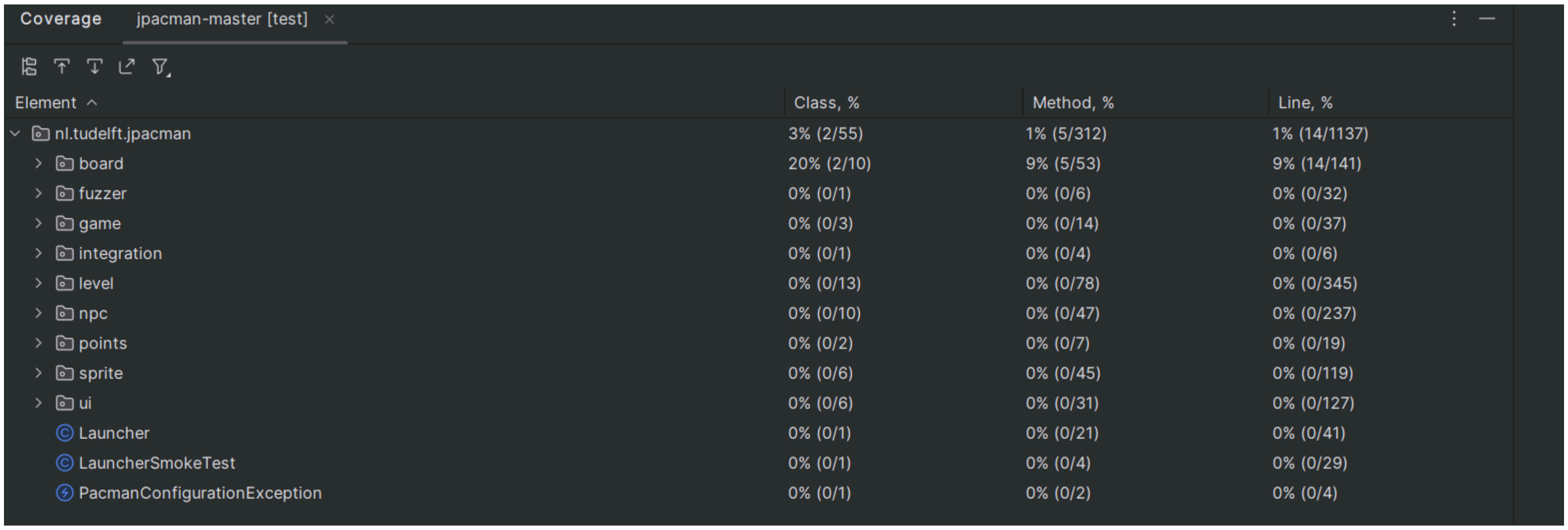
Tetsutada Kanetake

Repository fork link: munch

Unit Testing Report

Task 1:

No, the coverage is not good enough. The majority of the coverage is 0%.

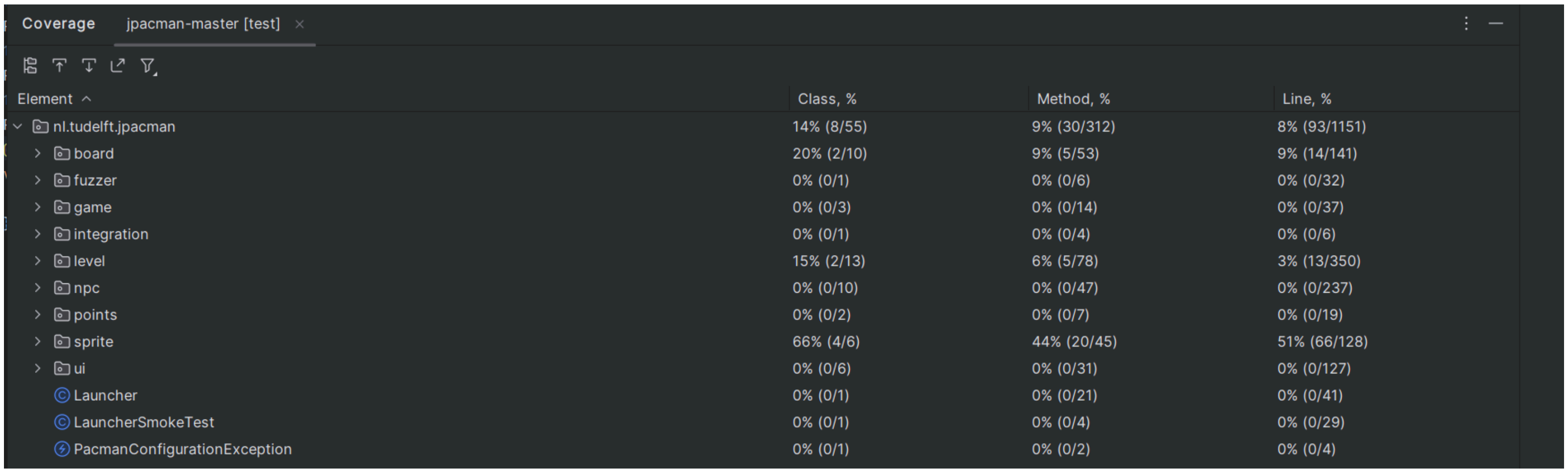


Coverage jpacman-master [test]

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)

Task 2:

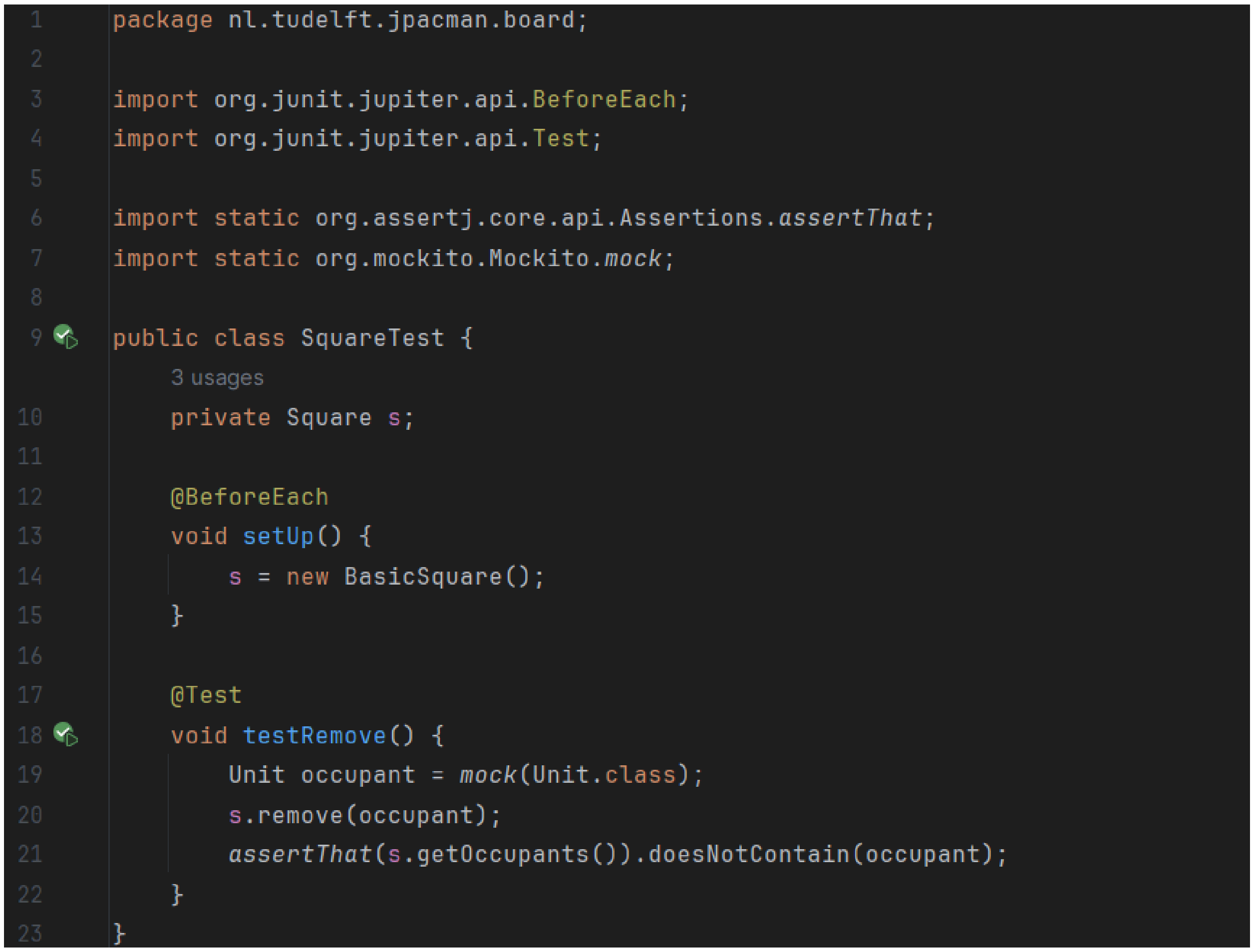
isAlive() added, coverage is improved



Coverage jpacman-master [test]

