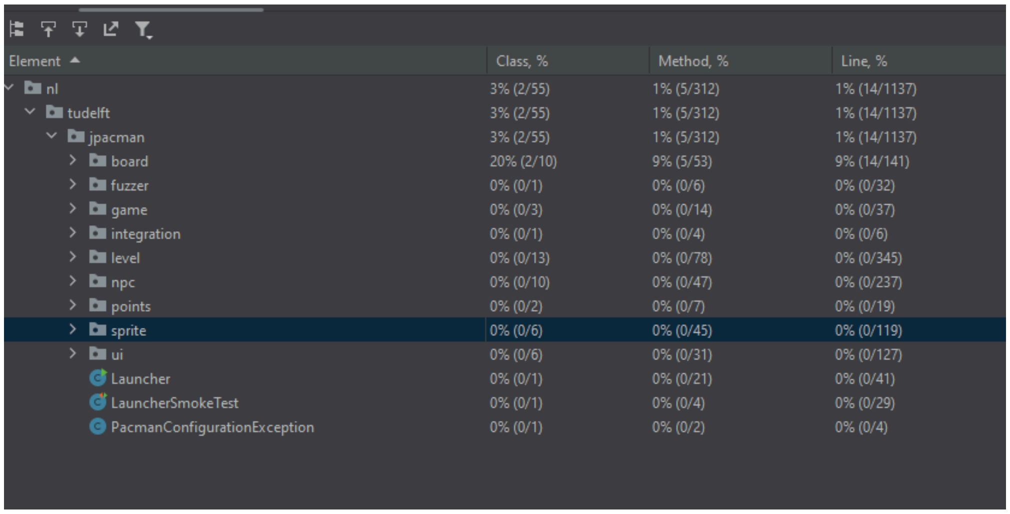
Lab 2 Report

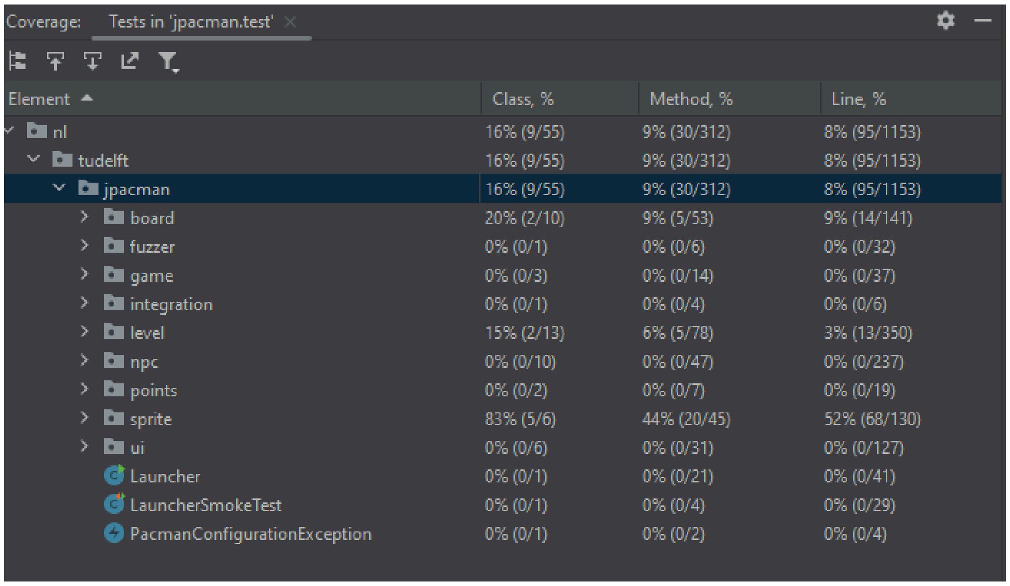
Task 1



Element	Class, %	Method, %	Line, %
nl	3% (2/55)	1% (5/312)	1% (14/1137)
tudelft	3% (2/55)	1% (5/312)	1% (14/1137)
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)

- The original converge of the test files, referencing mostly DirectionTest test for Direction in the board package.

Task 2



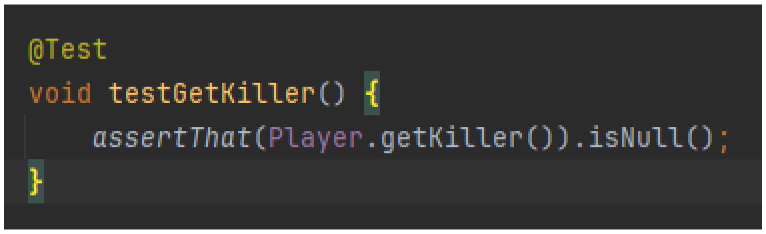
Coverage: Tests in 'jpacman.test'

Element	Class, %	Method, %	Line, %
nl	16% (9/55)	9% (30/312)	8% (95/1153)
tudelft	16% (9/55)	9% (30/312)	8% (95/1153)
jpacman	16% (9/55)	9% (30/312)	8% (95/1153)
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	83% (5/6)	44% (20/45)	52% (68/130)
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)

- Updated converge after isAlive test to player, showing an increase in the level coverage, with increases in the sprite package. Not entirely sure why, but could be due to references being made in the isAlive test. Similarity with player now covering more than the isAlive method is the method coverage may also be due to something similar.

