() {
14        return this.name;
15    }
16
17    public int getCurrentLifePoints() {
18        return this.currentLifePoints;
19    }
20 }
21
```

```
13     private LifeForm entity;
14
15     @BeforeEach
16     public void setup() throws Exception {
17         entity = new LifeForm("Bob", 40);
18     }
19
20
21     @Test
22     public void testInit() {
23         LifeForm entity;
24         entity = new LifeForm("Bob", 40);
25         assertEquals("Bob", entity.getName());
26     }
27
28
29     @Test
30     public void testName() {
31     }
32         assertEquals("Bob", entity.getName());
33     }
34
35     @Test
36     public void testGetCurrentLifePoints() {
37     }
38         assertEquals(40, entity.getCurrentLifePoints());
39     }
40
41     @Test
42     public void testNameIsNotNull() {
43         assertThrows(IllegalArgumentException.class, ()-> {
44             new LifeForm(null, 40);
45         });
46     }
47 }
```

testNameIsEmpty:

```
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 IllegalArgumentException("Bad Params in LifeForm Constructor");
9         this.name = name;
10        this.currentLifePoints = currentLifePoints;
11    }
12
13    public String getName() {
14        return this.name;
15    }
16
17    public int getCurrentLifePoints() {
18        return this.currentLifePoints;
19    }
20 }
```

```
8 import cscd212classes.lab6.LifeForm;
9
10 public class CSDC212Lab6LifeFormTests {
11     private LifeForm entity;
12
13     @BeforeEach
14     public void setup() throws Exception {
15         entity = new LifeForm("Bob", 40);
16     }
17
18     @Test
19     public void testInit() {
20         LifeForm entity;
21         entity = new LifeForm("Bob", 40);
22         assertEquals("Bob", entity.getName());
23     }
24
25     @Test
26     public void testGetName() {
27         assertEquals("Bob", entity.getName());
28     }
29
30     @Test
31     public void testGetCurrentLifePoints() {
32         assertEquals(40, entity.getCurrentLifePoints());
33     }
34
35     @Test
36     public void testNameIsNull() {
37         assertThrows(IllegalArgumentException.class, ()-> {
38             new LifeForm(null, 40);
39         });
40     }
41
42     @Test
43     public void testNameIsEmpty() {
44         assertThrows(IllegalArgumentException.class, ()-> {
45             new LifeForm("", 40);
46         });
47     }
48
49     @Test
50     public void testNameIsEmpty() {
51         assertThrows(IllegalArgumentException.class, ()-> {
52             new LifeForm("", 40);
53         });
54 }
```

```
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 IllegalArgumentException("Bad Params in LifeForm Constructor");
9         this.name = name;
10        this.currentLifePoints = currentLifePoints;
11    }
12
13    public String getName() {
14        return this.name;
15    }
16
17    public int getCurrentLifePoints() {
18        return this.currentLifePoints;
19    }
20 }
```

```
8 import cscd212classes.lab6.LifeForm;
9
10 public class CSDC212Lab6LifeFormTests {
11     private LifeForm entity;
12
13     @BeforeEach
14     public void setup() throws Exception {
15         entity = new LifeForm("Bob", 40);
16     }
17
18     @Test
19     public void testInit() {
20         LifeForm entity;
21         entity = new LifeForm("Bob", 40);
22         assertEquals("Bob", entity.getName());
23     }
24
25     @Test
26     public void testGetName() {
27         assertEquals("Bob", entity.getName());
28     }
29
30     @Test
31     public void testGetCurrentLifePoints() {
32         assertEquals(40, entity.getCurrentLifePoints());
33     }
34
35     @Test
36     public void testNameIsNull() {
37         assertThrows(IllegalArgumentException.class, ()-> {
38             new LifeForm(null, 40);
39         });
40     }
41
42     @Test
43     public void testNameIsEmpty() {
44         assertThrows(IllegalArgumentException.class, ()-> {
45             new LifeForm("", 40);
46         });
47     }
48
49     @Test
50     public void testNameIsEmpty() {
51         assertThrows(IllegalArgumentException.class, ()-> {
52             new LifeForm("", 40);
53         });
54 }
```

testCurrentLifePoints < 0:

```
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 IllegalArgumentException();
9         this.name = name;
10        this.currentLifePoints = currentLifePoints;
11    }
12
13    public String getName() {
14        return this.name;
15    }
16
17    public int getCurrentLifePoints() {
18        return this.currentLifePoints;
19    }
20 }
```

```
20
21
22
23
24
25
26
27
28
29● @Test
30     public void testGetName() {
31     {
32         assertEquals("Bob", entity.getName());
33     }
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 }
```

```
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 IllegalArgumentException();
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 }
```

```
20
21
22
23
24
25
26
27
28
29● @Test
30     public void testGetName() {
31     {
32         assertEquals("Bob", entity.getName());
33     }
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 }
```

testCurrentLifePoints == 0:

The screenshot shows two code editors side-by-side. The left editor contains `LifeForm.java` with the following code:

```
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 IllegalArgumentException();
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 getCurrentLifePoint() {
19        return this.currentLifePoints;
20    }
21 }
22
```

The right editor contains `CSDC212Lab6CellTest.java` with the following test cases:

```
1 CSDC212Lab6Cell...
2 CSDC212Lab6Life...
3 CSDC212Lab6Test...
4 CSDC212Lab6Envi...
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
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:

```
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     }
10    //return this.entity;
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 }
```

```
1 CSCD212Lab6Cell... x CSCD212Lab6Life... x CSCD212Lab6Test... x CSCD212Lab6Envi...
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
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
```

```

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 this.entity;
9     }
10
11     /*
12      * Add a LifeForm as a private data member, store it in class level variable
13      * Adds a LifeForm to this Cell.
14      * Will not add if LifeForm is already in the Cell.
15      * Returns true if added, false otherwise.
16      */
17     public boolean addLifeForm(final LifeForm entity) {
18         if (this.entity == null) {
19             /*Set the passed in null entity to the bob entity passed
20              * into this method
21             */
22             this.entity = entity;
23             return true;
24         }
25         return false;
26     }
27
28     /*
29      * - Removes the LifeForm in the Cell.
30      * - Returns the LifeForm removed, null
31      * if no LifeForm in the Cell.
32      */
33     public LifeForm removeLifeForm() {
34         if (this.entity != null) {
35             LifeForm removedLF = this.entity;
36             /*Set the removed entity to null
37             */
38             this.entity = null;
39             return removedLF;
40         }
41     }
42
43     /*
44      * - Returns the LifeForm removed, null
45      * if no LifeForm in the Cell.
46      */
47     public LifeForm removeLifeForm() {
48         if (this.entity != null) {
49             LifeForm removedLF = this.entity;
50             /*Set the removed entity to null
51             */
52             this.entity = null;
53             return removedLF;
54         }
55     }
56 }

```

```

1 CSDC212Lab6Cell... X CSDC212Lab6Life... CSDC212Lab6Test... CSDC212Lab6Env...
2
3     assertEquals(success);
4     assertEquals(bob, cell.getLifeForm());
5 }
6
7 @Test
8 public void testAddLifeFormOccupiedCell() {
9     cell.addLifeForm(bob);
10    boolean success = cell.addLifeForm(fred);
11    assertFalse(success);
12    assertEquals(bob, cell.getLifeForm());
13 }
14
15 //Test when one cell is empty and one is full
16 @Test
17 public void testRemoveLifeFormBeforeAdding() {
18     //If there is no LF, it should be null
19     assertNull(cell.removeLifeForm());
20 }
21
22 @Test
23 public void testRemoveLifeFormAfterAdding() {
24     //Add bob to the Cell via addLifeForm
25     cell.addLifeForm(bob);
26     //Make sure the LifeForm just added is equal to bob
27     assertEquals(bob, cell.removeLifeForm());
28     //Make sure the LifeForm is removed
29     assertNull(cell.getLifeForm());
30 }
31
32 @Test
33 public void testGetLifeForm() {
34     cell.addLifeForm(bob);
35     assertEquals(cell.getLifeForm(), bob);
36 }
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72

```

testAddLifeFormReceivesNotNull: *passed right away

```

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 this.entity;
9     }
10
11     /*
12      * Add a LifeForm as a private data member, store it in class level variable (static)
13      * Adds a LifeForm to this Cell.
14      * Will not add if LifeForm is already in the Cell.
15      * Returns true if added, false otherwise.
16      */
17     public boolean addLifeForm(final LifeForm entity) {
18         if (this.entity == null) {
19             /*Set the passed in null entity to the bob entity passed
20              * into this method
21             */
22             this.entity = entity;
23             return true;
24         }
25         return false;
26     }
27
28     /*
29      * - Removes the LifeForm in the Cell.
30      * - Returns the LifeForm removed, null
31      * if no LifeForm in the Cell.
32      */
33     public LifeForm removeLifeForm() {
34         if (this.entity != null) {
35             LifeForm removedLF = this.entity;
36             this.entity = null;
37             return removedLF;
38         }
39         return null;
40     }
41 }
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72

```

```

1 CSDC212Lab6Cell... X CSDC212Lab6Life... CSDC212Lab6Test... CSDC212Lab6Env...
2
3     }
4
5     @Test
6     public void testAddLifeFormEmptyCell() {
7         boolean success = cell.addLifeForm(bob);
8         assertTrue(success);
9         assertEquals(bob, cell.getLifeForm());
10    }
11
12    @Test
13    public void testAddLifeFormOccupiedCell() {
14        cell.addLifeForm(bob);
15        boolean success = cell.addLifeForm(fred);
16        assertFalse(success);
17        assertEquals(bob, cell.getLifeForm());
18    }
19
20    //Test when one cell is empty and one is full
21    @Test
22    public void testRemoveLifeFormBeforeAdding() {
23        //If there is no LF, it should be null
24        assertNull(cell.removeLifeForm());
25    }
26
27    @Test
28    public void testRemoveLifeFormAfterAdding() {
29        //Add bob to the Cell via addLifeForm
30        cell.addLifeForm(bob);
31        //Make sure the LifeForm just added is equal to bob
32        assertEquals(bob, cell.removeLifeForm());
33        //Make sure the LifeForm is removed
34        assertNull(cell.getLifeForm());
35    }
36
37    @Test
38    public void testGetLifeForm() {
39        cell.addLifeForm(bob);
40        assertEquals(cell.getLifeForm(), bob);
41    }
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72

```

testRemoveLifeFormReceivesNull: *passed right away

```

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

```

```

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     @Test
67     public void testAddLifeFormReceivesNotNull() {
68         cell.addLifeForm(bob);
69         assertFalse(cell.addLifeForm(bob));
70     }
71
72     @Test
73     public void testRemoveLifeFormReceivesNull() {
74         LifeForm sadBob = null;
75         cell.addLifeForm(sadBob);
76         assertNull(cell.removeLifeForm());
77     }
78 }

