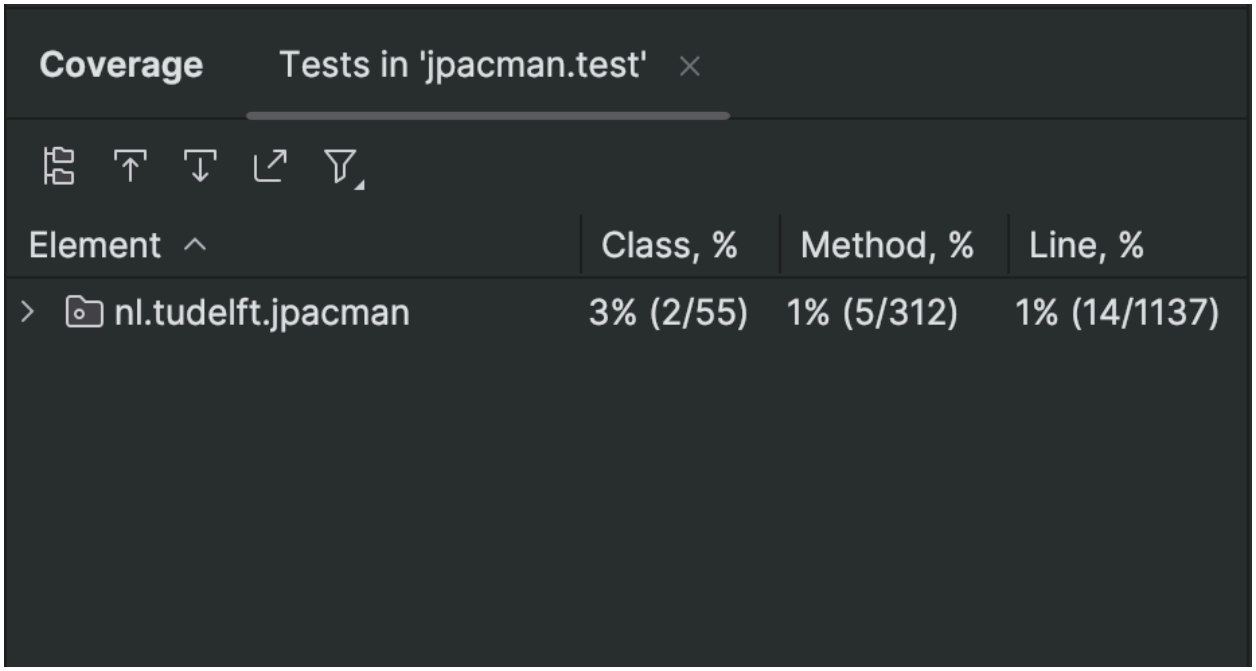
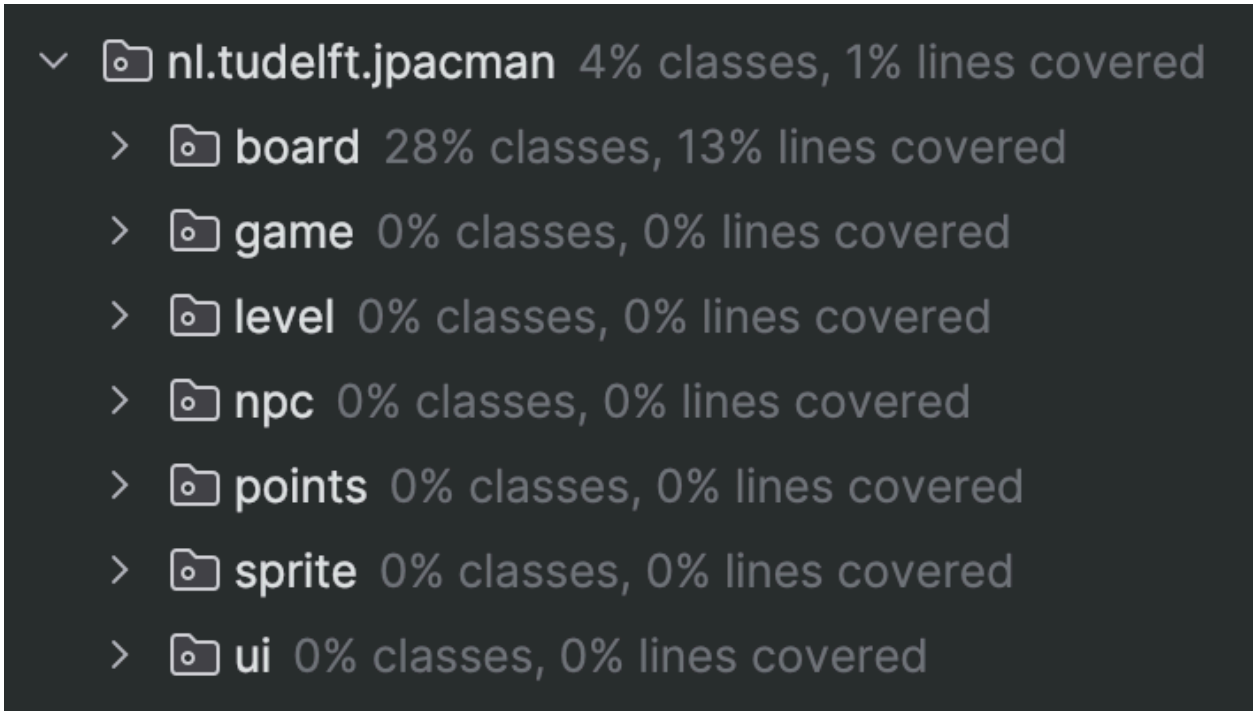
Kaitlyn Corpuz
CS 472