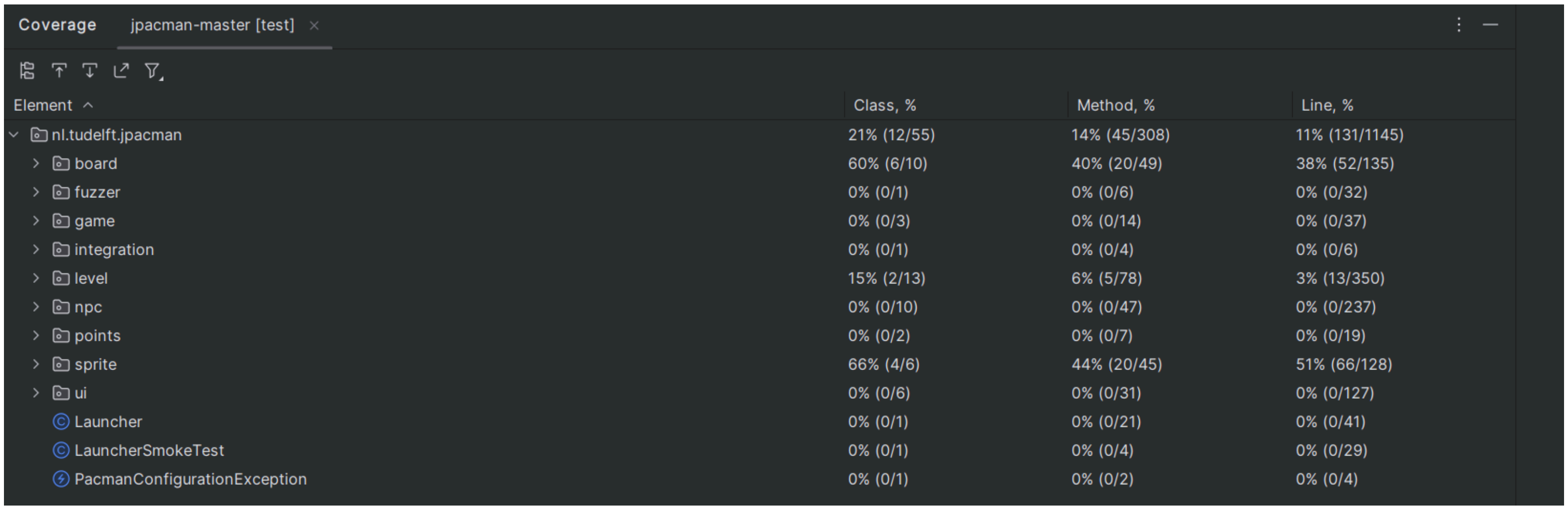
Element	Class, %	Method, %	Line, %
nl.tudelft.jpacman	14% (8/55)	9% (30/312)	8% (93/1151)
board	20% (2/10)	9% (5/53)	9% (14/141)
fuzzer	0% (0/1)	0% (0/6)	0% (0/32)
game	0% (0/3)	0% (0/14)	0% (0/37)
integration	0% (0/1)	0% (0/4)	0% (0/6)
level	15% (2/13)	6% (5/78)	3% (13/350)
npc	0% (0/10)	0% (0/47)	0% (0/237)
points	0% (0/2)	0% (0/7)	0% (0/19)
sprite	66% (4/6)	44% (20/45)	51% (66/128)
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)

Task 2.1:
I tested the remove() function in the Square class.



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

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

import static org.assertj.core.api.Assertions.assertThat;
import static org.mockito.Mockito.mock;

public class SquareTest {
    3 usages
    private Square s;

    @BeforeEach
    void setUp() {
        s = new BasicSquare();
    }

    @Test
    void testRemove() {
        Unit occupant = mock(Unit.class);
        s.remove(occupant);
        assertThat(s.getOccupants()).doesNotContain(occupant);
    }
}
```

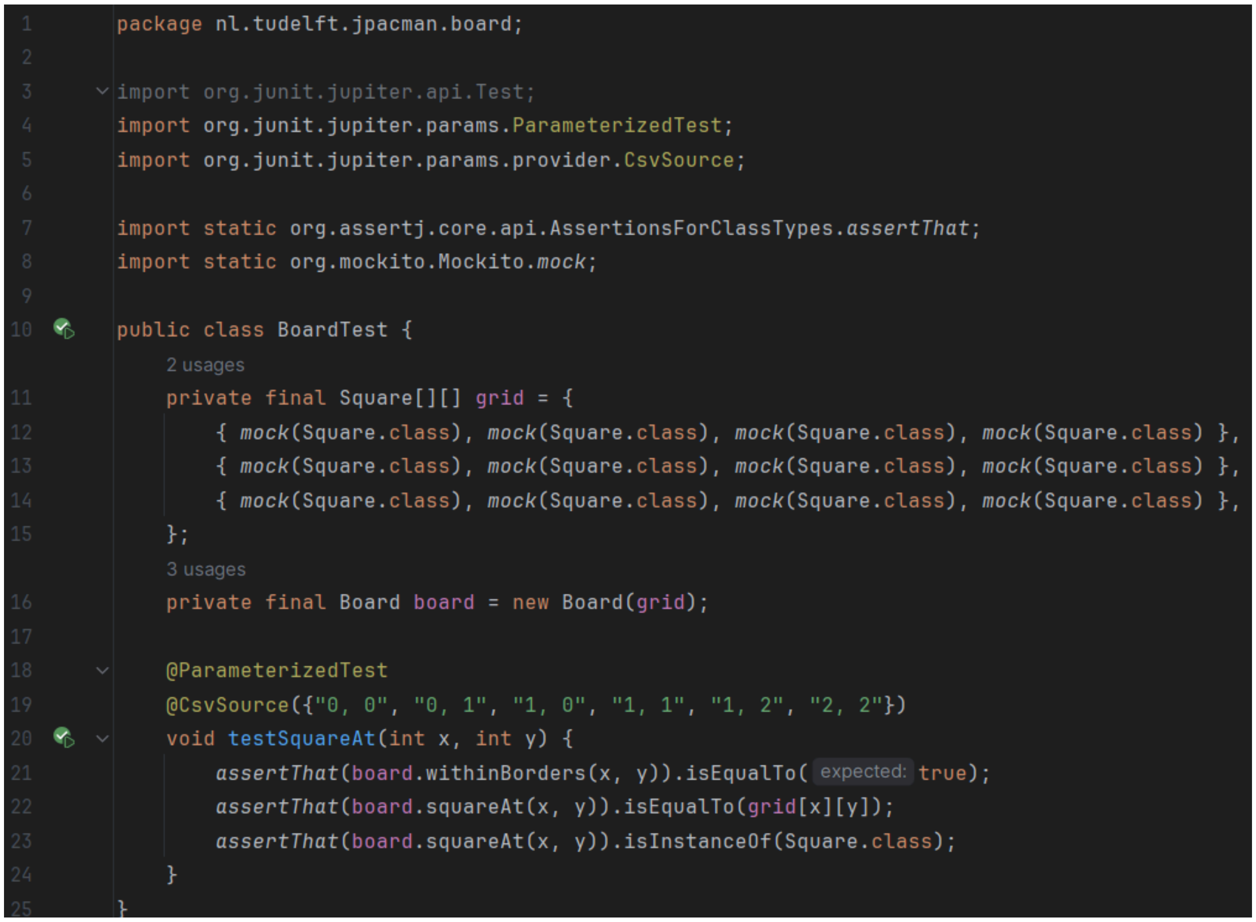
Coverage improved after testing remove():



Coverage   jpacman-master [test]

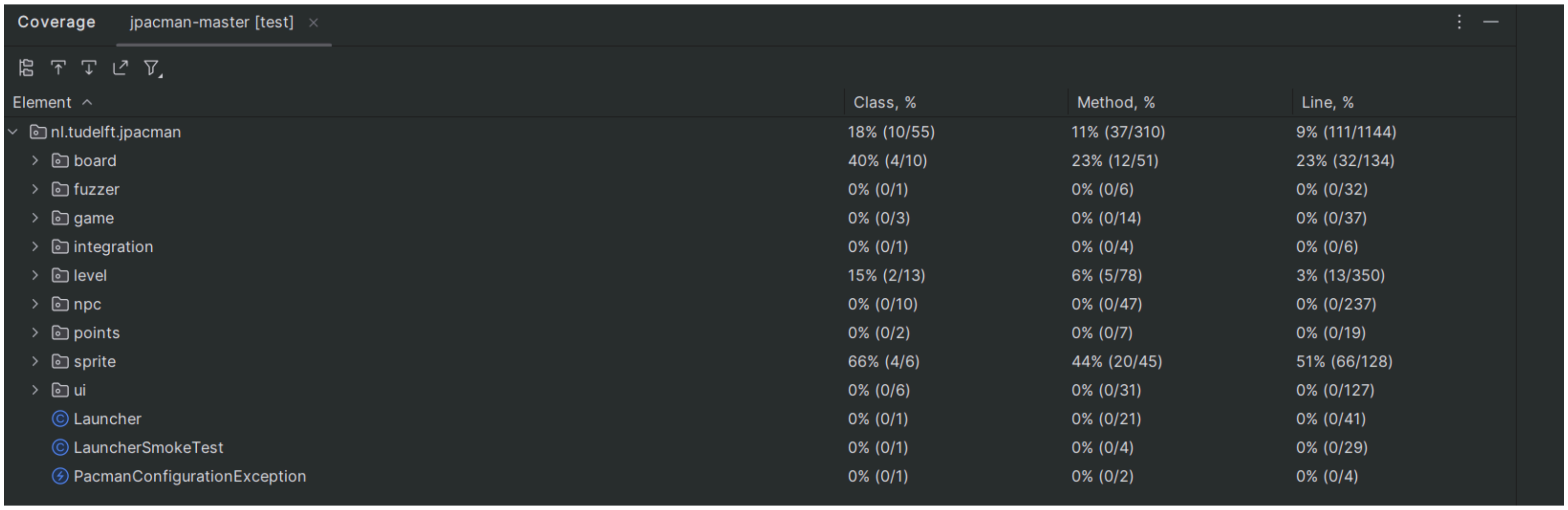
Element	Class, %	Method, %	Line, %
nl.tudelft.jpacman	21% (12/55)	14% (45/308)	11% (131/1145)
board	60% (6/10)	40% (20/49)	38% (52/135)
fuzzer	0% (0/1)	0% (0/6)	0% (0/32)
game	0% (0/3)	0% (0/14)	0% (0/37)
integration	0% (0/1)	0% (0/4)	0% (0/6)
level	15% (2/13)	6% (5/78)	3% (13/350)
npc	0% (0/10)	0% (0/47)	0% (0/237)
points	0% (0/2)	0% (0/7)	0% (0/19)
sprite	66% (4/6)	44% (20/45)	51% (66/128)
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)

I tested the squareAt() function in the Board class.



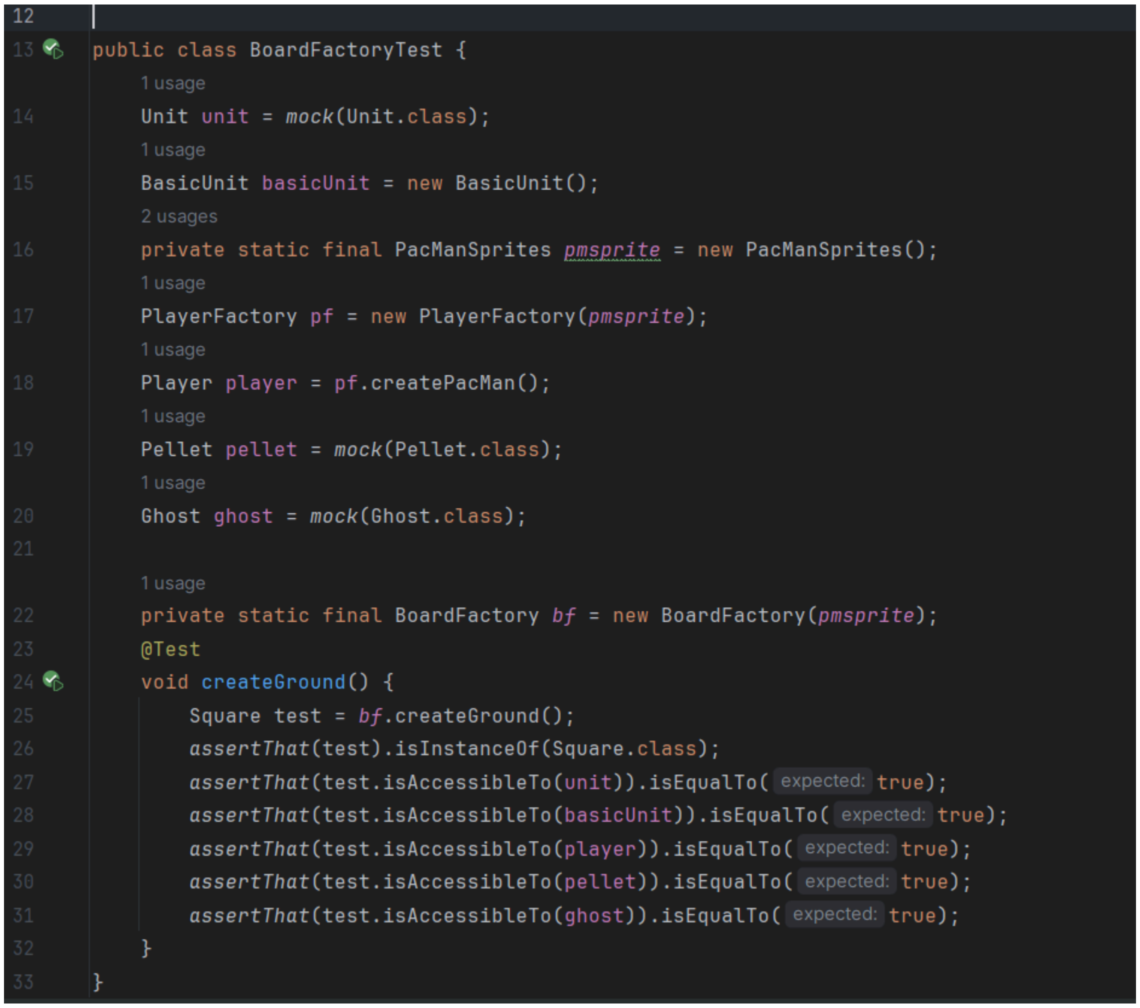
```
package nl.tudelft.jpacman.board;