Task 2.1

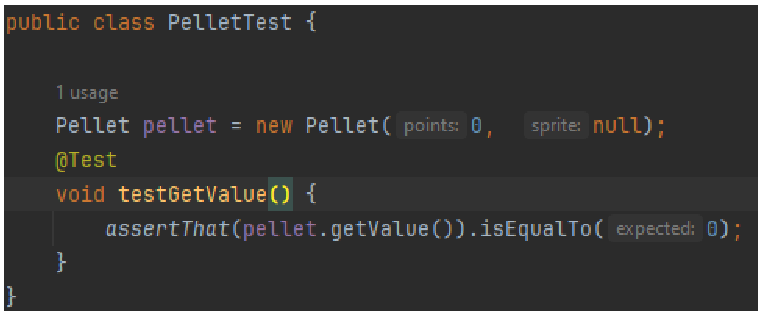
Method 1



```
@Test
void testGetKiller() {
    assertThat(Player.getKiller()).isNull();
}
```

- A test for getKiller in player using the player reference created earlier in the isAlive test. Checking if it returns anything that's not null, meaning either of the sprites.

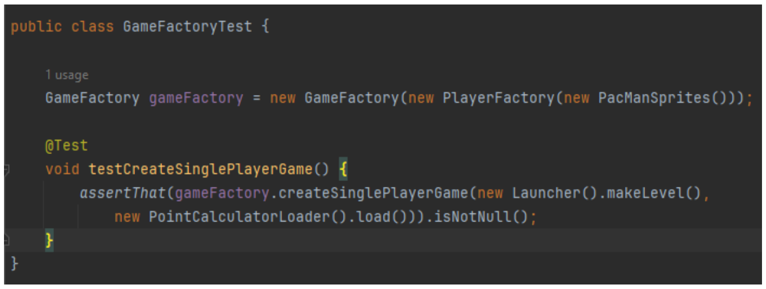
Method 2



```
public class PelletTest {

    1 usage
    Pellet pellet = new Pellet( points: 0,  sprite: null);
    @Test
    void testGetValue() {
        assertThat(pellet.getValue()).isEqualTo( expected: 0);
    }
}
```

- A test for getValue in the Pellet class, sprite is set to null as no use is required to obtain the values possible in getValue.

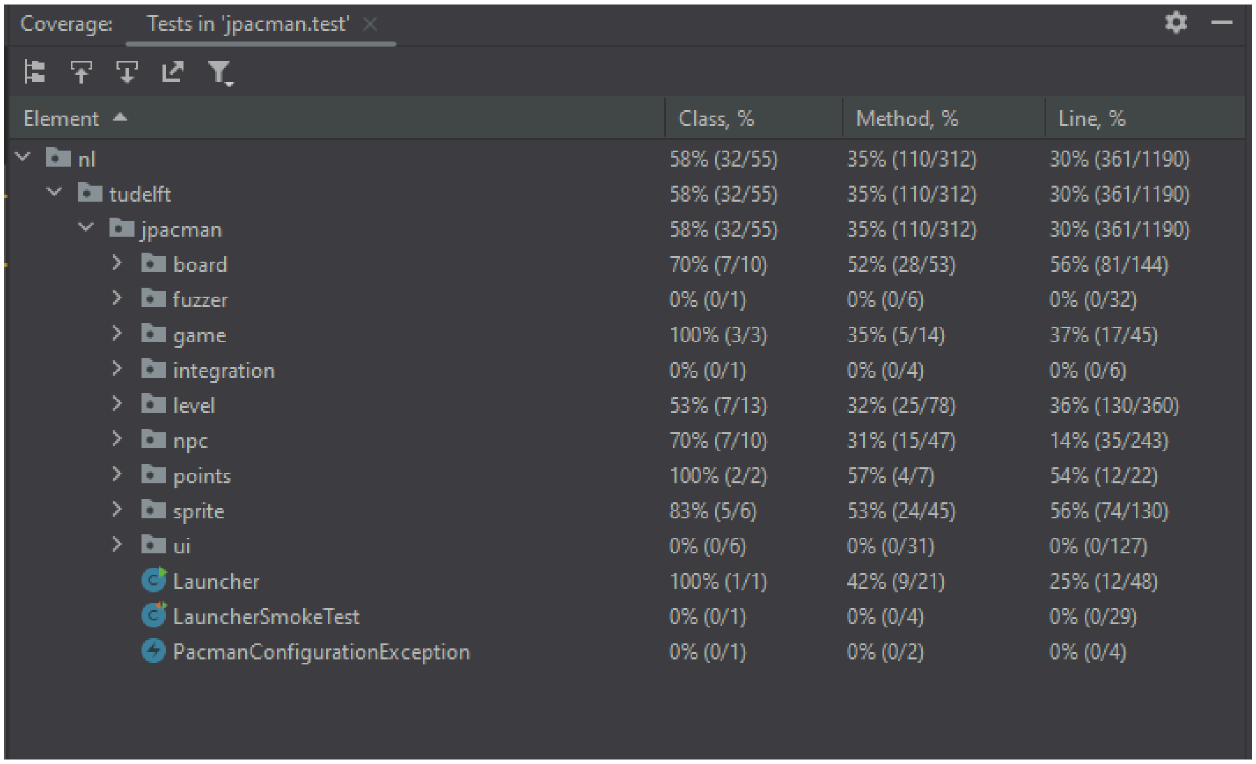
Method 3



```
public class GameFactoryTest {

    1 usage
    GameFactory gameFactory = new GameFactory(new PlayerFactory(new PacManSprites()));

    @Test
    void testCreateSinglePlayerGame() {
        assertThat(gameFactory.createSinglePlayerGame(new Launcher().makeLevel(),
            new PointCalculatorLoader().load())).isNotNull();
    }
}
```

