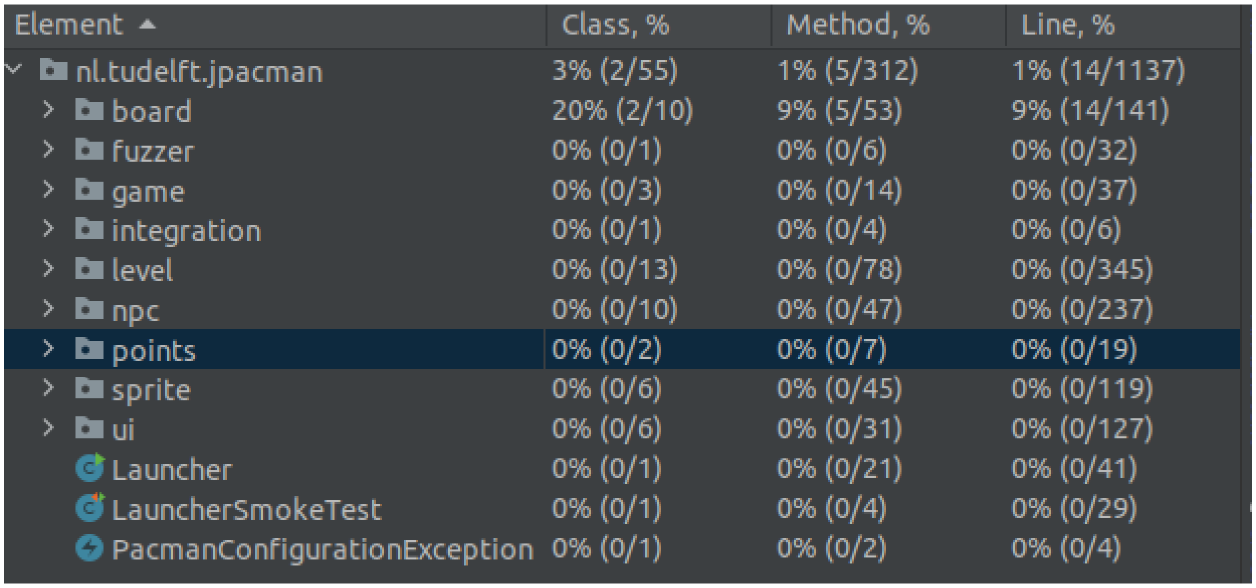
Forked Repository: https://github.com/StacL/munch

2.1: Writing Unit Tests

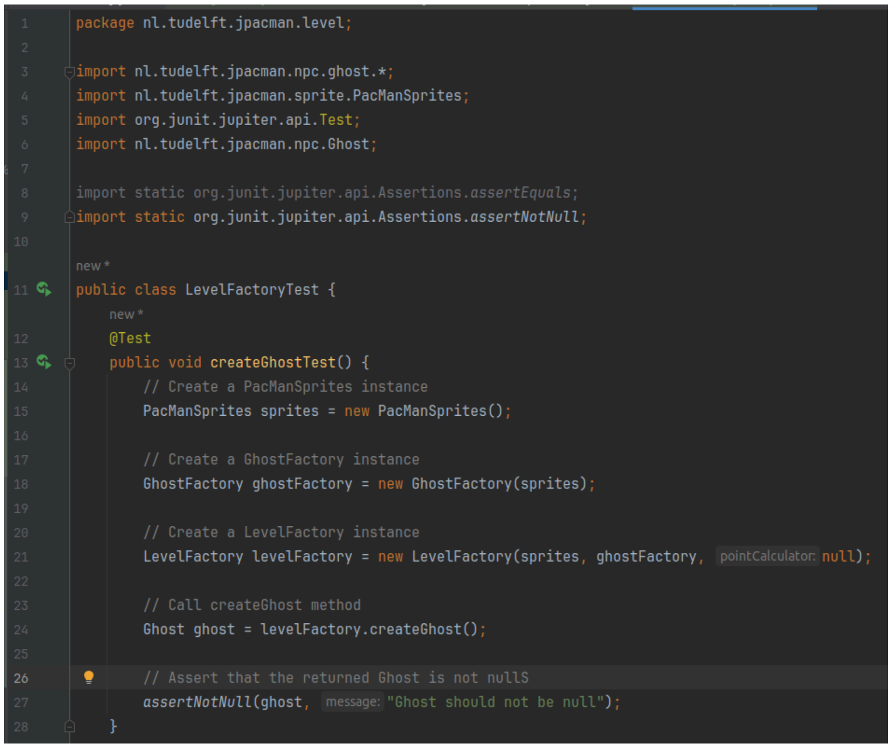


Element	Class, %	Method, %	Line, %
nl.tudelft.jpacman	3% (2/55)	1% (5/312)	1% (14/1137)
board	20% (2/10)	9% (5/53)	9% (14/141)
fuzzer	0% (0/1)	0% (0/6)	0% (0/32)
game	0% (0/3)	0% (0/14)	0% (0/37)
integration	0% (0/1)	0% (0/4)	0% (0/6)
level	0% (0/13)	0% (0/78)	0% (0/345)
npc	0% (0/10)	0% (0/47)	0% (0/237)
points	0% (0/2)	0% (0/7)	0% (0/19)
sprite	0% (0/6)	0% (0/45)	0% (0/119)
ui	0% (0/6)	0% (0/31)	0% (0/127)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)
LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)
PacmanConfigurationException	0% (0/1)	0% (0/2)	0% (0/4)