import org.junit.jupiter.api.Test;
import org.junit.jupiter.params.ParameterizedTest;
import org.junit.jupiter.params.provider.CsvSource;

import static org.assertj.core.api.AssertionsForClassTypes.assertThat;
import static org.mockito.Mockito.mock;

public class BoardTest {
    2 usages
    private final Square[][] grid = {
        { mock(Square.class), mock(Square.class), mock(Square.class), mock(Square.class) },
        { mock(Square.class), mock(Square.class), mock(Square.class), mock(Square.class) },
        { mock(Square.class), mock(Square.class), mock(Square.class), mock(Square.class) },
    };
    3 usages
    private final Board board = new Board(grid);

    @ParameterizedTest
    @CsvSource({"0, 0", "0, 1", "1, 0", "1, 1", "1, 2", "2, 2"})
    void testSquareAt(int x, int y) {
        assertThat(board.withinBorders(x, y)).isEqualTo( expected: true);
        assertThat(board.squareAt(x, y)).isEqualTo(grid[x][y]);
        assertThat(board.squareAt(x, y)).isInstanceOf(Square.class);
    }
}
```

Coverage improved after testing squareAt():



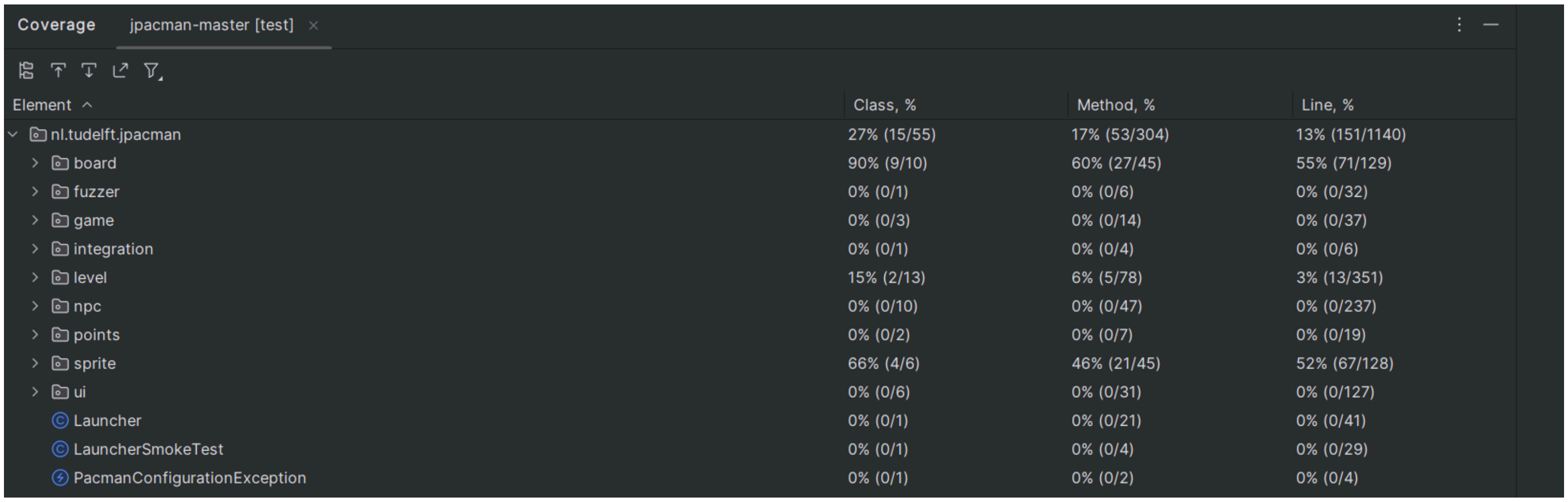
Element	Class, %	Method, %	Line, %
nl.tudelft.jpacman	18% (10/55)	11% (37/310)	9% (111/1144)
board	40% (4/10)	23% (12/51)	23% (32/134)
fuzzer	0% (0/1)	0% (0/6)	0% (0/32)
game	0% (0/3)	0% (0/14)	0% (0/37)
integration	0% (0/1)	0% (0/4)	0% (0/6)
level	15% (2/13)	6% (5/78)	3% (13/350)
npc	0% (0/10)	0% (0/47)	0% (0/237)
points	0% (0/2)	0% (0/7)	0% (0/19)
sprite	66% (4/6)	44% (20/45)	51% (66/128)
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)

I tested the createGround() function in the BoardFactory class.



```java

public class BoardFactoryTest {
    1 usage
    Unit unit = mock(Unit.class);
    1 usage
    BasicUnit basicUnit = new BasicUnit();
    2 usages
    private static final PacManSprites pmsprite = new PacManSprites();
    1 usage
    PlayerFactory pf = new PlayerFactory(pmsprite);
    1 usage
    Player player = pf.createPacMan();
    1 usage
    Pellet pellet = mock(Pellet.class);
    1 usage
    Ghost ghost = mock(Ghost.class);