```

***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

```

1 package cscd212classes.lab6;
2
3 public class Environment {
4     private Cell[][] cells;
5
6     public Environment(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(int row, int col, LifeForm entity) {
16        if (entity != null) {
17            cells[row][col].addLifeForm(entity);
18            return true;
19        }
20        return false;
21    }
22
23    public LifeForm getLifeForm(int row, int col) {
24        if (this.cells[row][col] == null) {
25            return null;
26        }
27        return this.cells[row][col].getLifeForm();
28    }
29
30    public LifeForm removeLifeForm(int row, int col) {
31        if (this.cells[row][col] != null) {
32            LifeForm tempCells = this.cells[row][col].getLifeForm();
33            this.cells[row][col].removeLifeForm();
34            return tempCells;
35        }
36        return null;
37    }

```

```

35     @Test
36     void testAddLifeForm() {
37         LifeForm fred = null;
38
39         assertFalse(envObj.addLifeForm(0, 1, fred));
40
41         assertTrue(envObj.addLifeForm(0, 0, bob));
42         assertEquals(envObj.getLifeForm(0, 0), bob);
43     }
44
45     @Test
46     void testRemoveLifeForm() {
47         // Add the bob LF to [0][0] of our new array
48         envObj.addLifeForm(0, 0, bob);
49         // Check if the LF removed matches that of bob (which we just added)
50         assertEquals(envObj.removeLifeForm(0, 0), bob);
51         // Check if the LF was actually deleted
52         assertEquals(null, envObj.getLifeForm(0, 0));
53         // Check if removeLifeForm returns null appropriately
54         assertEquals(null, envObj.removeLifeForm(0, 0));
55     }
56
57     @Test
58     void testGetLifeFormByAddingObject() {
59         envObj.addLifeForm(0, 0, bob);
60         assertEquals(envObj.getLifeForm(0, 0), bob);
61     }
62
63     @Test
64     void testGetLifeFormWhenALifeFormIsFillingTheCells() {
65         LifeForm jimmy = new LifeForm("jimmy", 40);
66
67         envObj.addLifeForm(1, 2, bob);
68         assertEquals(envObj.getLifeForm(1, 2), bob);
69         envObj.removeLifeForm(1, 2);
70         envObj.addLifeForm(1, 2, jimmy);
71         assertEquals(envObj.getLifeForm(1, 2), jimmy);
72     }
73
74     @Test
75     void testEnvironmentWhenCellsAreNull() {
76         assertThrows(NullPointerException.class, () -> {
77             new Environment(null);
78         });
79     }
80 }

```

testAddLifeFormWhenRowsAreNegative:

```

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 (entity != null) {
17            cells[row][col].addLifeForm(entity);
18            return true;
19        }
20        return false;
21    }
22
23    public LifeForm getLifeForm(final int row, final int col) {
24        if (this.cells[row][col] == null) return null;
25        return this.cells[row][col].getLifeForm();
26    }
27
28    public LifeForm removeLifeForm(final int row, final int col) {
29        if (this.cells[row][col] != null) {
30            LifeForm tempCells = this.cells[row][col].getLifeForm();
31            this.cells[row][col].removeLifeForm();
32            return tempCells;
33        }
34        return null;
35    }
36}
37

```

```

45    assertTrue(envObj.addLifeForm(0, 0, bob));
46    assertEquals(envObj.getLifeForm(0, 0), bob);
47}
48
49@Test
50 void testRemoveLifeForm() {
51    //Add the bob LF to [0][0] of our new array
52    envObj.addLifeForm(0, 0, bob);
53    //Check if the LF removed matches that of bob (which we just added)
54    assertEquals(envObj.removeLifeForm(0, 0), bob);
55    //Check if the LF was actually deleted
56    assertNull(envObj.getLifeForm(0, 0));
57    //Check if removeLifeForm returns null appropriately
58    assertNull(envObj.removeLifeForm(0, 0));
59}
60
61@Test
62 void testGetLifeFormByAddingObject() {
63    envObj.addLifeForm(0, 0, bob);
64    assertEquals(envObj.getLifeForm(0, 0), bob);
65}
66
67@Test
68 void testGetLifeFormWhenALifeFormIsFillingTheCells() {
69    LifeForm jimmy = new LifeForm("Jimmy", 40);
70
71    envObj.addLifeForm(1, 2, bob);
72    assertEquals(envObj.getLifeForm(1, 2), bob);
73    envObj.removeLifeForm(1, 2);
74    envObj.addLifeForm(1, 2, jimmy);
75    assertEquals(envObj.getLifeForm(1, 2), jimmy);
76}
77
78@Test
79 void testEnvironmentWhenCellsAreNull() {
80    assertThrows(NullPointerException.class, ()-> {
81        new Environment(null);
82    });
83}
84
85@Test
86 void testAddLifeFormWhenRowsAreNegative() {
87    assertThrows(IllegalArgumentException.class, ()->
88        envObj.addLifeForm(-1, 0, bob));
89}
90
91

```

```

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 (this.cells[row][col] == null) return null;
26        return this.cells[row][col].getLifeForm();
27    }
28
29    public LifeForm removeLifeForm(final int row, final int col) {
30        if (this.cells[row][col] != null) {
31            LifeForm tempCells = this.cells[row][col].getLifeForm();
32            this.cells[row][col].removeLifeForm();
33            return tempCells;
34        }
35        return null;
36    }
37}
38