- A test for createSinglePlayerGame in the GameFactory class, takes several references to both GameFactory and the classes Launcher, PointCalculatorLoader. Checking if it returns something besides null.

Coverage



Coverage: Tests in 'jpacman.test'

Element	Class, %	Method, %	Line, %
nl	58% (32/55)	35% (110/312)	30% (361/1190)
tudelft	58% (32/55)	35% (110/312)	30% (361/1190)
jpacman	58% (32/55)	35% (110/312)	30% (361/1190)
board	70% (7/10)	52% (28/53)	56% (81/144)
fuzzer	0% (0/1)	0% (0/6)	0% (0/32)
game	100% (3/3)	35% (5/14)	37% (17/45)
integration	0% (0/1)	0% (0/4)	0% (0/6)
level	53% (7/13)	32% (25/78)	36% (130/360)
npc	70% (7/10)	31% (15/47)	14% (35/243)
points	100% (2/2)	57% (4/7)	54% (12/22)
sprite	83% (5/6)	53% (24/45)	56% (74/130)
ui	0% (0/6)	0% (0/31)	0% (0/127)
Launcher	100% (1/1)	42% (9/21)	25% (12/48)
LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)
PacmanConfigurationException	0% (0/1)	0% (0/2)	0% (0/4)

- This created a notable increase in various parts of the coverage. Obvious ones like, level and game. However it increased coverage in Launcher for instance which was not directly tested. Which is most definitely now due to the fact that launcher is referenced and called in the GameFactory test. Also having used constructors for player, pellet and gameFactory in the tests, their coverage increased similarly.

Task 3

Questions

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

- They seem similar by viewing the percentage and missed lines and missed methods sections. Both Intellij and JaCoCo seem to agree on things like reference calls counting for coverage and what lines were missed. JaCoCo just seems to give actual specifics for missed areas.

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

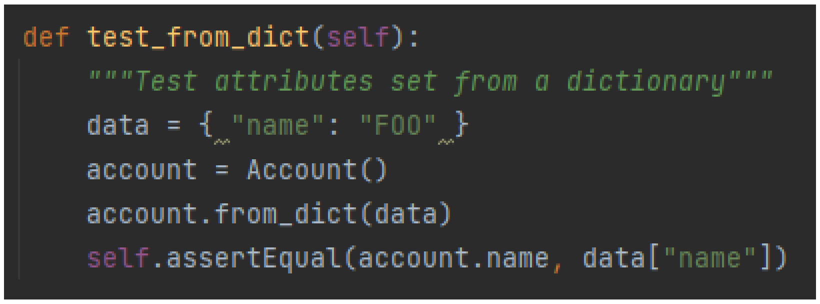
- Very helpful, clearly gives away areas that still need to be asserted and tested. Showing exactly what branches are not being tested even if parts are.

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

- I prefer JaCoCo's report as it gives much more in depth info on the actual branches missed rather than showing the number of missed lines. Also the bar visuals on coverage in JaCoCo are much easier to analyze at first glance than Intellij's.