    1 usage
    private static final BoardFactory bf = new BoardFactory(pmsprite);
    @Test
    void createGround() {
        Square test = bf.createGround();
        assertThat(test).isInstanceOf(Square.class);
        assertThat(test.isAccessibleTo(unit)).isEqualTo( expected: true);
        assertThat(test.isAccessibleTo(basicUnit)).isEqualTo( expected: true);
        assertThat(test.isAccessibleTo(player)).isEqualTo( expected: true);
        assertThat(test.isAccessibleTo(pellet)).isEqualTo( expected: true);
        assertThat(test.isAccessibleTo(ghost)).isEqualTo( expected: true);
    }
}
```

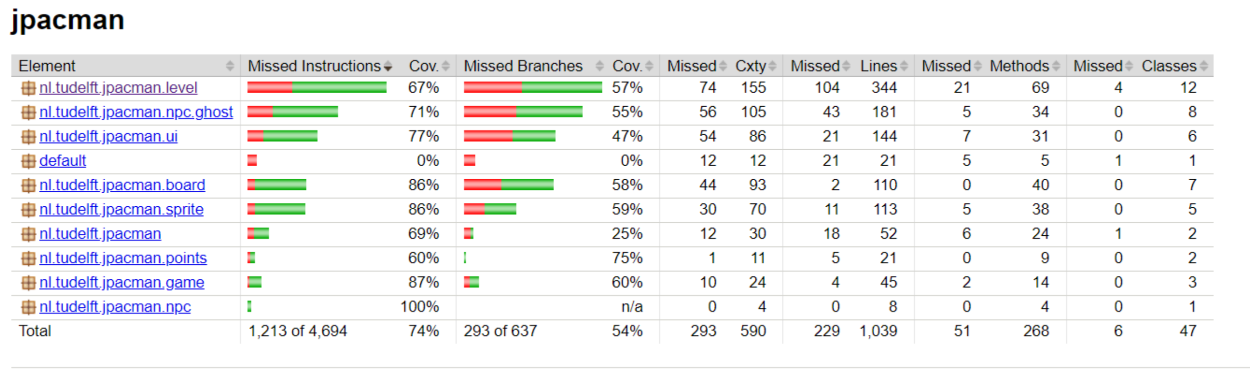
Coverage improved after testing createGround():



Coverage   jpacman-master [test]

Element	Class, %	Method, %	Line, %
nl.tudelft.jpacman	27% (15/55)	17% (53/304)	13% (151/1140)
board	90% (9/10)	60% (27/45)	55% (71/129)
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	15% (2/13)	6% (5/78)	3% (13/351)
npc	0% (0/10)	0% (0/47)	0% (0/237)
points	0% (0/2)	0% (0/7)	0% (0/19)
sprite	66% (4/6)	46% (21/45)	52% (67/128)
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)

Task 3:



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%	🟥🟩	58%	44	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	1,213 of 4,694	74%	293 of 637	54%	293	590	229	1,039	51	268	6	47

Are the coverage results from JaCoCo similar to the ones you got from IntelliJ in the last task? Why so or why not?

There are some similarities between JaCoCo and IntelliJ coverage, but it's mostly different, especially in terms of percentages. IntelliJ measures test coverage in class, methods, and lines. JaCoCo measures instructions and branches. JaCoCo also likely has more methods tests, which in line increases coverage for instructions and branches.

Did you find helpful the source code visualization from JaCoCo on uncovered branches?

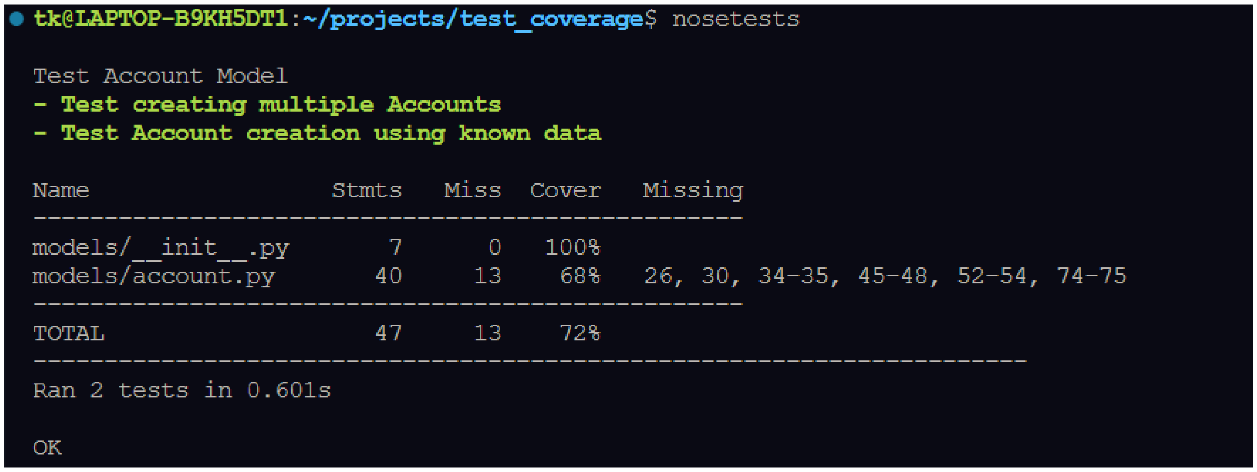
Yes, I found it helpful to see directly highlighted on the source which branches were covered by the tests and which ones weren't. It's a lot more intuitive in terms of visualizing. IntelliJ also has some color markers, but it isn't as obvious and informative. It feels like IntelliJ is more focused on the percentages, so JaCoCo's visualization was helpful.

Which visualization did you prefer and why? IntelliJ's coverage window or JaCoCo's report?

I think I prefer JaCoCo's report. I think the colors of the bars gives a quick and easy understanding of the current state of the tests. I also like the highlighting on the source code for branches. This more in-depth report is very useful which I prefer. That being said, I like the IntelliJ's coverage being built-in to the IDE, so there are positives to both, but I ultimately prefer JaCoCo.

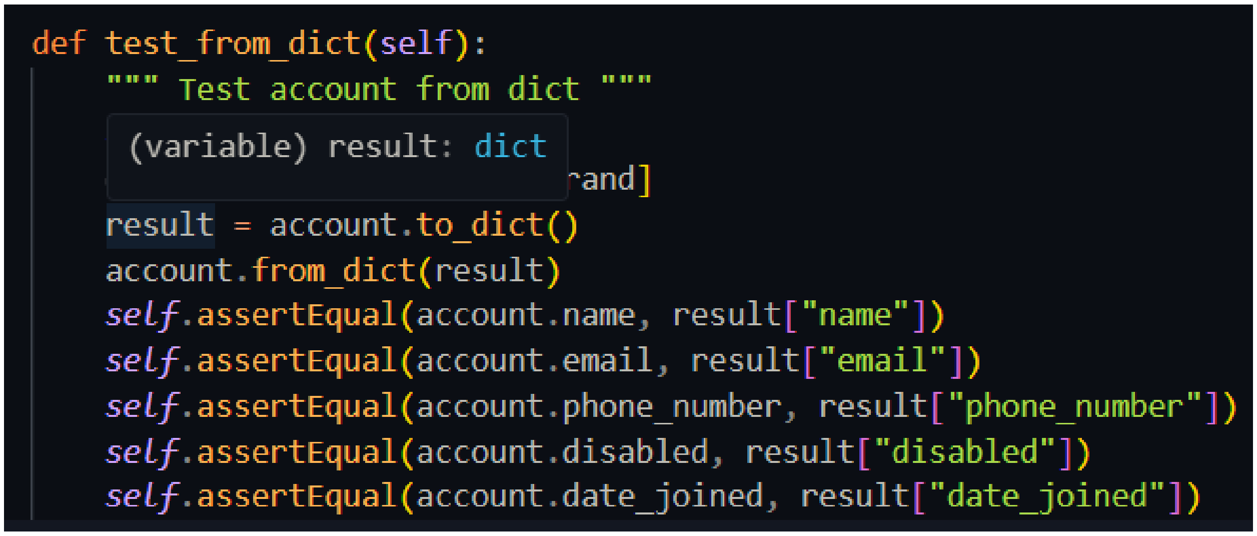
Task 4:

Start with coverage of 72%



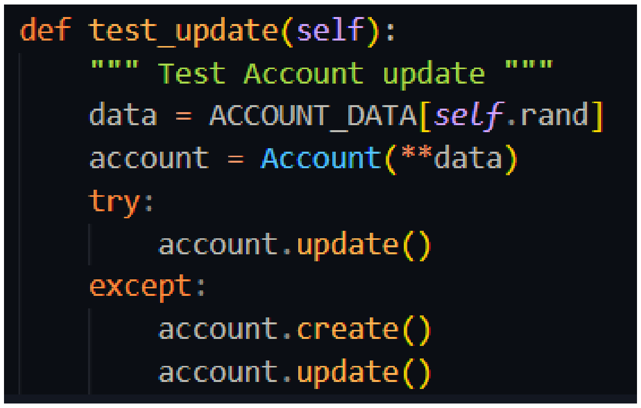
```
tk@LAPTOP-B9KH5DT1:~/projects/test_coverage$ nosetests