Testing coverage before creating any tests

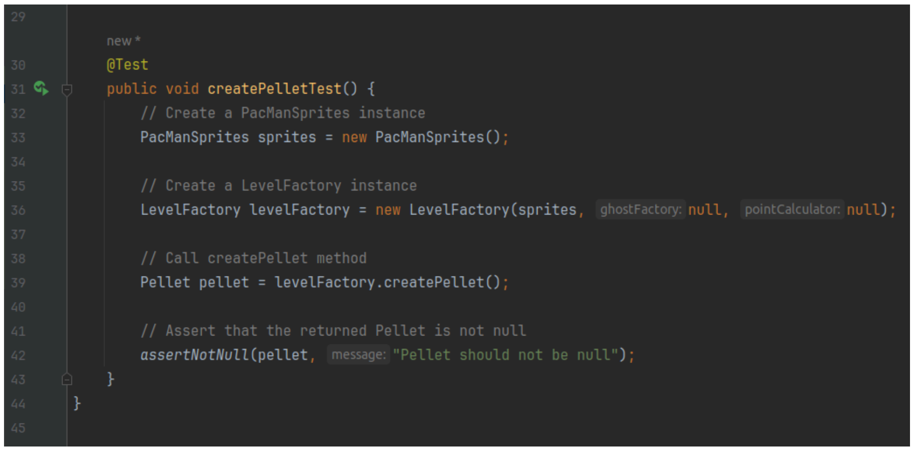
Before creating tests, the coverage is almost non-existent. Many have 0% covered on tests.

My unit tests are below:



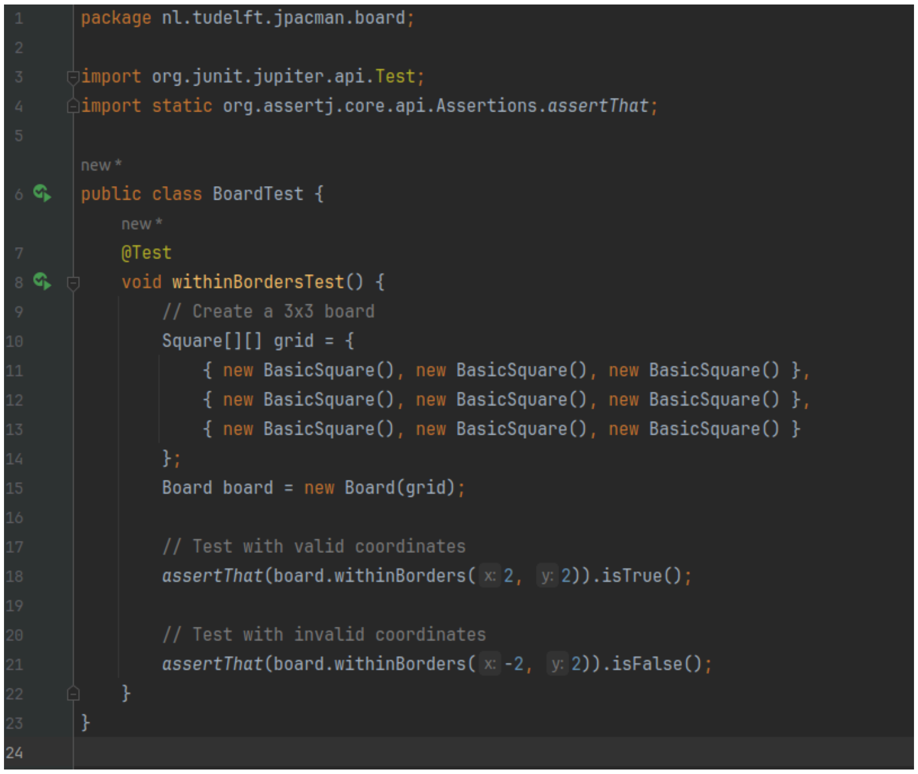
```java
package nl.tudelft.jpacman.level;

import nl.tudelft.jpacman.npc.ghost.*;
import nl.tudelft.jpacman.sprite.PacManSprites;
import org.junit.jupiter.api.Test;
import nl.tudelft.jpacman.npc.Ghost;

import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertNotNull;

new *
public class LevelFactoryTest {
    new *
    @Test
    public void createGhostTest() {
        // Create a PacManSprites instance
        PacManSprites sprites = new PacManSprites();

        // Create a GhostFactory instance
        GhostFactory ghostFactory = new GhostFactory(sprites);

        // Create a LevelFactory instance
        LevelFactory levelFactory = new LevelFactory(sprites, ghostFactory, pointCalculator: null);

        // Call createGhost method
        Ghost ghost = levelFactory.createGhost();

        // Assert that the returned Ghost is not nullS
        assertNotNull(ghost, message: "Ghost should not be null");
    }
```