```

```

45    assertTrue(envObj.addLifeForm(0, 0, bob));
46    assertEquals(envObj.getLifeForm(0, 0), bob);
47}
48
49@Test
50 void testRemoveLifeForm() {
51    //Add the bob LF to [0][0] of our new array
52    envObj.addLifeForm(0, 0, bob);
53    //Check if the LF removed matches that of bob (which we just added)
54    assertEquals(envObj.removeLifeForm(0, 0), bob);
55    //Check if the LF was actually deleted
56    assertNull(envObj.getLifeForm(0, 0));
57    //Check if removeLifeForm returns null appropriately
58    assertNull(envObj.removeLifeForm(0, 0));
59}
60
61@Test
62 void testGetLifeFormByAddingObject() {
63    envObj.addLifeForm(0, 0, bob);
64    assertEquals(envObj.getLifeForm(0, 0), bob);
65}
66
67@Test
68 void testGetLifeFormWhenALifeFormIsFillingTheCells() {
69    LifeForm jimmy = new LifeForm("Jimmy", 40);
70
71    envObj.addLifeForm(1, 2, bob);
72    assertEquals(envObj.getLifeForm(1, 2), bob);
73    envObj.removeLifeForm(1, 2);
74    envObj.addLifeForm(1, 2, jimmy);
75    assertEquals(envObj.getLifeForm(1, 2), jimmy);
76}
77
78@Test
79 void testEnvironmentWhenCellsAreNull() {
80    assertThrows(NullPointerException.class, ()-> {
81        new Environment(null);
82    });
83}
84
85@Test
86 void testAddLifeFormWhenRowsAreNegative() {
87    assertThrows(IllegalArgumentException.class, ()->
88        envObj.addLifeForm(-1, 0, bob));
89}
90
91

```

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

```

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 (this.cells[row][col] == null) return null;
26        return this.cells[row][col].getLifeForm();
27    }
28
29
30    public LifeForm removeLifeForm(final int row, final int col) {
31        if (this.cells[row][col] != null) {
32            LifeForm tempCells = this.cells[row][col].getLifeForm();
33            this.cells[row][col].removeLifeForm();
34            return tempCells;
35        }
36        return null;
37    }
38 }

```



```

51     //Add the bob LF to [0][0] of our new array
52     envObj.addLifeForm(0, 0, bob);
53     //Check if the LF removed matches that of bob (which we just added)
54     assertEquals(envObj.removeLifeForm(0, 0), bob);
55     //Check if the LF was actually deleted
56     assertNull(envObj.getLifeForm(0, 0));
57     //Check if removeLifeForm returns null appropriately
58     assertNull(envObj.removeLifeForm(0, 0));
59 }
60
61 @Test
62 void testGetLifeFormByAddingObject() {
63     envObj.addLifeForm(0, 0, bob);
64     assertEquals(envObj.getLifeForm(0, 0), bob);
65 }
66
67 @Test
68 void testGetLifeFormWhenALifeFormIsFillingTheCells() {
69     LifeForm jimmy = new LifeForm("Jimmy", 40);
70
71     envObj.addLifeForm(1, 2, bob);
72     assertEquals(envObj.getLifeForm(1, 2), bob);
73     envObj.removeLifeForm(1, 2);
74     envObj.addLifeForm(1, 2, jimmy);
75     assertEquals(envObj.getLifeForm(1, 2), jimmy);
76 }
77
78 @Test
79 void testEnvironmentWhenCellsAreNull() {
80     assertThrows(NullPointerException.class, ()-> {
81         new Environment(null);
82     });
83 }
84
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
97

```

testAddLifeFormWhenEntityIsNull: *passed right away

```

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 (this.cells[row][col] == null) return null;
26        return this.cells[row][col].getLifeForm();
27    }
28
29
30    public LifeForm removeLifeForm(final int row, final int col) {
31        if (this.cells[row][col] != null) {
32            LifeForm tempCells = this.cells[row][col].getLifeForm();
33            this.cells[row][col].removeLifeForm();
34            return tempCells;
35        }
36        return null;
37    }
38 }

```



```

55     //Check if the LF was actually deleted
56     assertNull(envObj.getLifeForm(0, 0));
57     //Check if removeLifeForm returns null appropriately
58     assertNull(envObj.removeLifeForm(0, 0));
59 }
60
61 @Test
62 void testGetLifeFormByAddingObject() {
63     envObj.addLifeForm(0, 0, bob);
64     assertEquals(envObj.getLifeForm(0, 0), bob);
65 }
66
67 @Test
68 void testGetLifeFormWhenALifeFormIsFillingTheCells() {
69     LifeForm jimmy = new LifeForm("Jimmy", 40);
70
71     envObj.addLifeForm(1, 2, bob);
72     assertEquals(envObj.getLifeForm(1, 2), bob);
73     envObj.removeLifeForm(1, 2);
74     envObj.addLifeForm(1, 2, jimmy);
75     assertEquals(envObj.getLifeForm(1, 2), jimmy);
76 }
77
78 @Test
79 void testEnvironmentWhenCellsAreNull() {
80     assertThrows(NullPointerException.class, ()-> {
81         new Environment(null);
82     });
83 }
84
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
97 @Test
98 void testAddLifeFormWhenEntityIsNull() {
99     assertFalse(envObj.addLifeForm(0, 0, null));
100 }
101