Task 4

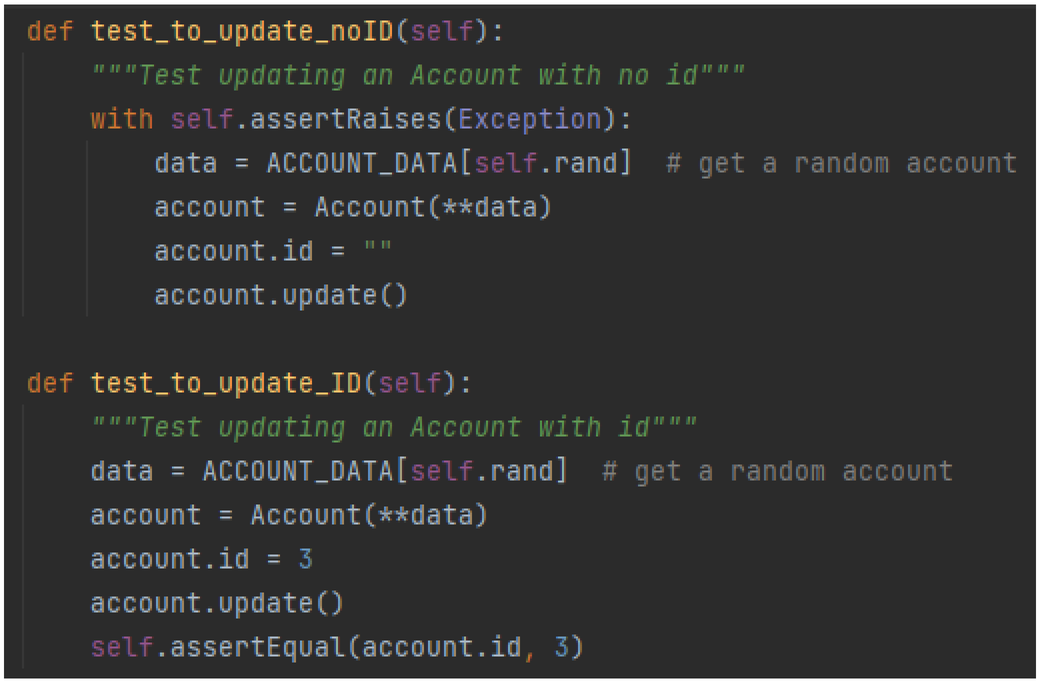
Lines 34-35



```python
def test_from_dict(self):
    """Test attributes set from a dictionary"""
    data = { "name": "FOO" }
    account = Account()
    account.from_dict(data)
    self.assertEqual(account.name, data["name"])
```

- Creating a new dict with just a name, then creating an empty account then using from_dict method to push the data into the account. Testing by checking the new account's name matches the original name input.

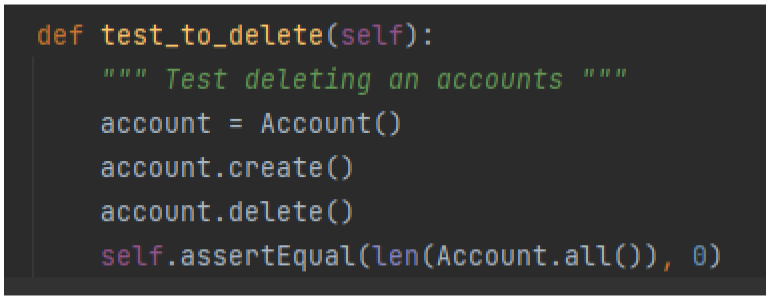
Lines 45-48



```python
def test_to_update_noID(self):
    """Test updating an Account with no id"""
    with self.assertRaises(Exception):
        data = ACCOUNT_DATA[self.rand]  # get a random account
        account = Account(**data)
        account.id = ""
        account.update()

def test_to_update_ID(self):
    """Test updating an Account with id"""
    data = ACCOUNT_DATA[self.rand]  # get a random account
    account = Account(**data)
    account.id = 3
    account.update()
    self.assertEqual(account.id, 3)
```

- Update testing for account required two separate tests for the exception that id could be null. No id testing asserts a raise when a random account is updated with no id. While with an id the assertion checks that well the id is updated and that it matches what was originally set as the id.

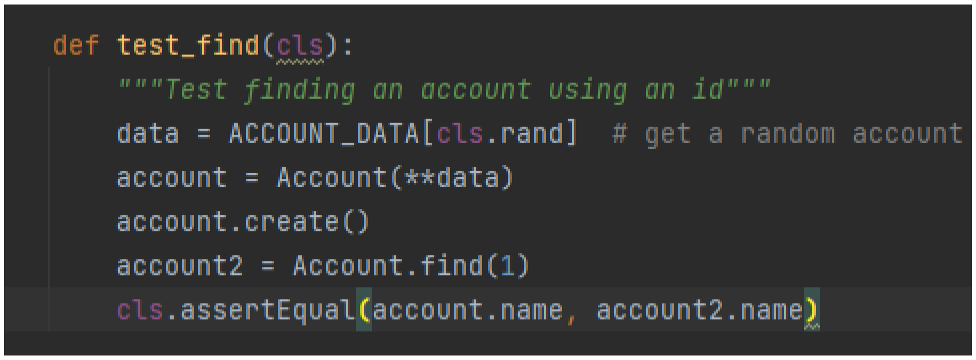
Lines 52-54



```python
def test_to_delete(self):
    """ Test deleting an accounts """
    account = Account()
    account.create()
    account.delete()
    self.assertEqual(len(Account.all()), 0)
```