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 0.601s

OK
```

Written tests aside from test_repr (line 26) and test_to_dict (line 30) which were given to us:
test_from_dict to cover lines 34-35



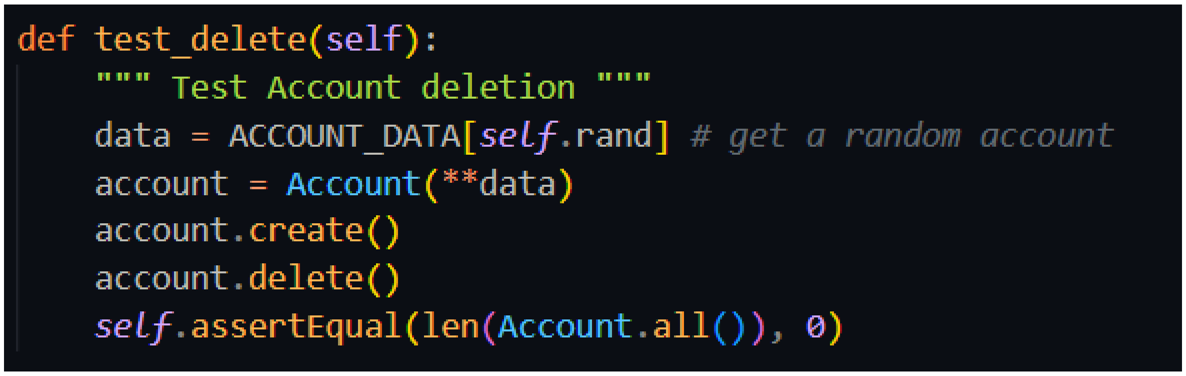
```python
def test_from_dict(self):
    """ Test account from dict """
    (variable) result: dict
                                 and]
    result = account.to_dict()
    account.from_dict(result)
    self.assertEqual(account.name, result["name"])
    self.assertEqual(account.email, result["email"])
    self.assertEqual(account.phone_number, result["phone_number"])
    self.assertEqual(account.disabled, result["disabled"])
    self.assertEqual(account.date_joined, result["date_joined"])
```

test_update to cover lines 45-48:



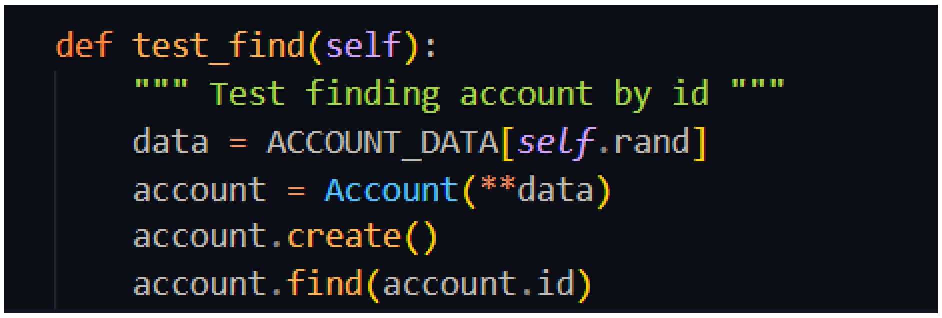
```python
def test_update(self):
    """ Test Account update """
    data = ACCOUNT_DATA[self.rand]
    account = Account(**data)
    try:
        account.update()
    except:
        account.create()
        account.update()
```

test_delete to cover lines 52-54:



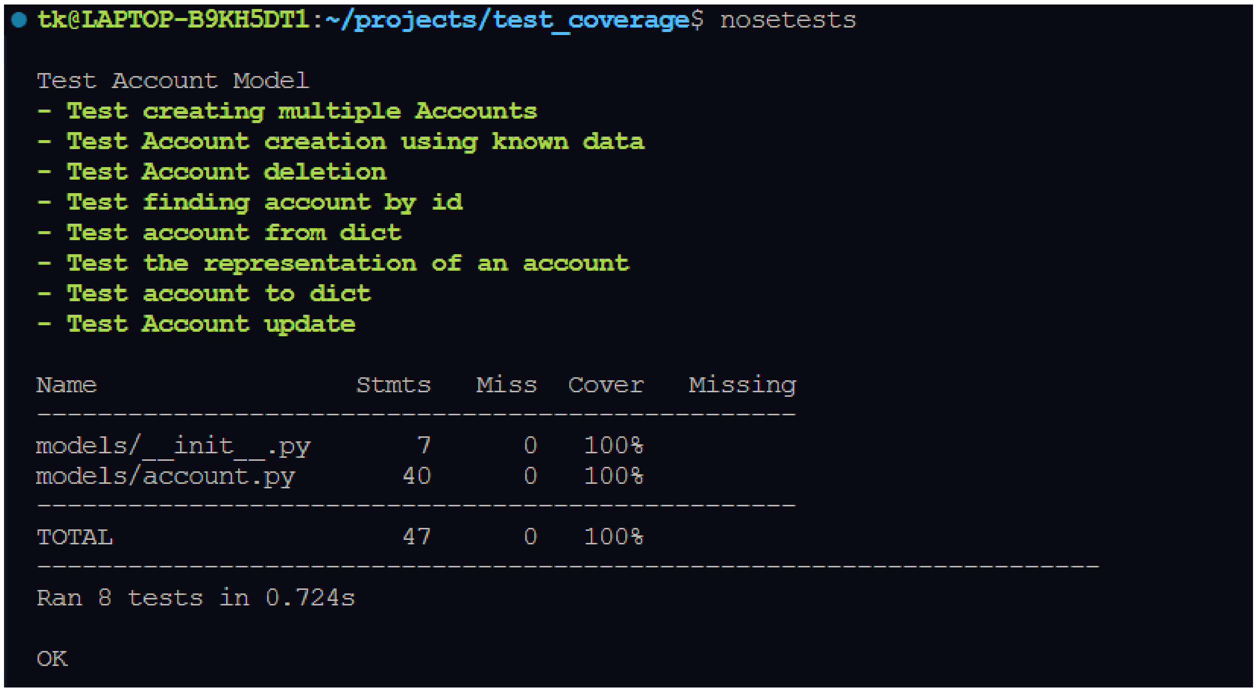
```python
def test_delete(self):
    """ Test Account deletion """
    data = ACCOUNT_DATA[self.rand] # get a random account
    account = Account(**data)
    account.create()
    account.delete()
    self.assertEqual(len(Account.all()), 0)
```

test_find to cover lines 74-75:



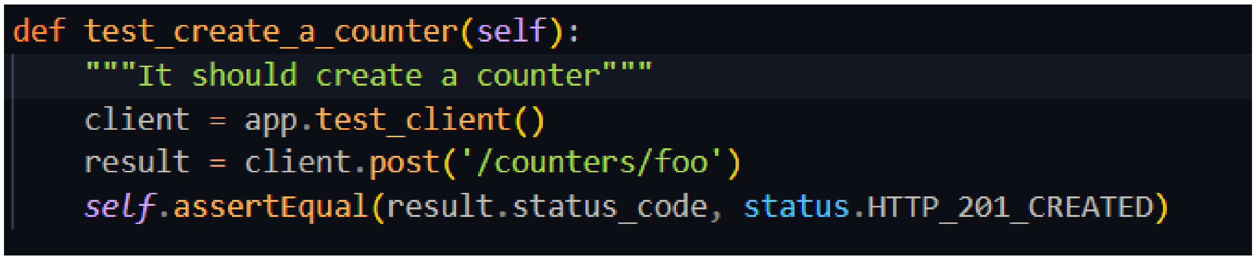
```python
def test_find(self):
    """ Test finding account by id """
    data = ACCOUNT_DATA[self.rand]
    account = Account(**data)
    account.create()
    account.find(account.id)
```

Final result: Coverage of 100%



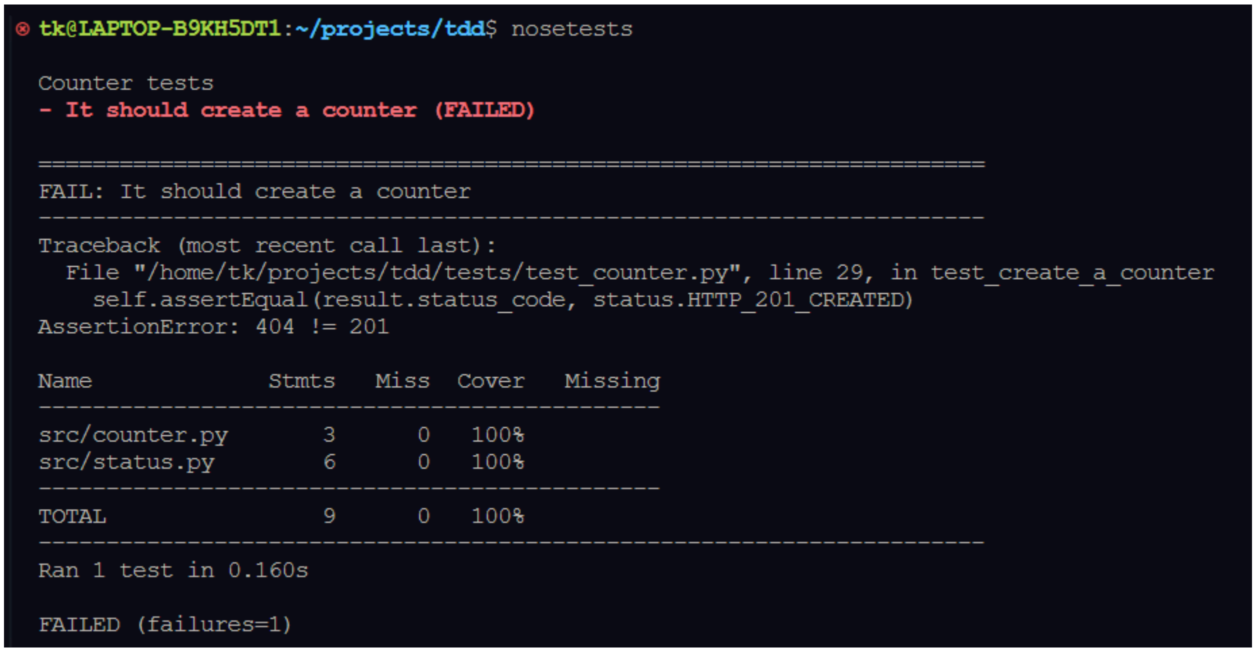
```
● tk@LAPTOP-B9KH5DT1:~/projects/test_coverage$ nosetests

  Test Account Model
  - Test creating multiple Accounts
  - Test Account creation using known data
  - Test Account deletion
  - Test finding account by id
  - Test account from dict
  - Test the representation of an account
  - Test account to dict
  - Test Account update