Unit test for CreateGhost



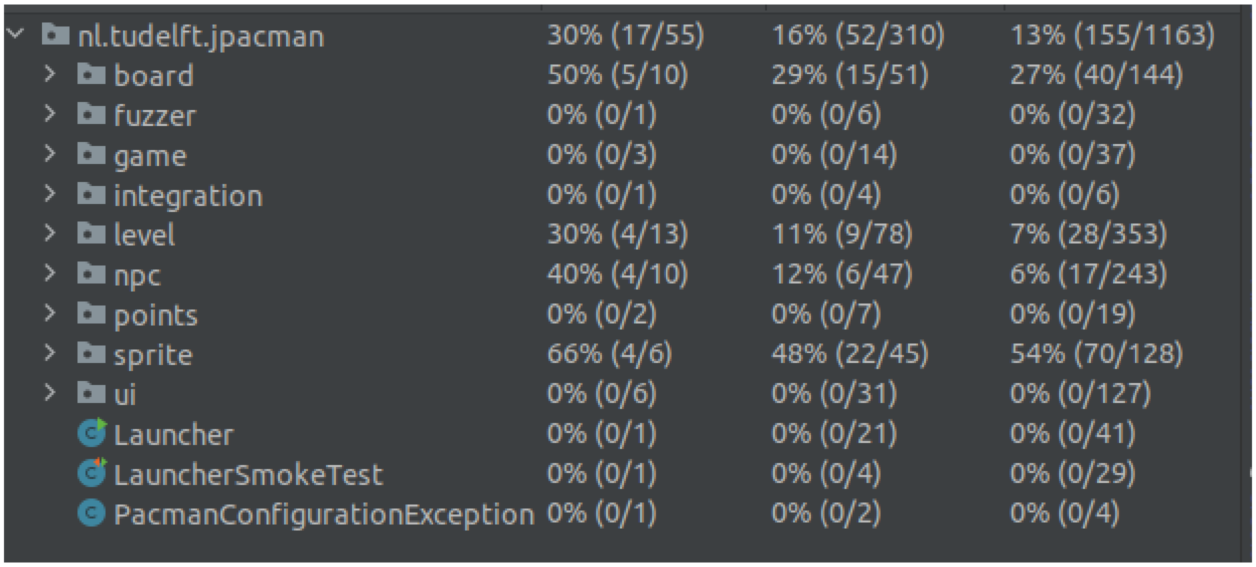
```
    new *
    @Test
    public void createPelletTest() {
        // Create a PacManSprites instance
        PacManSprites sprites = new PacManSprites();

        // Create a LevelFactory instance
        LevelFactory levelFactory = new LevelFactory(sprites, ghostFactory: null, pointCalculator: null);

        // Call createPellet method
        Pellet pellet = levelFactory.createPellet();

        // Assert that the returned Pellet is not null
        assertNotNull(pellet, message: "Pellet should not be null");
    }
}
```

Unit test for createPellet



```
package nl.tudelft.jpacman.board;

import org.junit.jupiter.api.Test;
import static org.assertj.core.api.Assertions.assertThat;

new *
public class BoardTest {
    new *
    @Test
    void withinBordersTest() {
        // Create a 3x3 board
        Square[][] grid = {
            { new BasicSquare(), new BasicSquare(), new BasicSquare() },
            { new BasicSquare(), new BasicSquare(), new BasicSquare() },
            { new BasicSquare(), new BasicSquare(), new BasicSquare() }
        };
        Board board = new Board(grid);

        // Test with valid coordinates
        assertThat(board.withinBorders( x: 2,  y: 2)).isTrue();

        // Test with invalid coordinates
        assertThat(board.withinBorders( x: -2,  y: 2)).isFalse();
    }
}
```

Unit test for withinBorders

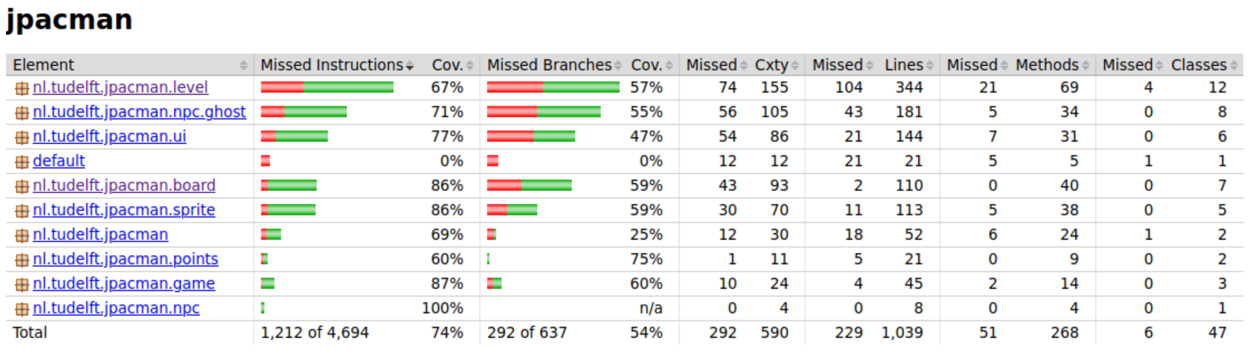


nl.tudelft.jpacman	30% (17/55)	16% (52/310)	13% (155/1163)
board	50% (5/10)	29% (15/51)	27% (40/144)
fuzzer	0% (0/1)	0% (0/6)	0% (0/32)
game	0% (0/3)	0% (0/14)	0% (0/37)
integration	0% (0/1)	0% (0/4)	0% (0/6)
level	30% (4/13)	11% (9/78)	7% (28/353)
npc	40% (4/10)	12% (6/47)	6% (17/243)
points	0% (0/2)	0% (0/7)	0% (0/19)
sprite	66% (4/6)	48% (22/45)	54% (70/128)
ui	0% (0/6)	0% (0/31)	0% (0/127)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)
LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)
PacmanConfigurationException	0% (0/1)	0% (0/2)	0% (0/4)

Coverage after creating tests

After creating the 3 unit tests, coverage increased based on where the methods I chose were located (in the Board and Level).