- First adding an account (empty or not) to the account list using the create method and then deleting it. Before asserting that the account list is empty, meaning the created account has been deleted and the delete method is working.

Lines 74-75

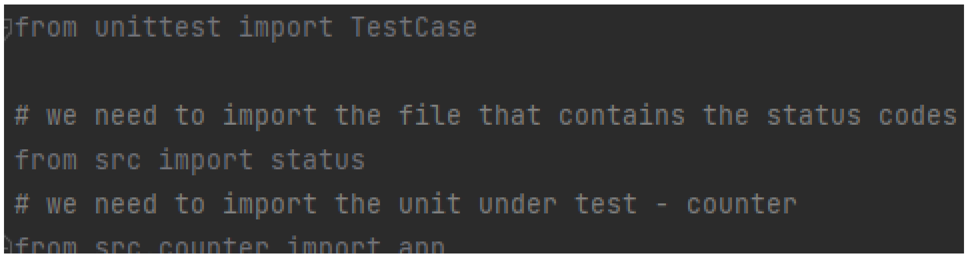


```
def test_find(cls):
    """Test finding an account using an id"""
    data = ACCOUNT_DATA[cls.rand]  # get a random account
    account = Account(**data)
    account.create()
    account2 = Account.find(1)
    cls.assertEqual(account.name, account2.name)
```

- To test, two accounts need to be created, one literally created by the create method. While the other took on the account with the actual storage area from the created method. Testing both's names (should be the same) to make sure that the account was successfully found after adding it to the storage area.

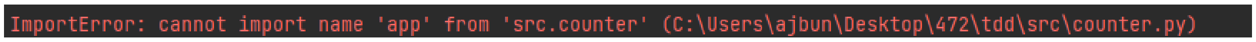
Task 4

After importing the initial imports, the program got an import error as the app component did not exist in the counter file after creating the counter py file.



```
from unittest import TestCase

# we need to import the file that contains the status codes
from src import status
# we need to import the unit under test - counter
from src.counter import app
```

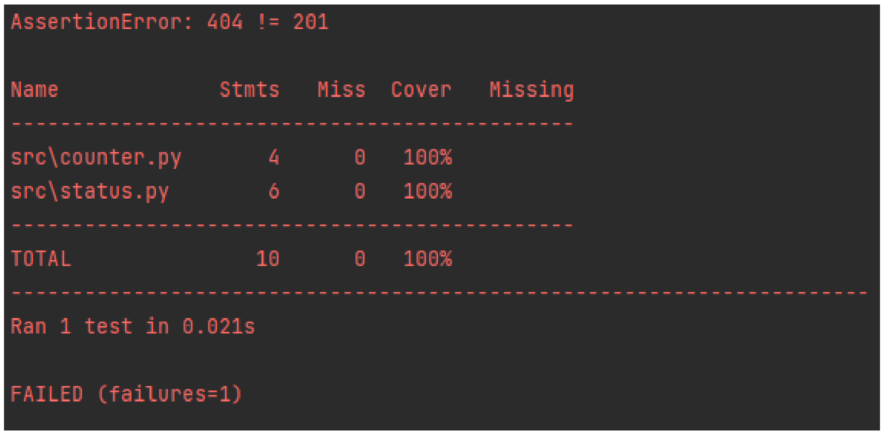
Error reported:



```
ImportError: cannot import name 'app' from 'src.counter' (C:\Users\ajbun\Desktop\472\tdd\src\counter.py)
```

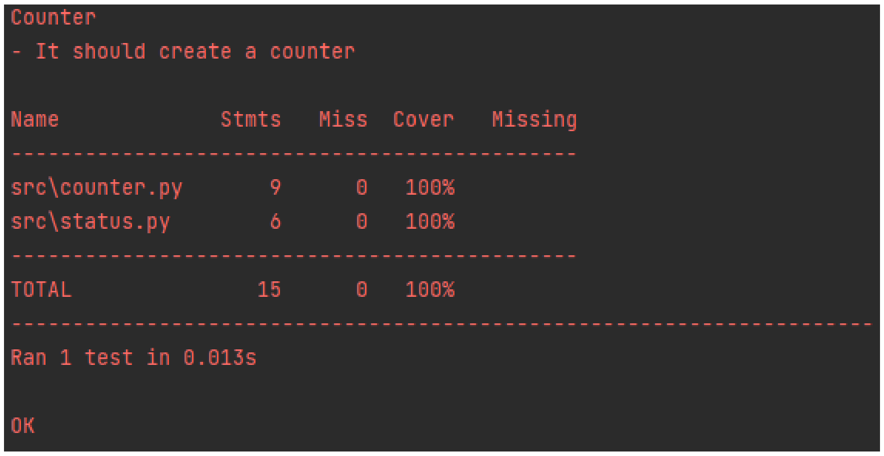
After adding the first test case, the test create a counter. I received the assertion error. As no actual method existed to test the create a counter test cas