```

testGetLifeFormWhenRowsAreNegative:

```

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 ((this.cells[row][col] == null)) return null;
26        return this.cells[row][col].getLifeForm();
27    }
28
29
30    public LifeForm removeLifeForm(final int row, final int col) {
31        if (this.cells[row][col] != null) {
32            LifeForm tempCells = this.cells[row][col].getLifeForm();
33            this.cells[row][col].removeLifeForm();
34            return tempCells;
35        }
36        return null;
37    }
38 }

```



```

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) throw new IllegalArgumentException("Bad Params in getLifeForm");
26        if ((this.cells[row][col] == null)) return null;
27        return this.cells[row][col].getLifeForm();
28    }
29
30    public LifeForm removeLifeForm(final int row, final int col) {
31        if (this.cells[row][col] != null) {
32            LifeForm tempCells = this.cells[row][col].getLifeForm();
33            this.cells[row][col].removeLifeForm();
34            return tempCells;
35        }
36        return null;
37    }
38 }

```



```

1 CSDC212Lab6CellTests.java 2 CSDC212Lab6LifeFormTest... 3 CSDC212Lab6Tests.java 4 CSDC212Lab6Environment...
61  @Test
62  void testGetLifeFormByAddingObject() {
63      envObj.addLifeForm(0, 0, bob);
64      assertEquals(envObj.getLifeForm(0, 0), bob);
65  }
66
67  @Test
68  void testGetLifeFormWhenALifeFormIsFillingTheCells() {
69      LifeForm jimmy = new LifeForm("Jimmy", 40);
70
71      envObj.addLifeForm(1, 2, bob);
72      assertEquals(envObj.getLifeForm(1, 2), bob);
73      envObj.removeLifeForm(1, 2);
74      envObj.addLifeForm(1, 2, jimmy);
75      assertEquals(envObj.getLifeForm(1, 2), jimmy);
76  }
77
78  @Test
79  void testEnvironmentWhenCellsAreNull() {
80      assertThrows(NullPointerException.class, ()-> {
81          new Environment(null);
82      });
83  }
84
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 }

```



```

1 CSDC212Lab6CellTests.java 2 CSDC212Lab6LifeFormTest... 3 CSDC212Lab6Tests.java 4 CSDC212Lab6Environment...
61  @Test
62  void testGetLifeFormByAddingObject() {
63      envObj.addLifeForm(0, 0, bob);
64      assertEquals(envObj.getLifeForm(0, 0), bob);
65  }
66
67  @Test
68  void testGetLifeFormWhenALifeFormIsFillingTheCells() {
69      LifeForm jimmy = new LifeForm("Jimmy", 40);
70
71      envObj.addLifeForm(1, 2, bob);
72      assertEquals(envObj.getLifeForm(1, 2), bob);
73      envObj.removeLifeForm(1, 2);
74      envObj.addLifeForm(1, 2, jimmy);
75      assertEquals(envObj.getLifeForm(1, 2), jimmy);
76  }
77
78  @Test
79  void testEnvironmentWhenCellsAreNull() {
80      assertThrows(NullPointerException.class, ()-> {
81          new Environment(null);
82      });
83  }
84
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 }

```

testRemoveLifeFormWhenRowsAreNegative:

```

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("Bad Params in getLifeForm");
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 (this.cells[row][col] != null) {
33            LifeForm tempCells = this.cells[row][col].getLifeForm();
34            this.cells[row][col].removeLifeForm();
35            return tempCells;
36        }
37        return null;
38    }
39 }

```



```

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("Bad Params in getLifeForm");
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("Bad Params in removeLifeForm");
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 }

```



```

1 CSDC212Lab6CellTests.java 2 CSDC212Lab6LifeFormTest... 3 CSDC212Lab6Tests.java 4 CSDC212Lab6Environment...
73     envObj.removeLifeForm(1, 2);
74     envObj.addLifeForm(1, 2, jimmy);
75     assertEquals(envObj.getLifeForm(1, 2), jimmy);
76 }
77
78 @Test
79 void testEnvironmentWhenCellsAreNull() {
80     assertThrows(NullPointerException.class, ()-> {
81         new Environment(null);
82     });
83 }
84
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 }
97 @Test
98 void testAddLifeFormWhenEntityIsNull() {
99     assertFalse(envObj.addLifeForm(0, 0, null));
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

```



```

1 CSDC212Lab6CellTests.java 2 CSDC212Lab6LifeFormTest... 3 CSDC212Lab6Tests.java 4 CSDC212Lab6Environment...
73     envObj.removeLifeForm(1, 2);
74     envObj.addLifeForm(1, 2, jimmy);
75     assertEquals(envObj.getLifeForm(1, 2), jimmy);
76 }
77
78 @Test
79 void testEnvironmentWhenCellsAreNull() {
80     assertThrows(NullPointerException.class, ()-> {
81         new Environment(null);
82     });
83 }
84
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 }
97 void testAddLifeFormWhenEntityIsNull() {
98     assertFalse(envObj.addLifeForm(0, 0, null));
99 }
100 @Test
101 void testGetLifeFormWhenRowsAreNegative() {
102     assertThrows(IllegalArgumentException.class, ()-> {
103         envObj.getLifeForm(-1, 0);
104     });
105 }
106 @Test
107 void testGetLifeFormWhenColsAreNegative() {
108     assertThrows(IllegalArgumentException.class, ()-> {
109         envObj.getLifeForm(0, -1);
110     });
111 }
112 @Test
113 void testRemoveLifeFormWhenRowsAreNegative() {
114     assertThrows(IllegalArgumentException.class, ()-> {
115         envObj.removeLifeForm(-1, 0);
116     });
117 }
118 }
119