3: JaCoCo Report on JPacman



jpacman

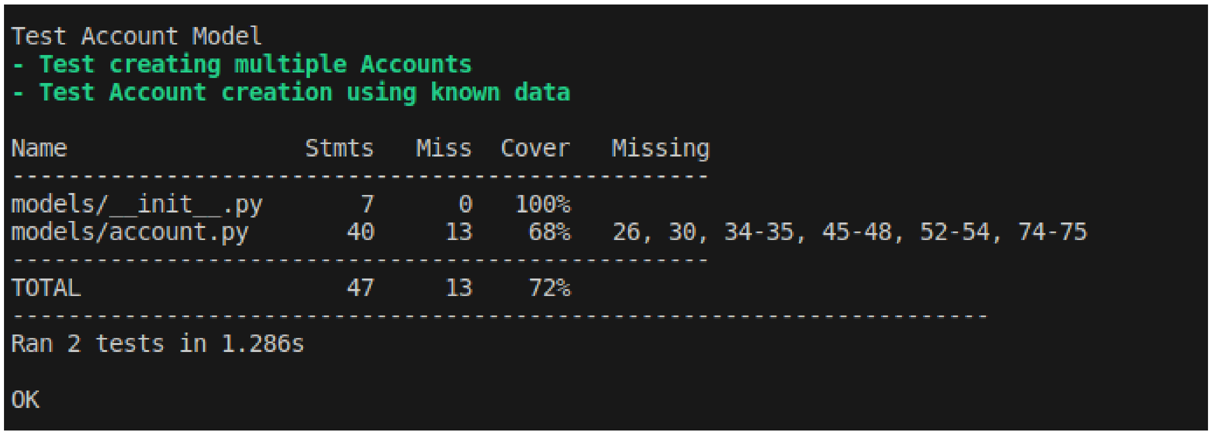
Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
nl.tudelft.jpacman.level		67%		57%	74	155	104	344	21	69	4	12
nl.tudelft.jpacman.npc.ghost		71%		55%	56	105	43	181	5	34	0	8
nl.tudelft.jpacman.ui		77%		47%	54	86	21	144	7	31	0	6
default		0%		0%	12	12	21	21	5	5	1	1
nl.tudelft.jpacman.board		86%		59%	43	93	2	110	0	40	0	7
nl.tudelft.jpacman.sprite		86%		59%	30	70	11	113	5	38	0	5
nl.tudelft.jpacman		69%		25%	12	30	18	52	6	24	1	2
nl.tudelft.jpacman.points		60%		75%	1	11	5	21	0	9	0	2
nl.tudelft.jpacman.game		87%		60%	10	24	4	45	2	14	0	3
nl.tudelft.jpacman.npc		100%		n/a	0	4	0	8	0	4	0	1
Total	1,212 of 4,694	74%	292 of 637	54%	292	590	229	1,039	51	268	6	47

The values on JaCoCo and the IntelliJ are slightly different, but that might be because the JaCoCo coverage has a lot more details. Most of the code is not covered, which is consistent with the IntelliJ report.

JaCoCo's report adds a lot more details and visuals, which was nice to see. I liked the convenience of IntelliJ's coverage window, but JaCoCo's report had a lot more details that could give a more comprehensive overview of testing.

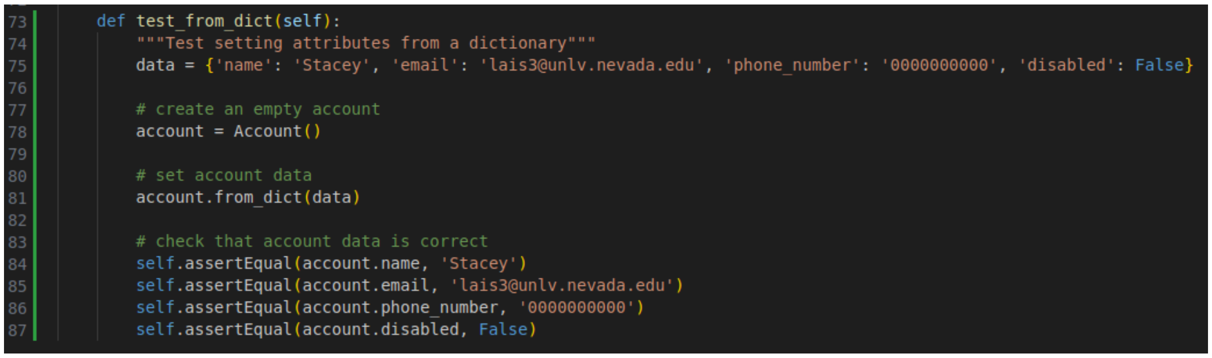
I liked JaCoCo's report because of the additional details that let me know additional details, and it was helpful seeing the visual of how much coverage there was.

4: Test Coverage for Account.py



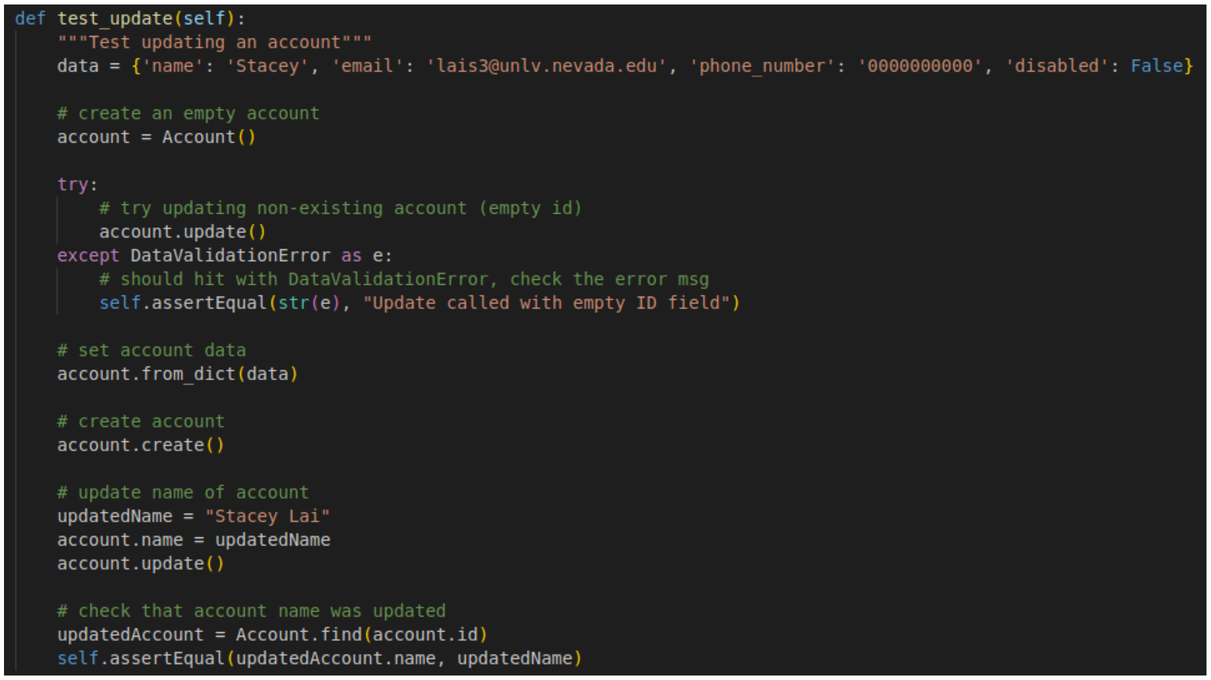
```
Test Account Model
- Test creating multiple Accounts
- Test Account creation using known data

Name                    Stmts   Miss  Cover   Missing
-----------------------------------------------------
models/__init__.py          7      0   100%
models/account.py          40     13    68%   26, 30, 34-35, 45-48, 52-54, 74-75
-----------------------------------------------------
TOTAL                      47     13    72%
----------------------------------------------------------------------
Ran 2 tests in 1.286s

OK
```