Error reported:



```
AssertionError: 404 != 201

Name             Stmts   Miss  Cover   Missing
----------------------------------------------
src\counter.py       4      0   100%
src\status.py        6      0   100%
----------------------------------------------
TOTAL               10      0   100%
----------------------------------------------------------------------
Ran 1 test in 0.021s

FAILED (failures=1)
```

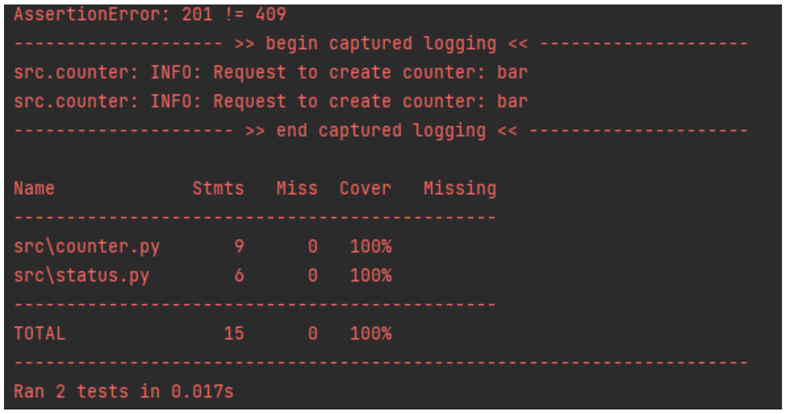
After placing in the code in counter.py related to create a counter:



```
Counter
- 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 0.013s

OK
```

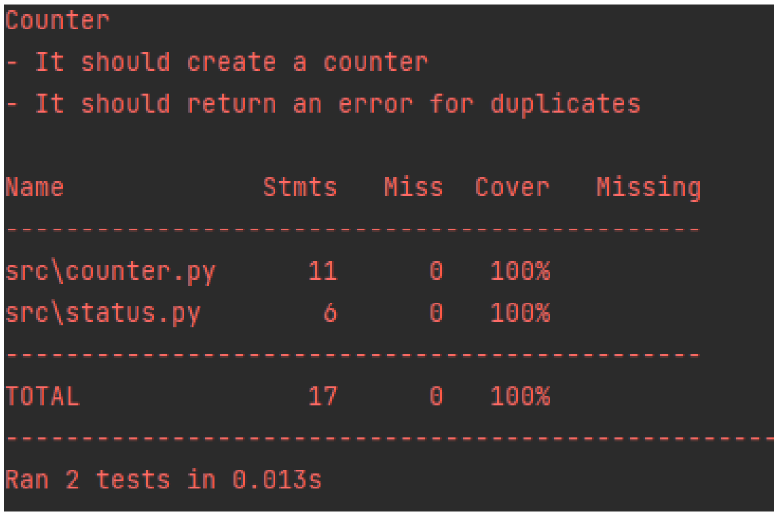
Next I placed in the update test method, this once again created an assertion error. This time because the error didn't exist for when multiple counters of the same name are created. This was fixed with the if statement refactor.

Error reported:



```
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 0.017s
```

After fixing:

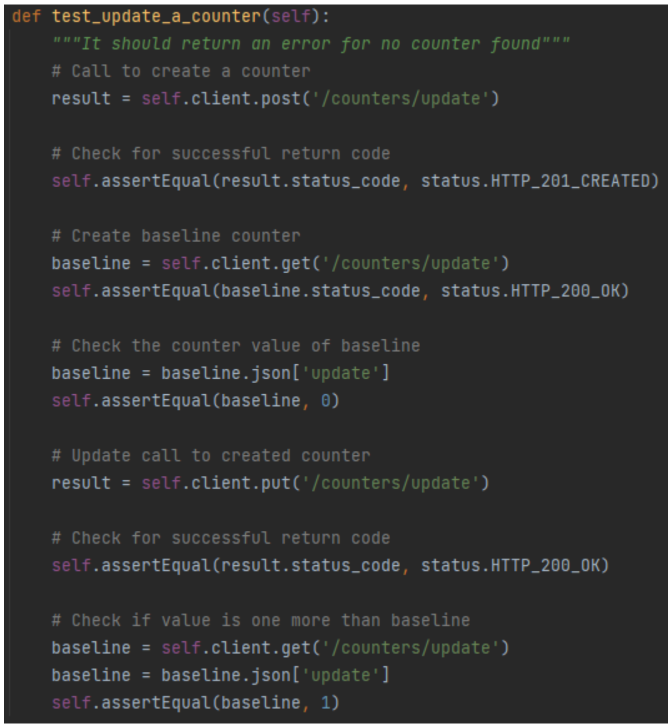


```
Counter
- It should create a counter
- It should return an error for duplicates

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 0.013s
```