Task 1

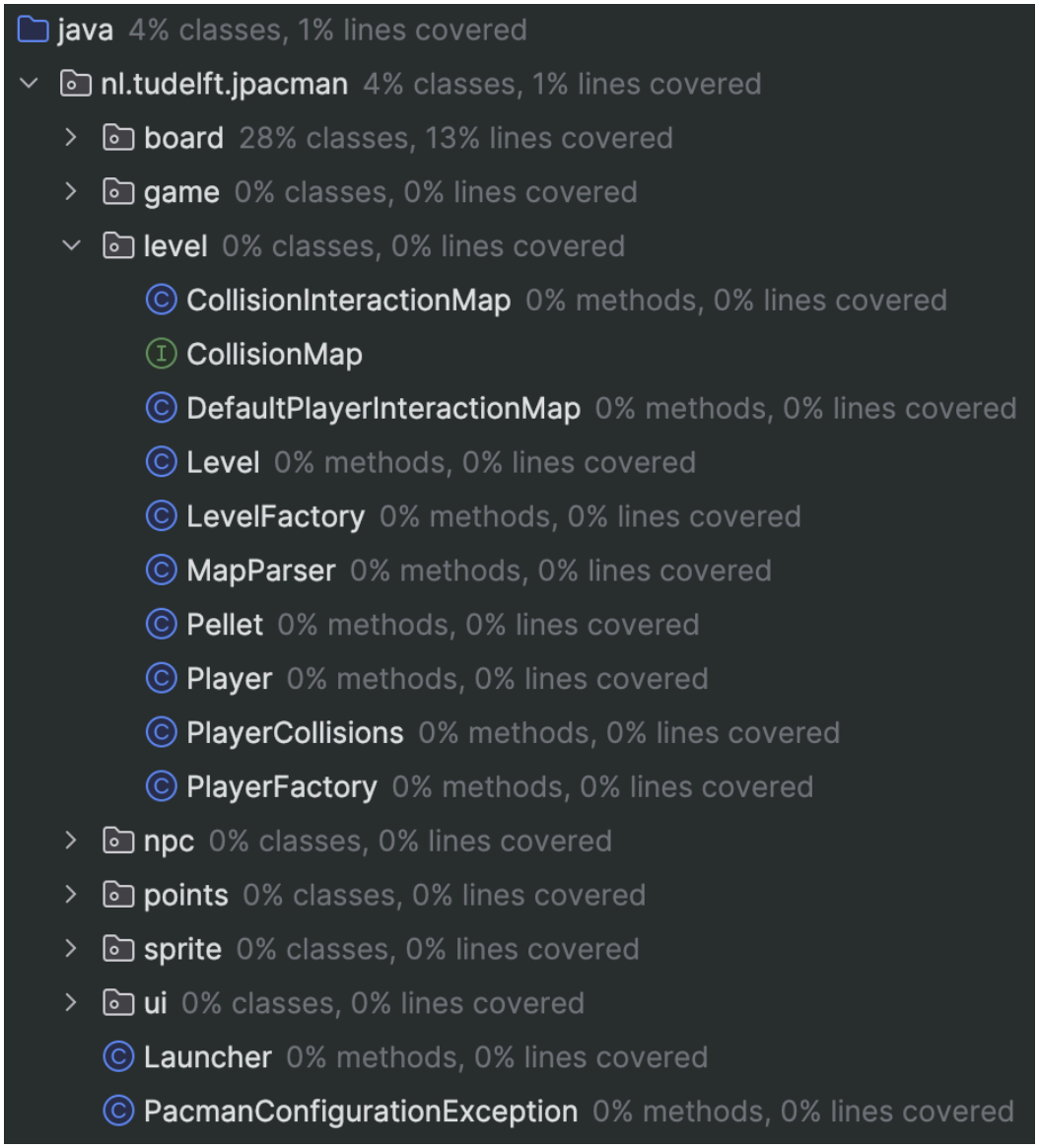




This coverage is not good enough since not even 5% of the code has coverage. Ideally we would like a 100% test coverage. Some parts of the code even have 0% coverage.