```

testRemoveLifeFormWhenColsAreNegative:

```

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("Bad Params in getLifeForm");
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) throw new IllegalArgumentException("Bad Params in removeLifeForm");
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 }

```



```

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("Bad Params in getLifeForm");
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("Bad Params in removeLifeForm");
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 }

```



```

79 void testEnvironmentWhenCellsAreNull() {
80     assertThrows(NullPointerException.class, ()-> {
81         new Environment(null);
82     });
83 }
84
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 }
97 @Test
98 void testAddLifeFormWhenEntityIsNull() {
99     assertFalse(envObj.addLifeForm(0, 0, null));
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

```



```

79 void testEnvironmentWhenCellsAreNull() {
80     assertThrows(NullPointerException.class, ()-> {
81         new Environment(null);
82     });
83 }
84
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 }
97 @Test
98 void testAddLifeFormWhenEntityIsNull() {
99     assertFalse(envObj.addLifeForm(0, 0, null));
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

```

testAddLifeFormThrowsRightMessage:

```

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 }

```

```

1 CSDC212Lab6CellTests.java 2 CSDC212Lab6LifeFormTes... 3 CSDC212Lab6Tests.java 4 CSDC212Lab6Environmen...
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131

```

```

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 }

```

```

1 CSDC212Lab6CellTests.java 2 CSDC212Lab6LifeFormTes... 3 CSDC212Lab6Tests.java 4 CSDC212Lab6Environmen...
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131

```

testGetLifeFormThrowsRightMessage:

```

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 }

```



```

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("Bad Params in getLifeForm");
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 }

```



```

1 CSCD212Lab6CellTests.java 2 CSCD212Lab6LifeFormTes... 3 CSCD212Lab6Tests.java 4 CSCD212Lab6Environmen...
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● @Test
101 void testGetLifeFormWhenRowsAreNegative() {
102     assertThrows(IllegalArgumentException.class, ()->
103         envObj.getLifeForm(-1, 0));
104 }
105● @Test
106 void testGetLifeFormWhenColsAreNegative() {
107     assertThrows(IllegalArgumentException.class, ()->
108         envObj.getLifeForm(0, -1));
109 }
110● @Test
111 void testRemoveLifeFormWhenRowsAreNegative() {
112     assertThrows(IllegalArgumentException.class, ()->
113         envObj.removeLifeForm(-1, 0));
114 }
115● @Test
116 void testRemoveLifeFormWhenColsAreNegative() {
117     assertThrows(IllegalArgumentException.class, ()->
118         envObj.removeLifeForm(0, -1));
119 }
120● @Test void testAddLifeFormThrowsRightMessage() {
121     Throwable exception = assertThrows(IllegalArgumentException.class, ()->
122         envObj.addLifeForm(-1, 0, bob));
123     assertEquals("Bad Params in addLifeForm", exception.getMessage());
124 }
125● @Test void testGetLifeFormThrowsRightMessage() {
126     Throwable exception = assertThrows(IllegalArgumentException.class, ()->
127         envObj.getLifeForm(-1, 0));
128     assertEquals("Bad Params in getLifeForm", exception.getMessage());
129 }
130● @Test void testRemoveLifeFormThrowsRightMessage() {
131     Throwable exception = assertThrows(IllegalArgumentException.class, ()->
132         envObj.removeLifeForm(-1, 0));
133     assertEquals("Bad Params in removeLifeForm", exception.getMessage());
134 }
135 }
136 }

137

```



```

1 CSCD212Lab6CellTests.java 2 CSCD212Lab6LifeFormTes... 3 CSCD212Lab6Tests.java 4 CSCD212Lab6Environmen...
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● @Test
101 void testGetLifeFormWhenRowsAreNegative() {
102     assertThrows(IllegalArgumentException.class, ()->
103         envObj.getLifeForm(-1, 0));
104 }
105● @Test
106 void testGetLifeFormWhenColsAreNegative() {
107     assertThrows(IllegalArgumentException.class, ()->
108         envObj.getLifeForm(0, -1));
109 }
110● @Test
111 void testRemoveLifeFormWhenRowsAreNegative() {
112     assertThrows(IllegalArgumentException.class, ()->
113         envObj.removeLifeForm(-1, 0));
114 }
115● @Test
116 void testRemoveLifeFormWhenColsAreNegative() {
117     assertThrows(IllegalArgumentException.class, ()->
118         envObj.removeLifeForm(0, -1));
119 }
120● @Test void testAddLifeFormThrowsRightMessage() {
121     Throwable exception = assertThrows(IllegalArgumentException.class, ()->
122         envObj.addLifeForm(-1, 0, bob));
123     assertEquals("Bad Params in addLifeForm", exception.getMessage());
124 }
125● @Test void testGetLifeFormThrowsRightMessage() {
126     Throwable exception = assertThrows(IllegalArgumentException.class, ()->
127         envObj.getLifeForm(-1, 0));
128     assertEquals("Bad Params in getLifeForm", exception.getMessage());
129 }
130● @Test void testRemoveLifeFormThrowsRightMessage() {
131     Throwable exception = assertThrows(IllegalArgumentException.class, ()->
132         envObj.removeLifeForm(-1, 0));
133     assertEquals("Bad Params in removeLifeForm", exception.getMessage());
134 }
135 }
136 }

137