Testing coverage before creating any tests



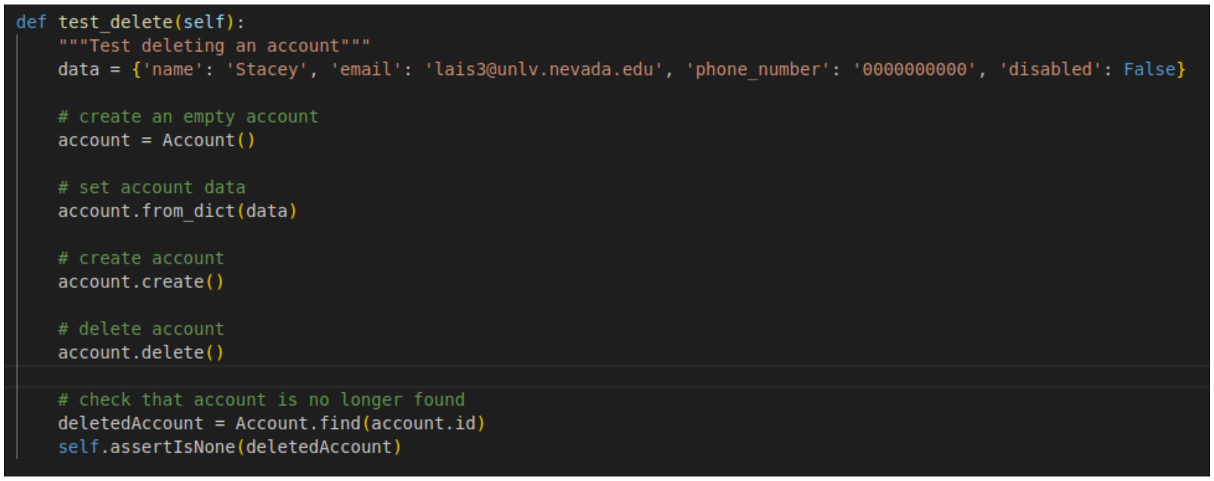
```
    def test_from_dict(self):
        """Test setting attributes from a dictionary"""
        data = {'name': 'Stacey', 'email': 'lais3@unlv.nevada.edu', 'phone_number': '0000000000', 'disabled': False}

        # create an empty account
        account = Account()

        # set account data
        account.from_dict(data)

        # check that account data is correct
        self.assertEqual(account.name, 'Stacey')
        self.assertEqual(account.email, 'lais3@unlv.nevada.edu')
        self.assertEqual(account.phone_number, '0000000000')
        self.assertEqual(account.disabled, False)
```

Test for lines 34-35 (from_dict)



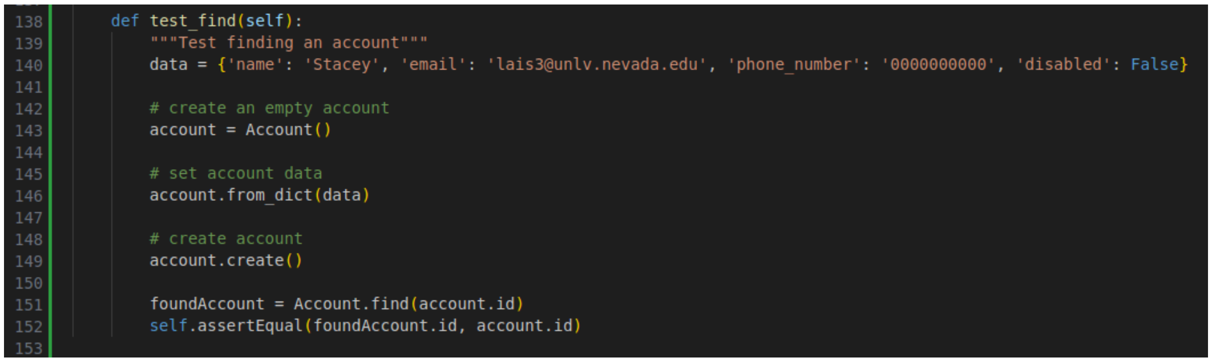
```
def test_update(self):
    """Test updating an account"""
    data = {'name': 'Stacey', 'email': 'lais3@unlv.nevada.edu', 'phone_number': '0000000000', 'disabled': False}

    # create an empty account
    account = Account()

    try:
        # try updating non-existing account (empty id)
        account.update()
    except DataValidationError as e:
        # should hit with DataValidationError, check the error msg
        self.assertEqual(str(e), "Update called with empty ID field")

    # set account data
    account.from_dict(data)

    # create account
    account.create()

    # update name of account
    updatedName = "Stacey Lai"
    account.name = updatedName
    account.update()

    # check that account name was updated
    updatedAccount = Account.find(account.id)
    self.assertEqual(updatedAccount.name, updatedName)
```