  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 0.724s

  OK
```

Task 5:
Test for creating counter:



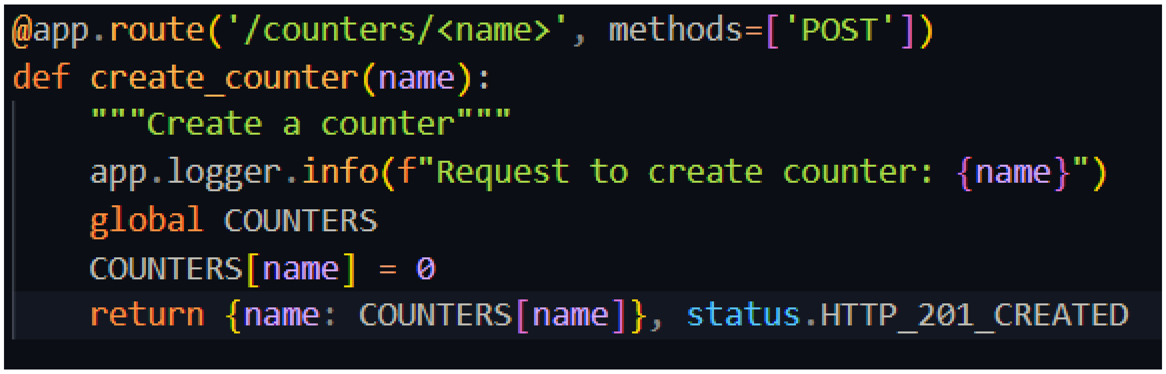
```python
def test_create_a_counter(self):
    """It should create a counter"""
    client = app.test_client()
    result = client.post('/counters/foo')
    self.assertEqual(result.status_code, status.HTTP_201_CREATED)
```

Leads to red phase: AssertionError 404 != 201



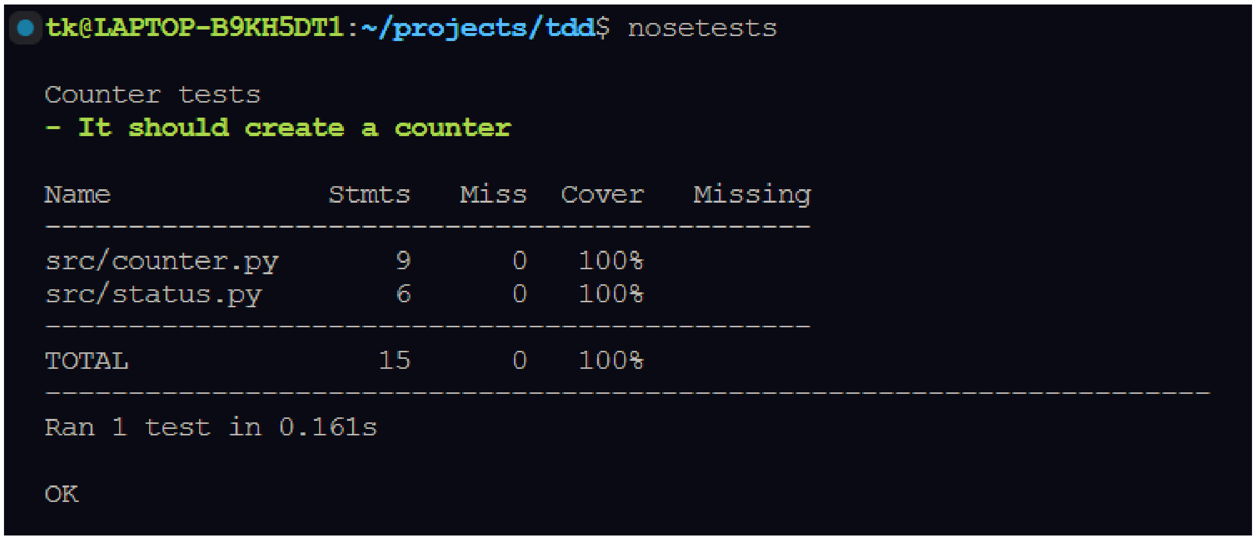
```
⊗ tk@LAPTOP-B9KH5DT1:~/projects/tdd$ nosetests

  Counter tests
  - It should create a counter (FAILED)

  ======================================================================
  FAIL: It should create a counter
  ----------------------------------------------------------------------
  Traceback (most recent call last):
    File "/home/tk/projects/tdd/tests/test_counter.py", line 29, in test_create_a_counter
      self.assertEqual(result.status_code, status.HTTP_201_CREATED)
  AssertionError: 404 != 201

  Name               Stmts   Miss  Cover   Missing
  ------------------------------------------------
  src/counter.py         3      0   100%
  src/status.py          6      0   100%
  ------------------------------------------------
  TOTAL                  9      0   100%
  ----------------------------------------------------------------------
  Ran 1 test in 0.160s

  FAILED (failures=1)
```

Function for creating counter:



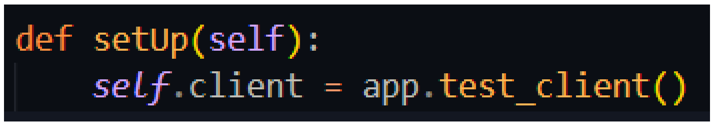
```python
@app.route('/counters/<name>', methods=['POST'])
def create_counter(name):
    """Create a counter"""
    app.logger.info(f"Request to create counter: {name}")
    global COUNTERS
    COUNTERS[name] = 0
    return {name: COUNTERS[name]}, status.HTTP_201_CREATED
```

Function resolves error, now in green phase:



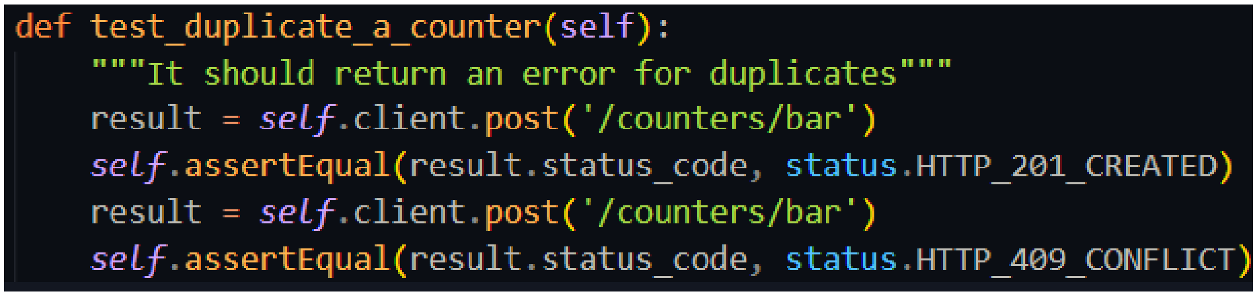
```
tk@LAPTOP-B9KH5DT1:~/projects/tdd$ nosetests

Counter tests
- It should create a counter

Name                Stmts   Miss  Cover   Missing
-------------------------------------------------
src/counter.py          9      0   100%
src/status.py           6      0   100%
-------------------------------------------------
TOTAL                  15      0   100%
----------------------------------------------------------------------
Ran 1 test in 0.161s

OK
```

Refactor phase:



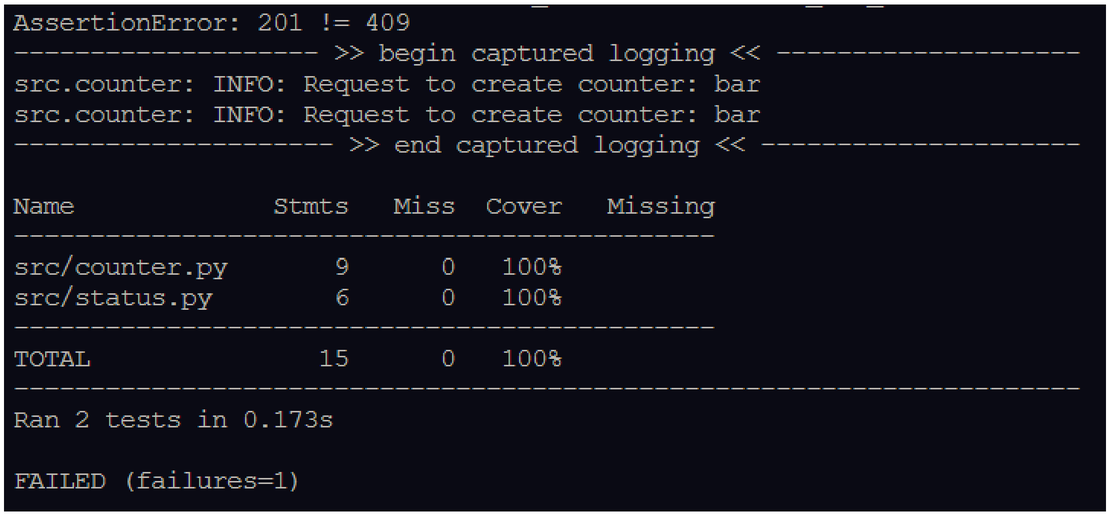
```python
def setUp(self):
    self.client = app.test_client()
```

Test for duplicate counters:



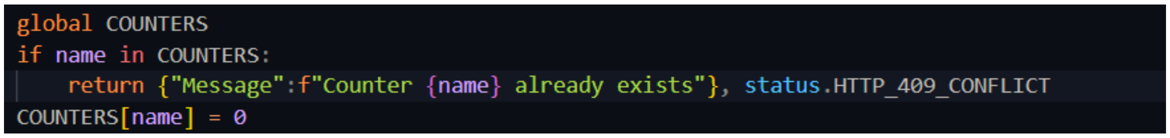
```python
def test_duplicate_a_counter(self):
    """It should return an error for duplicates"""
    result = self.client.post('/counters/bar')
    self.assertEqual(result.status_code, status.HTTP_201_CREATED)
    result = self.client.post('/counters/bar')
    self.assertEqual(result.status_code, status.HTTP_409_CONFLICT)
```

Leads to red phase: Assertion Error 201 != 409



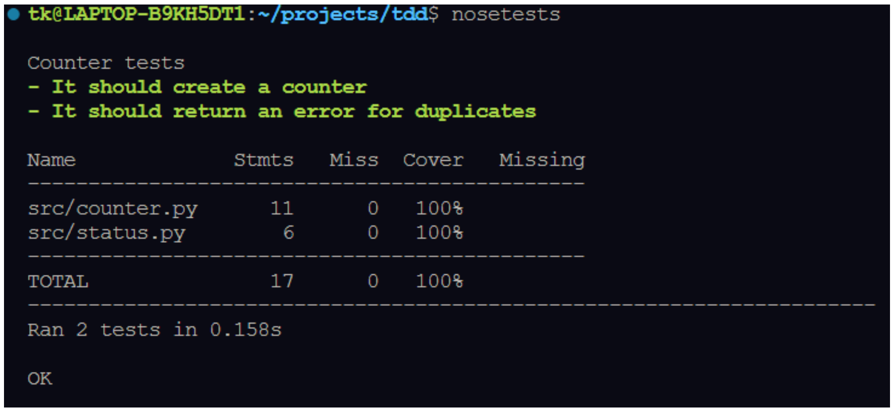
```
AssertionError: 201 != 409
-------------------- >> begin captured logging << --------------------
src.counter: INFO: Request to create counter: bar
src.counter: INFO: Request to create counter: bar
--------------------- >> end captured logging << ---------------------