My tasks

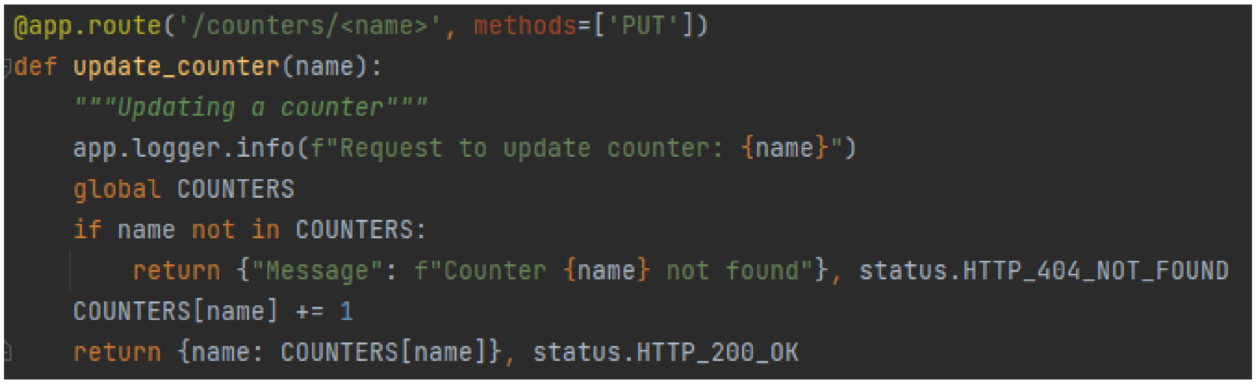
Update

Following along the tasks for the update test case, checking to make sure every case was followed. Notably using the json data field of the counter to check that the counter was actually updated.



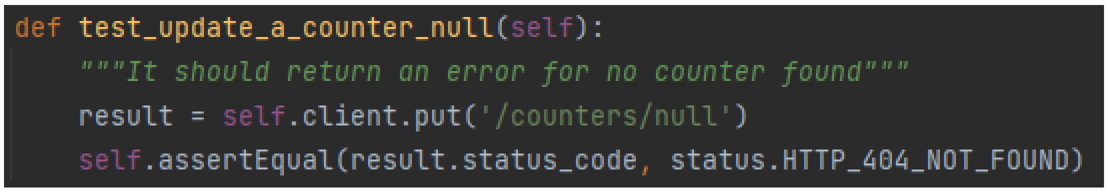
```python
def test_update_a_counter(self):
    """It should return an error for no counter found"""
    # Call to create a counter
    result = self.client.post('/counters/update')

    # Check for successful return code
    self.assertEqual(result.status_code, status.HTTP_201_CREATED)

    # Create baseline counter
    baseline = self.client.get('/counters/update')
    self.assertEqual(baseline.status_code, status.HTTP_200_OK)

    # Check the counter value of baseline
    baseline = baseline.json['update']
    self.assertEqual(baseline, 0)

    # Update call to created counter
    result = self.client.put('/counters/update')

    # Check for successful return code
    self.assertEqual(result.status_code, status.HTTP_200_OK)

    # Check if value is one more than baseline
    baseline = self.client.get('/counters/update')
    baseline = baseline.json['update']
    self.assertEqual(baseline, 1)
```

Following similarity with the actual update method, incrementing the counter by 1. Also checking for the counter if it didn’t exist as a call could be made to a nonexistent counter.



```python
@app.route('/counters/<name>', methods=['PUT'])
def update_counter(name):
    """Updating a counter"""
    app.logger.info(f"Request to update counter: {name}")
    global COUNTERS
    if name not in COUNTERS:
        return {"Message": f"Counter {name} not found"}, status.HTTP_404_NOT_FOUND
    COUNTERS[name] += 1
    return {name: COUNTERS[name]}, status.HTTP_200_OK
```