```

testRemoveLifeFormThrowsRightMessage:

```

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("Bad Params in getLifeForm");
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 }

```

```

1 CSDC212Lab6CellTests.java 2 CSDC212Lab6LifeFormTests.java 3 CSDC212Lab6Tests.java 4 CSDC212Lab6Environment.java
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 @Test void testGetLifeFormThrowsRightMessage() {
132     Throwable exception = assertThrows(IllegalArgumentException.class, ()->
133         envObj.getLifeForm(-1, 0));
134     assertEquals("Bad Params in getLifeForm", exception.getMessage());
135 }
136
137 @Test void testRemoveLifeFormThrowsRightMessage() {
138     Throwable exception = assertThrows(IllegalArgumentException.class, ()->
139         envObj.removeLifeForm(-1, 0));
140     assertEquals("Bad Params in removeLifeForm", exception.getMessage());
141 }
142 }
143

```

```

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("Bad Params in getLifeForm");
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("Bad Params in removeLifeForm");
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 }

```

```

1 CSDC212Lab6CellTests.java 2 CSDC212Lab6LifeFormTests.java 3 CSDC212Lab6Tests.java 4 CSDC212Lab6Environment.java
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 @Test void testGetLifeFormThrowsRightMessage() {
132     Throwable exception = assertThrows(IllegalArgumentException.class, ()->
133         envObj.getLifeForm(-1, 0));
134     assertEquals("Bad Params in getLifeForm", exception.getMessage());
135 }
136
137 @Test void testRemoveLifeFormThrowsRightMessage() {
138     Throwable exception = assertThrows(IllegalArgumentException.class, ()->
139         envObj.removeLifeForm(-1, 0));
140     assertEquals("Bad Params in removeLifeForm", exception.getMessage());
141 }
142 }
143

```

testGetLifeFormWhenCellsIsNull:

```

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("Bad Params in getLifeForm");
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("Bad Params in removeLifeForm");
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 }

```



```

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("Bad Params in getLifeForm");
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("Bad Params in removeLifeForm");
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 }

```



```

1 CSDC212Lab6CellTest.java 2 CSDC212Lab6LifeFormTest.java 3 CSDC212Lab6Tests.java 4 CSDC212Lab6Environment.java
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 @Test void testGetLifeFormThrowsRightMessage() {
132     Throwable exception = assertThrows(IllegalArgumentException.class, ()->
133         envObj.getLifeForm(-1, 0));
134     assertEquals("Bad Params in getLifeForm", exception.getMessage());
135 }
136
137 @Test void testRemoveLifeFormThrowsRightMessage() {
138     Throwable exception = assertThrows(IllegalArgumentException.class, ()->
139         envObj.removeLifeForm(-1, 0));
140     assertEquals("Bad Params in removeLifeForm", exception.getMessage());
141 }
142
143 @Test void getLifeFormWhenCellsIsNull() {
144     cells[0][0] = null;
145     assertNull(envObj.getLifeForm(0, 0));
146 }
147
148
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 @Test void testGetLifeFormThrowsRightMessage() {
132     Throwable exception = assertThrows(IllegalArgumentException.class, ()->
133         envObj.getLifeForm(-1, 0));
134     assertEquals("Bad Params in getLifeForm", exception.getMessage());
135 }
136
137 @Test void testRemoveLifeFormThrowsRightMessage() {
138     Throwable exception = assertThrows(IllegalArgumentException.class, ()->
139         envObj.removeLifeForm(-1, 0));
140     assertEquals("Bad Params in removeLifeForm", exception.getMessage());
141 }
142
143 @Test void getLifeFormWhenCellsIsNull() {
144     cells[0][0] = null;
145     assertNull(envObj.getLifeForm(0, 0));
146 }
147
148

```

testRemoveLifeFormWhenCellsIsNull: *passed right away

```

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("Bad Params in getLifeForm");
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("Bad Params in removeLifeForm");
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 }

```



```

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    @Test void testGetLifeFormThrowsRightMessage() {
132        Throwable exception = assertThrows(IllegalArgumentException.class, ()->
133            envObj.getLifeForm(-1, 0));
134        assertEquals("Bad Params in getLifeForm", exception.getMessage());
135    }
136
137    @Test void testRemoveLifeFormThrowsRightMessage() {
138        Throwable exception = assertThrows(IllegalArgumentException.class, ()->
139            envObj.removeLifeForm(-1, 0));
140        assertEquals("Bad Params in removeLifeForm", exception.getMessage());
141    }
142
143    @Test void testGetLifeFormWhenCellsIsNull() {
144        cells[0][0] = null;
145        assertNull(envObj.getLifeForm(0, 0));
146    }
147
148    @Test void testRemoveLifeFormWhenCellsIsNull() {
149        cells[0][0] = null;
150        assertNull(envObj.removeLifeForm(0, 0));
151    }
152 }