Test for lines 45-48 (update)



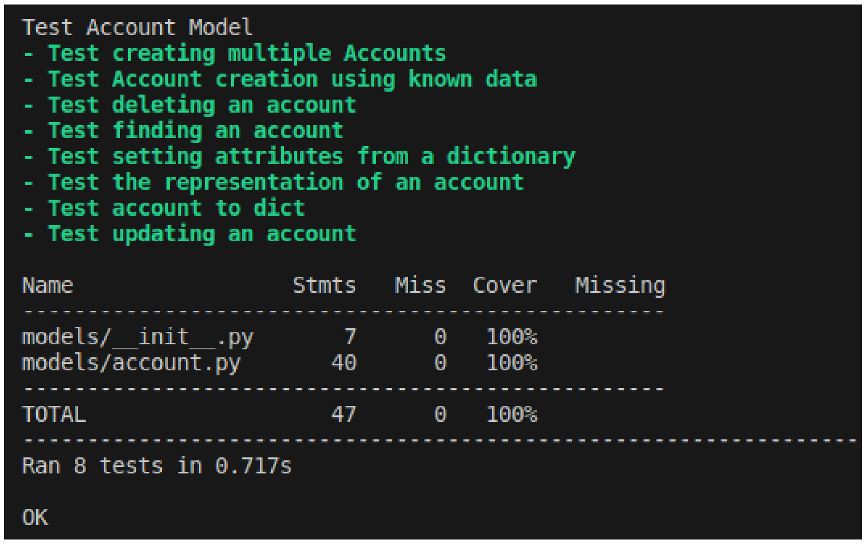
```python
def test_delete(self):
    """Test deleting an account"""
    data = {'name': 'Stacey', 'email': 'lais3@unlv.nevada.edu', 'phone_number': '0000000000', 'disabled': False}

    # create an empty account
    account = Account()

    # set account data
    account.from_dict(data)

    # create account
    account.create()

    # delete account
    account.delete()

    # check that account is no longer found
    deletedAccount = Account.find(account.id)
    self.assertIsNone(deletedAccount)
```

Test for lines 52-54 (delete)



```python
    def test_find(self):
        """Test finding an account"""
        data = {'name': 'Stacey', 'email': 'lais3@unlv.nevada.edu', 'phone_number': '0000000000', 'disabled': False}

        # create an empty account
        account = Account()

        # set account data
        account.from_dict(data)

        # create account
        account.create()

        foundAccount = Account.find(account.id)
        self.assertEqual(foundAccount.id, account.id)
```

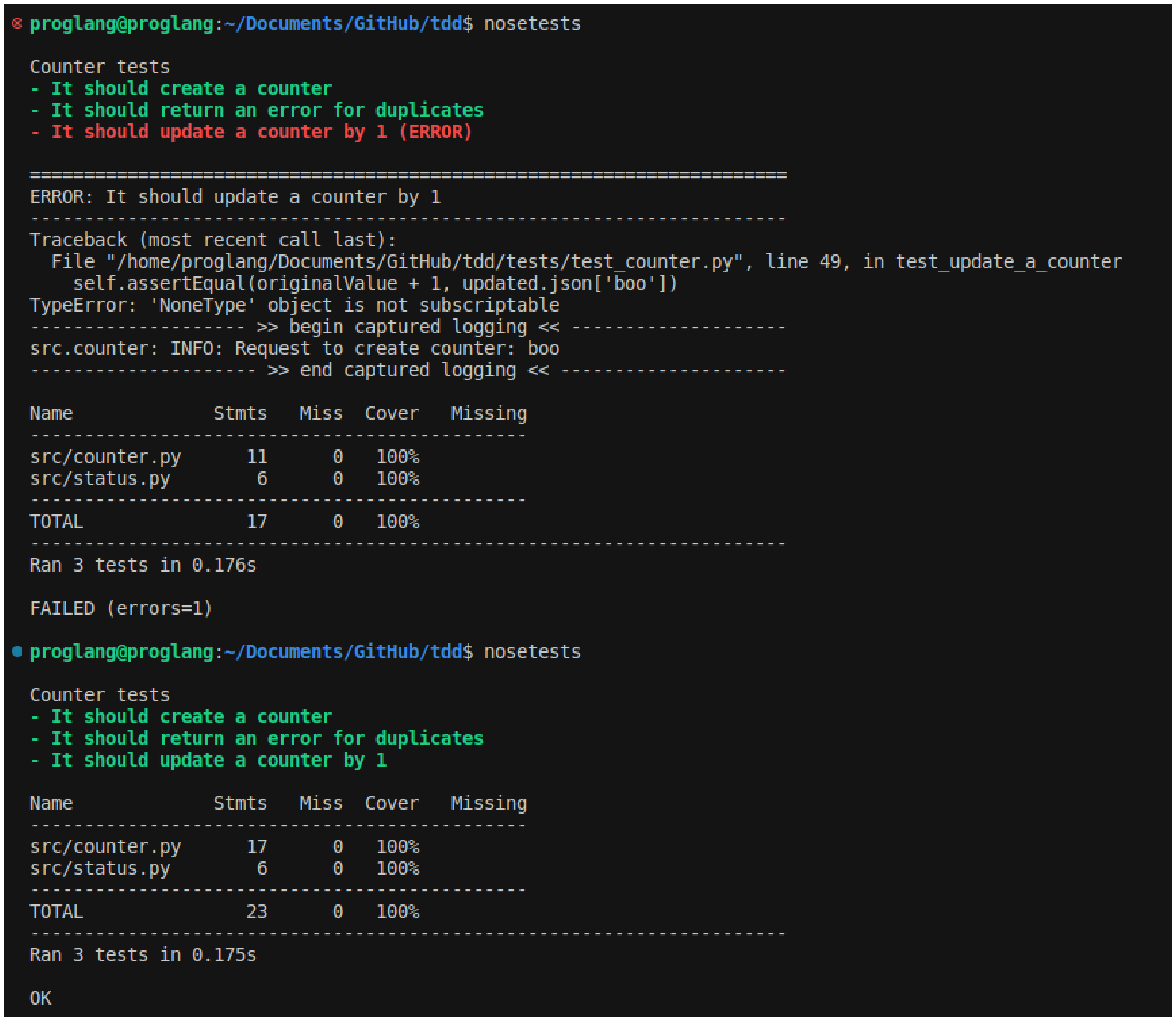
Test for lines 74-75 (find)



```
Test Account Model
- Test creating multiple Accounts
- Test Account creation using known data
- Test deleting an account
- Test finding an account
- Test setting attributes from a dictionary
- Test the representation of an account
- Test account to dict
- Test updating an account

Name                    Stmts   Miss  Cover   Missing
-----------------------------------------------------
models/__init__.py          7      0   100%
models/account.py          40      0   100%
-----------------------------------------------------
TOTAL                      47      0   100%
----------------------------------------------------------------------
Ran 8 tests in 0.717s

OK
```