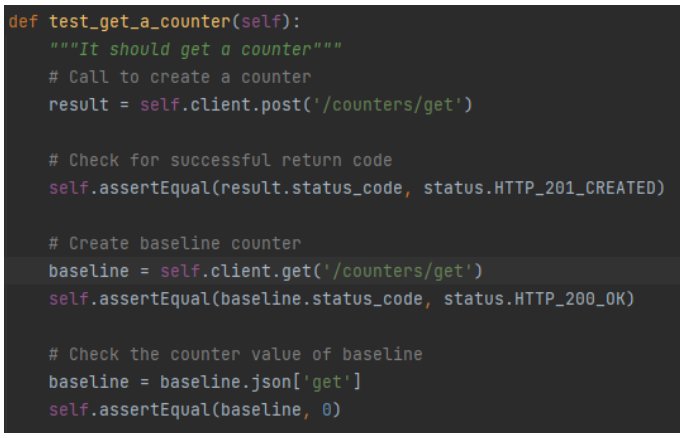
This was checked with another test case for the 404 not found error and putting the counter.



```python
def test_update_a_counter_null(self):
    """It should return an error for no counter found"""
    result = self.client.put('/counters/null')
    self.assertEqual(result.status_code, status.HTTP_404_NOT_FOUND)
```

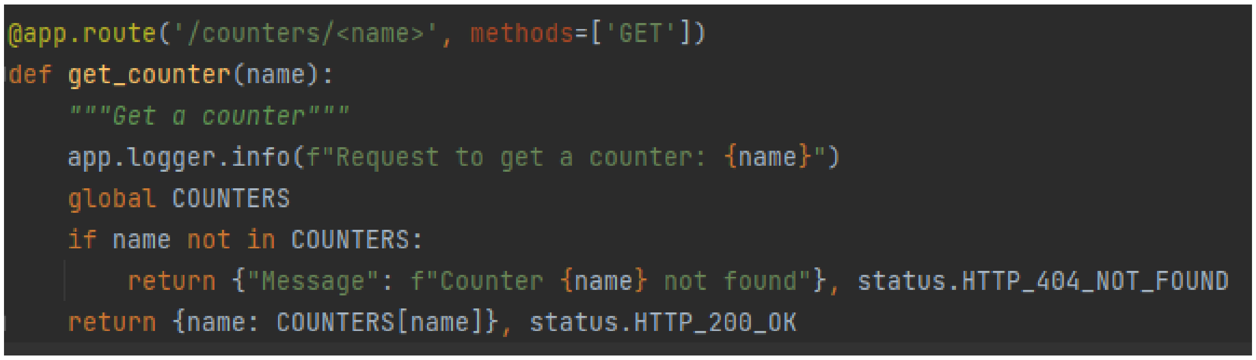
Get

For get, I based it on the Update test cases as it contained much of what would need to be tested. Namely if a get would return something and check to make it has something in it. First creating the counter and checking it, then getting the counter and checking it’s json value for what the counter is set at and that’s it’s set to 0 a newly created counter.



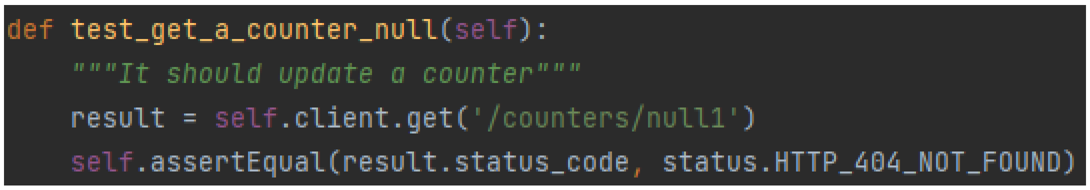
```python
def test_get_a_counter(self):
    """It should get a counter"""
    # Call to create a counter
    result = self.client.post('/counters/get')

    # Check for successful return code
    self.assertEqual(result.status_code, status.HTTP_201_CREATED)

    # Create baseline counter
    baseline = self.client.get('/counters/get')
    self.assertEqual(baseline.status_code, status.HTTP_200_OK)

    # Check the counter value of baseline
    baseline = baseline.json['get']
    self.assertEqual(baseline, 0)
```

For the get counter method in counter.py I based it on the update method except for incrementing it. As all the request needed was that the request was made, leaving a 200 ok response. Leaving the 404 error in as it can still not exist when called.



```
@app.route('/counters/<name>', methods=['GET'])
def get_counter(name):
    """Get a counter"""
    app.logger.info(f"Request to get a counter: {name}")
    global COUNTERS
    if name not in COUNTERS:
        return {"Message": f"Counter {name} not found"}, status.HTTP_404_NOT_FOUND
    return {name: COUNTERS[name]}, status.HTTP_200_OK
```

Similarly to update, I also checked the 404 error with a get call and checking if it returns a 404 error.



```
def test_get_a_counter_null(self):
    """It should update a counter"""
    result = self.client.get('/counters/null1')
    self.assertEqual(result.status_code, status.HTTP_404_NOT_FOUND)
```