Name                Stmts   Miss  Cover   Missing
-------------------------------------------------
src/counter.py          9      0   100%
src/status.py           6      0   100%
-------------------------------------------------
TOTAL                  15      0   100%
----------------------------------------------------------------------
Ran 2 tests in 0.173s

FAILED (failures=1)
```

Refactor:



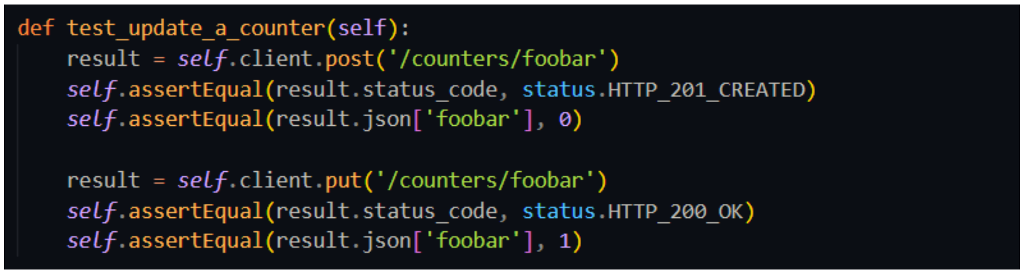
```
global COUNTERS
if name in COUNTERS:
    return {"Message":f"Counter {name} already exists"}, status.HTTP_409_CONFLICT
COUNTERS[name] = 0
```

Function resolves error, now in green phase:



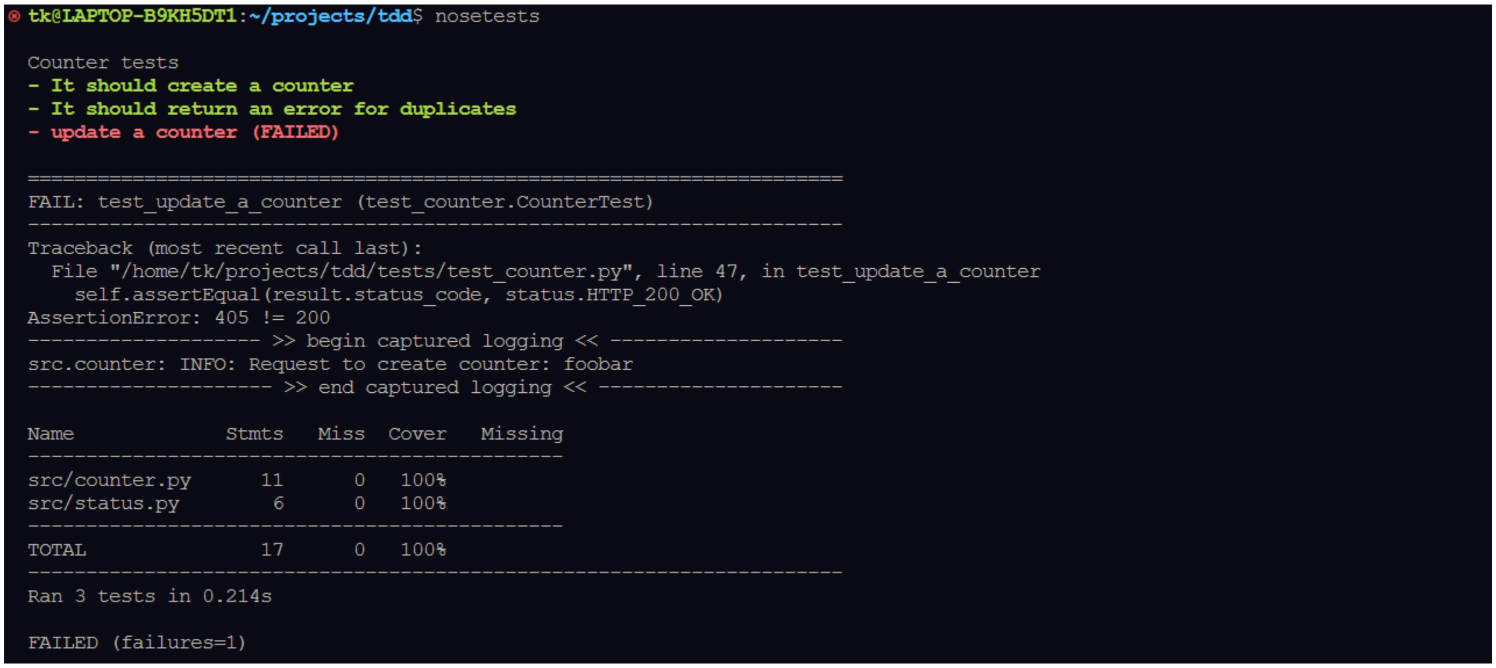
```
tk@LAPTOP-B9KH5DT1:~/projects/tdd$ nosetests

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

Name              Stmts   Miss  Cover   Missing
-----------------------------------------------
src/counter.py       11      0   100%
src/status.py         6      0   100%
-----------------------------------------------
TOTAL                17      0   100%
----------------------------------------------------------------------
Ran 2 tests in 0.158s

OK
```

Test for updating counter:



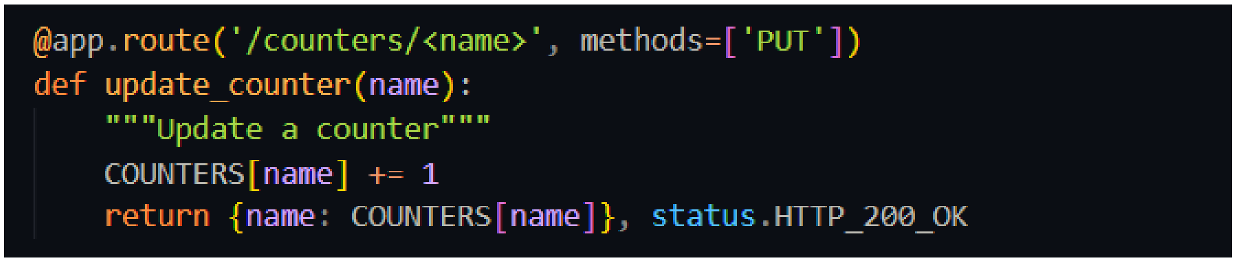
```
def test_update_a_counter(self):
    result = self.client.post('/counters/foobar')
    self.assertEqual(result.status_code, status.HTTP_201_CREATED)
    self.assertEqual(result.json['foobar'], 0)

    result = self.client.put('/counters/foobar')
    self.assertEqual(result.status_code, status.HTTP_200_OK)
    self.assertEqual(result.json['foobar'], 1)
```

Leads to red phase: AssertionError 405 != 200



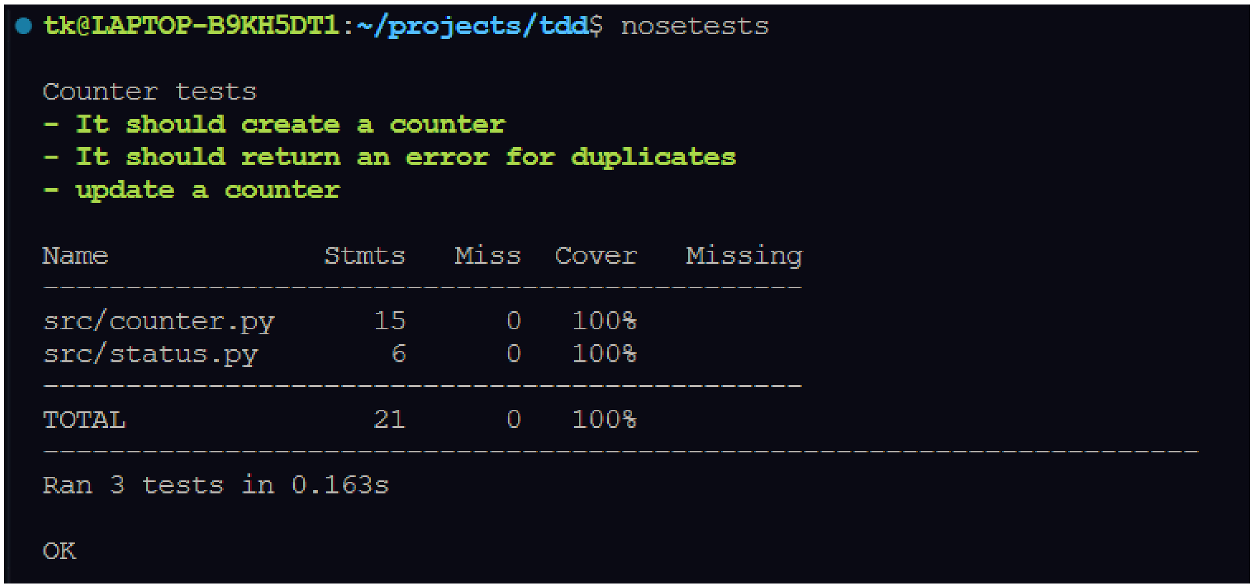
```
tk@LAPTOP-B9KH5DT1:~/projects/tdd$ nosetests

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

======================================================================
FAIL: test_update_a_counter (test_counter.CounterTest)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/tk/projects/tdd/tests/test_counter.py", line 47, in test_update_a_counter
    self.assertEqual(result.status_code, status.HTTP_200_OK)
AssertionError: 405 != 200
-------------------- >> begin captured logging << --------------------
src.counter: INFO: Request to create counter: foobar
--------------------- >> 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 0.214s