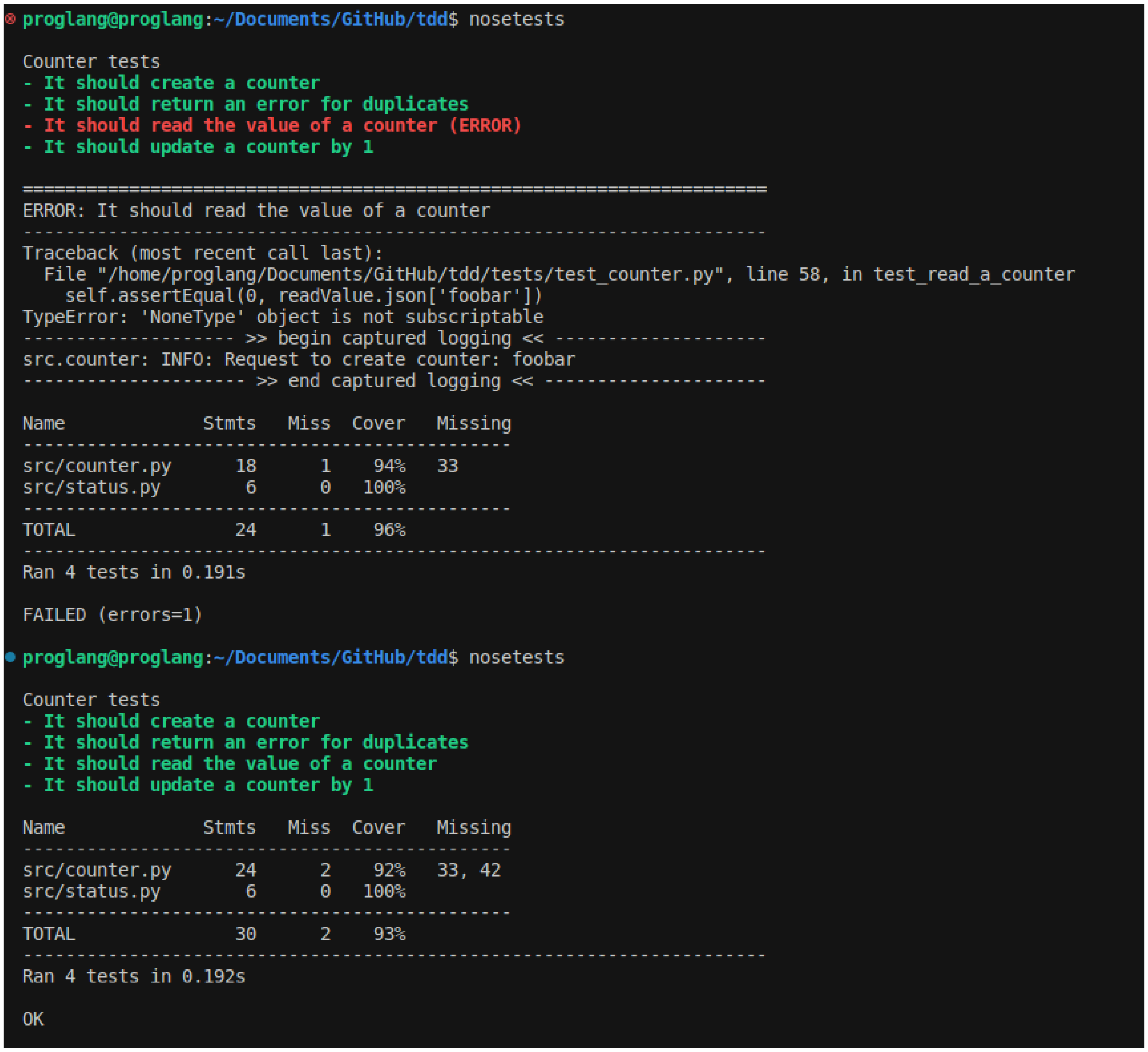
Test Coverage at 100%

Task 5: Test Driven Development (TDD)