Task 2

Initial coverage



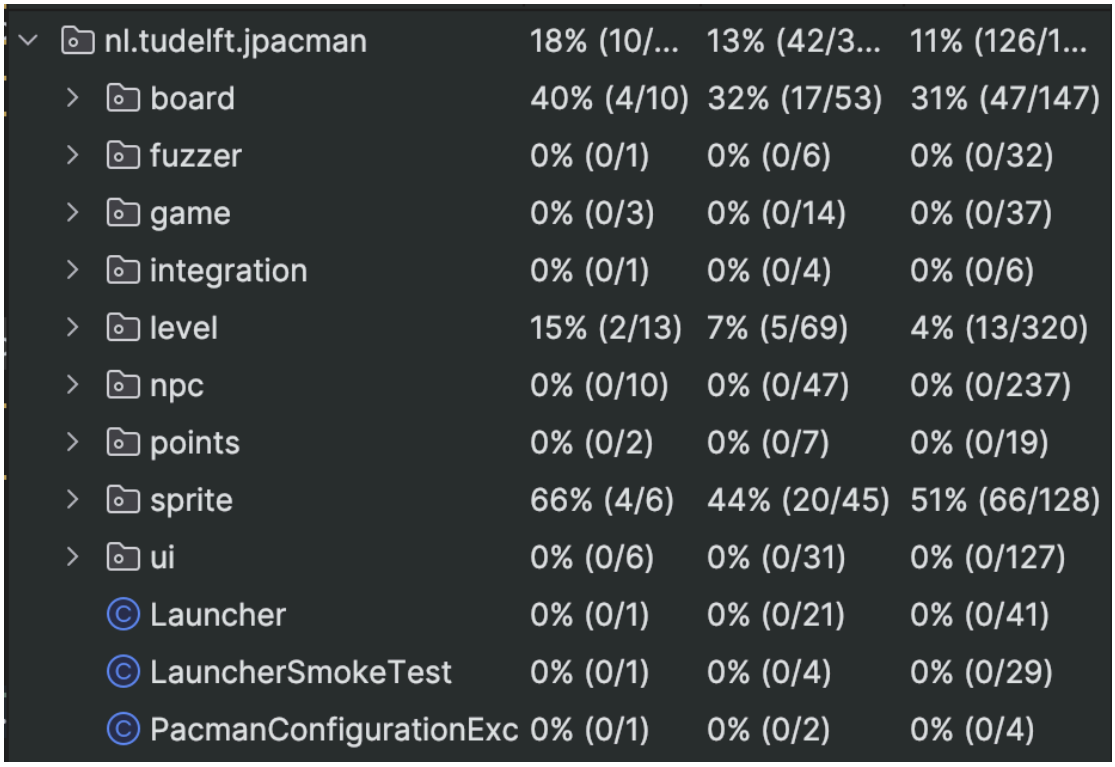
Coverage after including PlayerTest

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)
CollisionInteractionMa	0% (0/2)	0% (0/9)	0% (0/41)
CollisionMap	100% (0/0)	100% (0/0)	100% (0/0)
DefaultPlayerInteracti	0% (0/1)	0% (0/5)	0% (0/13)
Level	0% (0/2)	0% (0/17)	0% (0/113)
LevelFactory	0% (0/2)	0% (0/7)	0% (0/27)
LevelTest	0% (0/1)	0% (0/9)	0% (0/30)
MapParser	0% (0/1)	0% (0/10)	0% (0/71)
Pellet	0% (0/1)	0% (0/3)	0% (0/5)
Player	100% (1/1)	25% (2/8)	33% (8/24)
PlayerCollisions	0% (0/1)	0% (0/7)	0% (0/21)
PlayerFactory	100% (1/1)	100% (3/3)	100% (5/5)
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)
PacmanConfigurationExc	0% (0/1)	0% (0/2)	0% (0/4)

Coverage with new tests for Square.put(), Square.remove(), and Unit.hasSquare()

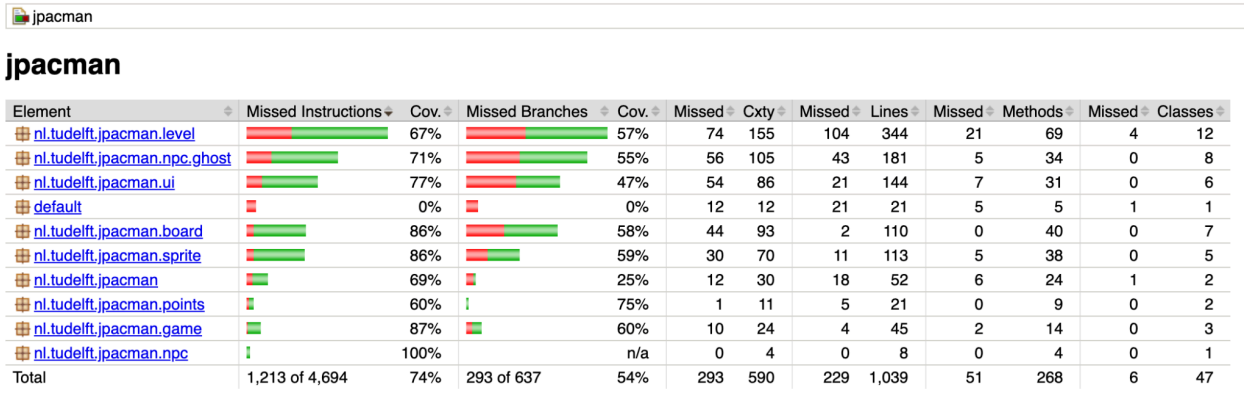
```
package nl.tudelft.jpacman.board;

import nl.tudelft.jpacman.board.Square;
import nl.tudelft.jpacman.board.Unit;
import nl.tudelft.jpacman.sprite.PacManSprites;
```

```java
import nl.tudelft.jpacman.sprite.Sprite;
import org.junit.jupiter.api.Test;

import static org.assertj.core.api.Assertions.assertThat;

public class BoardTest {
   private BasicSquare TestSquare = new BasicSquare();
   private BasicUnit TestUnit = new BasicUnit();

   @Test
   void testPut(){
       //puts Unit on Square
       //then checks to see if Square occupants includes new unit
       TestSquare.put(TestUnit);
       assert TestSquare.getOccupants().contains(TestUnit);
   }

   @Test
   void testRemove(){
       //removes Unit from Square
       //then checks to see if Square occupants does not include unit
       TestSquare.remove(TestUnit);
       assert !TestSquare.getOccupants().contains(TestUnit);
   }

   @Test
   void testHasSquare(){
       //moves Unit to Square
       //then checks to see if hasSquare() is true
       TestUnit.occupy(TestSquare);
       assertThat(TestUnit.hasSquare()).isEqualTo(true);
   }
}
```



nl.tudelft.jpacman	18% (10/...	13% (42/3...	11% (126/1...
board	40% (4/10)	32% (17/53)	31% (47/147)
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)	7% (5/69)	4% (13/320)
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)
PacmanConfigurationExc	0% (0/1)	0% (0/2)	0% (0/4)

Task 3



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%		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

1. JaCoCo and IntelliJ have different coverage results. This may be because they calculate coverage differently. For example, JaCoCo includes a distinction between missed instructions and missed branches.
2. I do find JaCoCo's source code visualization on uncovered branches helpful because it provides a useful idea on how much coverage I have and how much more I need to do.
3. I prefer JaCoCo's visualization better because it showcases more than just numbers which helps me understand the coverage better. Using green and red is also a simple way to display covered vs non-covered code.