```

testAddLifeFormOutOfBoundsProperMessage:

```

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 (row > this.cells.length || col > this.cells.length) throw new ArrayIndexOutOfBoundsException();
18        if (entity != null) {
19            cells[row][col].addLifeForm(entity);
20            return true;
21        }
22        return false;
23    }
24
25    public LifeForm getLifeForm(final int row, final int col) {
26        if (row < 0 || col < 0) throw new IllegalArgumentException("Bad Params in getLifeForm");
27        if (this.cells[row][col] == null) return null;
28        return this.cells[row][col].getLifeForm();
29    }
30
31
32    public LifeForm removeLifeForm(final int row, final int col) {
33        if (row < 0 || col < 0) throw new IllegalArgumentException("Bad Params in removeLifeForm");
34        if (this.cells[row][col] != null) {
35            LifeForm tempCells = this.cells[row][col].getLifeForm();
36            this.cells[row][col].removeLifeForm();
37            return tempCells;
38        }
39        return null;
40    }
41 }

```



```

118
119    @Test
120    void testRemoveLifeFormWhenColsAreNegative() {
121        assertThrows(IllegalArgumentException.class, ()->
122            envObj.removeLifeForm(0, -1));
123    }
124
125    @Test
126    void testAddLifeFormThrowsRightMessage() {
127        Throwable exception = assertThrows(IllegalArgumentException.class, ()->
128            envObj.addLifeForm(-1, 0, bob));
129        assertEquals("Bad Params in addLifeForm", exception.getMessage());
130    }
131
132    @Test
133    void testGetLifeFormThrowsRightMessage() {
134        Throwable exception = assertThrows(IllegalArgumentException.class, ()->
135            envObj.getLifeForm(-1, 0));
136        assertEquals("Bad Params in getLifeForm", exception.getMessage());
137    }
138
139    @Test
140    void testRemoveLifeFormThrowsRightMessage() {
141        Throwable exception = assertThrows(IllegalArgumentException.class, ()->
142            envObj.removeLifeForm(-1, 0));
143        assertEquals("Bad Params in removeLifeForm", exception.getMessage());
144    }
145
146    @Test
147    void testGetLifeFormWhenCellsIsNull() {
148        cells[0][0] = null;
149        assertNull(envObj.getLifeForm(0, 0));
150    }
151
152    @Test
153    void testRemoveLifeFormWhenCellsIsNull() {
154        cells[0][0] = null;
155        assertNull(envObj.removeLifeForm(0, 0));
156    }
157
158    @Test
159    void testAddLifeFormOutOfBounds() {
160        Throwable exception = assertThrows(ArrayIndexOutOfBoundsException.class, ()->
161            envObj.addLifeForm(5, 0, bob));
162        assertEquals("Row and Col Params OutOfBounds", exception.getMessage());
163    }
164 }

```

```

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 (row > this.cells.length || col > this.cells.length) throw new ArrayIndexOutOfBoundsException();
18        if (entity != null) {
19            cells[row][col].addLifeForm(entity);
20            return true;
21        }
22        return false;
23    }
24
25    public LifeForm getLifeForm(Final int row, Final int col) {
26        if (row < 0 || col < 0) throw new IllegalArgumentException("Bad Params in getLifeForm");
27        if (this.cells[row][col] == null) return null;
28        return this.cells[row][col].getLifeForm();
29    }
30
31
32    public LifeForm removeLifeForm(Final int row, Final int col) {
33        if (row < 0 || col < 0) throw new IllegalArgumentException("Bad Params in removeLifeForm");
34        if (this.cells[row][col] != null) {
35            LifeForm tempCells = this.cells[row][col].getLifeForm();
36            this.cells[row][col].removeLifeForm();
37            return tempCells;
38        }
39        return null;
40    }
41 }

```

```

1 CSDC212Lab6CellTests.java 2 CSDC212Lab6LifeFormTes... 3 CSDC212Lab6Tests.java 4 CSDC212Lab6Environmen...
118
119    @Test
120    void testRemoveLifeFormWhenColsAreNegative() {
121        assertThrows(IllegalArgumentException.class, ()->
122            envObj.removeLifeForm(0, -1));
123    }
124
125    @Test
126    void testAddLifeFormThrowsRightMessage() {
127        Throwable exception = assertThrows(IllegalArgumentException.class, ()->
128            envObj.addLifeForm(-1, 0, bob));
129        assertEquals("Bad Params in addLifeForm", exception.getMessage());
130    }
131
132    @Test
133    void testGetLifeFormThrowsRightMessage() {
134        Throwable exception = assertThrows(IllegalArgumentException.class, ()->
135            envObj.getLifeForm(-1, 0));
136        assertEquals("Bad Params in getLifeForm", exception.getMessage());
137    }
138
139    @Test
140    void testRemoveLifeFormThrowsRightMessage() {
141        Throwable exception = assertThrows(IllegalArgumentException.class, ()->
142            envObj.removeLifeForm(-1, 0));
143        assertEquals("Bad Params in removeLifeForm", exception.getMessage());
144    }
145
146    @Test
147    void testGetLifeFormWhenCellsIsNull() {
148        cells[0][0] = null;
149        assertNull(envObj.getLifeForm(0, 0));
150    }
151
152    @Test
153    void testRemoveLifeFormWhenCellsIsNull() {
154        cells[0][0] = null;
155        assertNull(envObj.removeLifeForm(0, 0));
156    }
157
158    @Test
159    void testAddLifeFormOutOfBoundsProperMessage() {
160        Throwable exception = assertThrows(ArrayIndexOutOfBoundsException.class, ()->
161            envObj.addLifeForm(5, 0, bob));
162        assertEquals("Row and Col Params OutOfBounds", exception.getMessage());
163    }
164 }

```

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