FAILED (failures=1)
```

Function for updating counter:



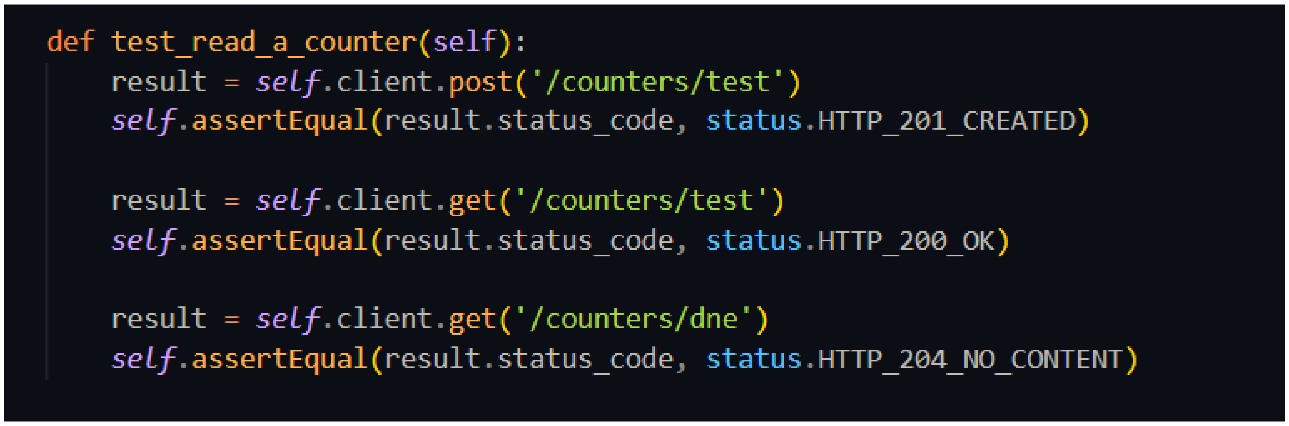
```
@app.route('/counters/<name>', methods=['PUT'])
def update_counter(name):
    """Update a counter"""
    COUNTERS[name] += 1
    return {name: COUNTERS[name]}, status.HTTP_200_OK
```

Function resolves error, now in green phase:



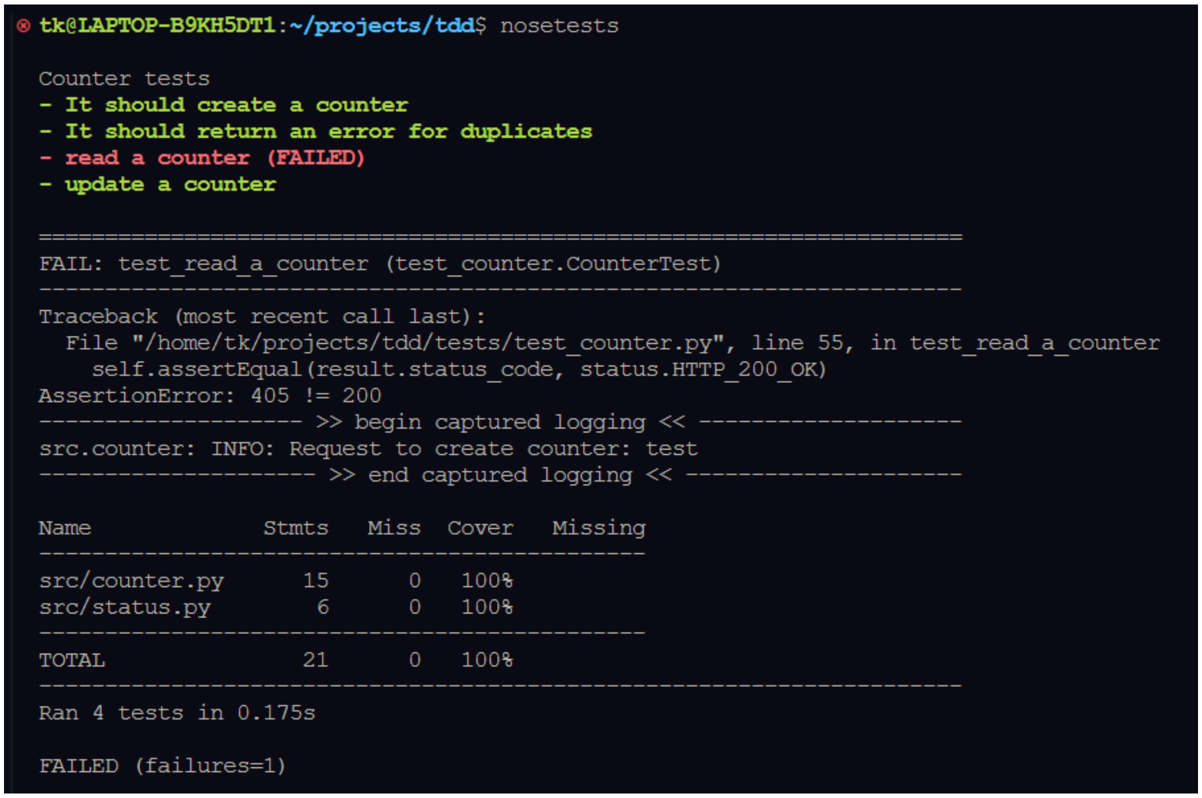
```
● tk@LAPTOP-B9KH5DT1:~/projects/tdd$ nosetests

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

  Name                Stmts   Miss  Cover   Missing
  ----------------------------------------------
  src/counter.py         15      0   100%
  src/status.py           6      0   100%
  ----------------------------------------------
  TOTAL                  21      0   100%
  ----------------------------------------------------------------------
  Ran 3 tests in 0.163s

  OK
```

Test for reading counter:



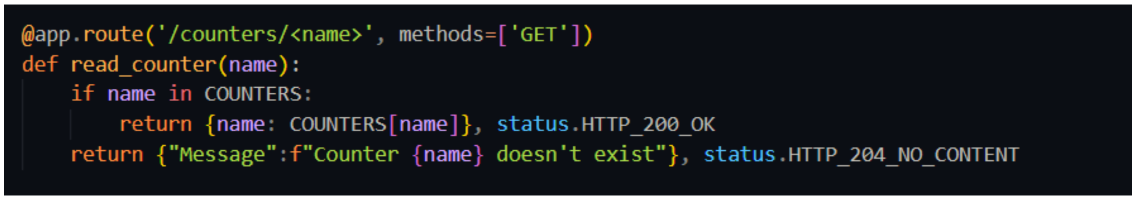
```
def test_read_a_counter(self):
    result = self.client.post('/counters/test')
    self.assertEqual(result.status_code, status.HTTP_201_CREATED)

    result = self.client.get('/counters/test')
    self.assertEqual(result.status_code, status.HTTP_200_OK)

    result = self.client.get('/counters/dne')
    self.assertEqual(result.status_code, status.HTTP_204_NO_CONTENT)
```

Leads to red phase: AssertionError: 405 != 200



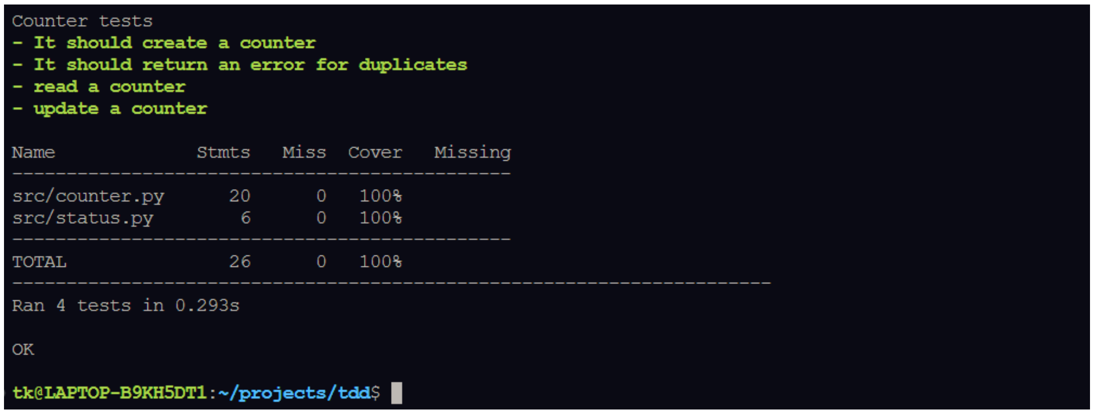
```
⊗ tk@LAPTOP-B9KH5DT1:~/projects/tdd$ nosetests

  Counter tests
  - It should create a counter
  - It should return an error for duplicates
  - read a counter (FAILED)
  - update a counter

  ======================================================================
  FAIL: test_read_a_counter (test_counter.CounterTest)
  ----------------------------------------------------------------------
  Traceback (most recent call last):
    File "/home/tk/projects/tdd/tests/test_counter.py", line 55, in test_read_a_counter
      self.assertEqual(result.status_code, status.HTTP_200_OK)
  AssertionError: 405 != 200
  -------------------- >> begin captured logging << --------------------
  src.counter: INFO: Request to create counter: test
  --------------------- >> end captured logging << ---------------------

  Name              Stmts   Miss  Cover   Missing
  -----------------------------------------------
  src/counter.py       15      0   100%
  src/status.py         6      0   100%
  -----------------------------------------------
  TOTAL                21      0   100%
  ----------------------------------------------------------------------
  Ran 4 tests in 0.175s

  FAILED (failures=1)
```

Function for reading counter:



```
@app.route('/counters/<name>', methods=['GET'])
def read_counter(name):
    if name in COUNTERS:
        return {name: COUNTERS[name]}, status.HTTP_200_OK
    return {"Message":f"Counter {name} doesn't exist"}, status.HTTP_204_NO_CONTENT
```

Function resolves error, now in green phase:



```
Counter tests
- It should create a counter
- It should return an error for duplicates
- read a counter
- update a counter

Name              Stmts   Miss  Cover   Missing
-----------------------------------------------
src/counter.py       20      0   100%
src/status.py         6      0   100%
-----------------------------------------------
TOTAL                26      0   100%
----------------------------------------------------------------------
Ran 4 tests in 0.293s

OK

tk@LAPTOP-B9KH5DT1:~/projects/tdd$ 
```

Now, we have tests and functions for creating a counter, duplicate counters, updating counters, and reading counters. We have 100% coverage for counter.py.