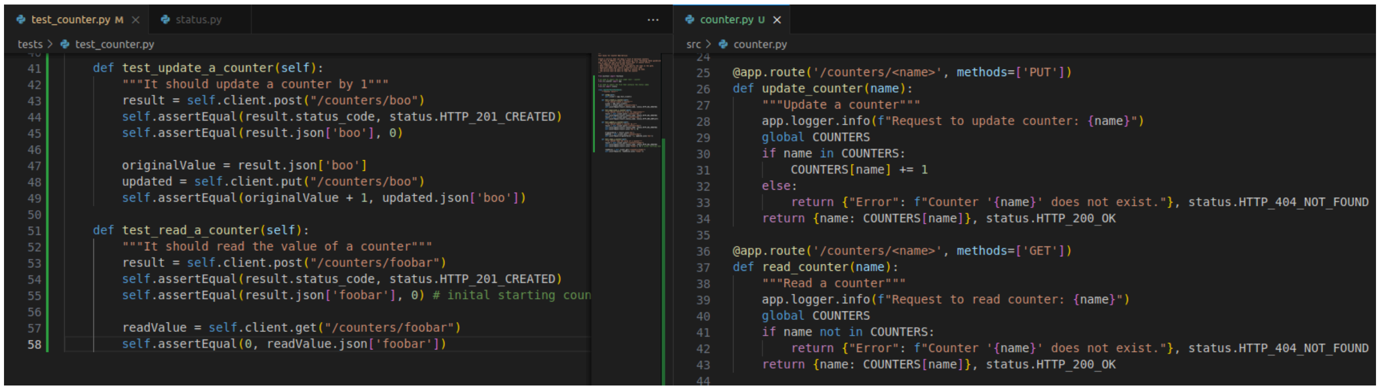
```
⊗ proglang@proglang:~/Documents/GitHub/tdd$ nosetests

  Counter tests
  - It should create a counter
  - It should return an error for duplicates
  - It should update a counter by 1 (ERROR)

  ======================================================================
  ERROR: It should update a counter by 1
  ----------------------------------------------------------------------
  Traceback (most recent call last):
    File "/home/proglang/Documents/GitHub/tdd/tests/test_counter.py", line 49, in test_update_a_counter
      self.assertEqual(originalValue + 1, updated.json['boo'])
  TypeError: 'NoneType' object is not subscriptable
  -------------------- >> begin captured logging << --------------------
  src.counter: INFO: Request to create counter: boo
  --------------------- >> end captured logging << ---------------------

  Name             Stmts   Miss  Cover   Missing
  ----------------------------------------------
  src/counter.py      11      0   100%
  src/status.py        6      0   100%
  ----------------------------------------------
  TOTAL               17      0   100%
  ----------------------------------------------------------------------
  Ran 3 tests in 0.176s

  FAILED (errors=1)

● proglang@proglang:~/Documents/GitHub/tdd$ nosetests

  Counter tests
  - It should create a counter
  - It should return an error for duplicates
  - It should update a counter by 1

  Name             Stmts   Miss  Cover   Missing
  ----------------------------------------------
  src/counter.py      17      0   100%
  src/status.py        6      0   100%
  ----------------------------------------------
  TOTAL               23      0   100%
  ----------------------------------------------------------------------
  Ran 3 tests in 0.175s

  OK
```

Red Phase and Green Phase after refactoring for Update Counter



```
proglang@proglang:~/Documents/GitHub/tdd$ nosetests

Counter tests
- It should create a counter
- It should return an error for duplicates
- It should read the value of a counter (ERROR)
- It should update a counter by 1

======================================================================
ERROR: It should read the value of a counter
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/proglang/Documents/GitHub/tdd/tests/test_counter.py", line 58, in test_read_a_counter
    self.assertEqual(0, readValue.json['foobar'])
TypeError: 'NoneType' object is not subscriptable
-------------------- >> begin captured logging << --------------------
src.counter: INFO: Request to create counter: foobar
--------------------- >> end captured logging << ---------------------

Name              Stmts   Miss  Cover   Missing
-----------------------------------------------
src/counter.py       18      1    94%   33
src/status.py         6      0   100%
-----------------------------------------------
TOTAL                24      1    96%
----------------------------------------------------------------------
Ran 4 tests in 0.191s

FAILED (errors=1)

proglang@proglang:~/Documents/GitHub/tdd$ nosetests

Counter tests
- It should create a counter
- It should return an error for duplicates
- It should read the value of a counter
- It should update a counter by 1

Name              Stmts   Miss  Cover   Missing
-----------------------------------------------
src/counter.py       24      2    92%   33, 42
src/status.py         6      0   100%
-----------------------------------------------
TOTAL                30      2    93%
----------------------------------------------------------------------
Ran 4 tests in 0.192s

OK
```

Red Phase and Green Phase after refactoring for Read Counter



```
41      def test_update_a_counter(self):
42          """It should update a counter by 1"""
43          result = self.client.post("/counters/boo")
44          self.assertEqual(result.status_code, status.HTTP_201_CREATED)
45          self.assertEqual(result.json['boo'], 0)
46
47          originalValue = result.json['boo']
48          updated = self.client.put("/counters/boo")
49          self.assertEqual(originalValue + 1, updated.json['boo'])
50
51      def test_read_a_counter(self):
52          """It should read the value of a counter"""
53          result = self.client.post("/counters/foobar")
54          self.assertEqual(result.status_code, status.HTTP_201_CREATED)
55          self.assertEqual(result.json['foobar'], 0) # inital starting coun
56
57          readValue = self.client.get("/counters/foobar")
58          self.assertEqual(0, readValue.json['foobar'])
```

```
25  @app.route('/counters/<name>', methods=['PUT'])
26  def update_counter(name):
27      """Update a counter"""
28      app.logger.info(f"Request to update counter: {name}")
29      global COUNTERS
30      if name in COUNTERS:
31          COUNTERS[name] += 1
32      else:
33          return {"Error": f"Counter '{name}' does not exist."}, status.HTTP_404_NOT_FOUND
34      return {name: COUNTERS[name]}, status.HTTP_200_OK
35
36  @app.route('/counters/<name>', methods=['GET'])
37  def read_counter(name):
38      """Read a counter"""
39      app.logger.info(f"Request to read counter: {name}")
40      global COUNTERS
41      if name not in COUNTERS:
42          return {"Error": f"Counter '{name}' does not exist."}, status.HTTP_404_NOT_FOUND
43      return {name: COUNTERS[name]}, status.HTTP_200_OK
44
```

Code For Update Counter and Read Counter