Task 4

```
(venv) kameluna@Kaitlyns-MacBook-Pro test_coverage % nosetests

Test Account Model
- Test creating multiple Accounts
- Test Account creation using known data
- Test Account Deletion
- Test Account Search
- Test account from dict
- Test the representation of an account
- Test account to dict
- Test account update
Invalid account
- Test invalid account update

Name                 Stmts   Miss  Cover   Missing
--------------------------------------------------
models/__init__.py       7      0   100%
models/account.py       40      0   100%
--------------------------------------------------
TOTAL                   47      0   100%
----------------------------------------------------------------------
Ran 9 tests in 0.409s

OK
```

```python
def test_from_dict(self):
    """ Test account from dict """
    testDict = ACCOUNT_DATA[self.rand]
    account = Account()
    account.from_dict(testDict)
    self.assertEqual(account.name, testDict["name"])
    self.assertEqual(account.email, testDict["email"])
    self.assertEqual(account.phone_number, testDict["phone_number"])
    self.assertEqual(account.disabled, testDict["disabled"])

def test_update(self):
    """ Test account update """
    data = ACCOUNT_DATA[self.rand] # get a random account
    account = Account(**data)
```

```
    account.create()
    newData = ACCOUNT_DATA[self.rand] # get a new set of data to replace old
data
    account.from_dict(newData)
    account.update()
    self.assertEqual(len(Account.all()), 1) # make sure that same account was
updated
    self.assertEqual(account.name, newData["name"]) # make sure account matches
new data
    self.assertEqual(account.email, newData["email"])
    self.assertEqual(account.phone_number, newData["phone_number"])
    self.assertEqual(account.disabled, newData["disabled"])

def test_updateFail(self):
    """ Test invalid account update """
    data = ACCOUNT_DATA[self.rand] # get a random account
    account = Account(**data)
    account.create()
    newData = ACCOUNT_DATA[self.rand] # get a new set of data to replace old
data
    account.from_dict(newData)
    account.id = False  # turn account into invalid account
    try:
        account.update()
    except:
        print("Invalid account")

def test_delete_an_account(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) # make sure there are 0 accounts

def test_find_account(self):
    """ Test Account Search """
    data = ACCOUNT_DATA[self.rand] # get a random account
    account = Account(**data)
    account.create()
    searchID = account.id
    result = Account.find(searchID)
    self.assertEqual(result, account)# make sure found account is correct
```

Task 5

```python
def test_update_a_counter(self):
    """It should update a counter"""
    result = self.client.post('/counters/test')
    self.assertEqual(result.status_code, status.HTTP_201_CREATED)
    initialVal = result.json['test']
    updateTest = self.client.put('/counters/test')
    self.assertEqual(updateTest.status_code, status.HTTP_200_OK)
    updateValue = updateTest.json['test']
    self.assertEqual(initialVal + 1, updateValue)
```

```
(.venv) kameluna@Kaitlyns-MacBook-Pro tdd % nosetests

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

======================================================================
FAIL: It should update a counter
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/kameluna/Documents/GitHub/tdd/tests/test_counter.py", line 47, in test_update_a_counter
    self.assertEqual(updateTest.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      11      0   100%
src/status.py        6      0   100%
----------------------------------------------
TOTAL               17      0   100%
----------------------------------------------------------------------
Ran 3 tests in 0.132s

FAILED (failures=1)

(.venv) kameluna@Kaitlyns-MacBook-Pro tdd %
```

Since I have only implemented the test code and not the code itself, I am still in the **Red** phase. In my test code, I:

- Create a counter
- Validate its status code
- Retrieve its value
- Update the counter
- Validate its status code

- Retrieve its updated value
- Validate that the value has been update by one

```
@app.route('/counters/<name>', methods=['PUT'])
def update_counter(name):
    """Update a counter"""
    app.logger.info(f"Request to update counter: {name}")
    if name in COUNTERS:    # if counter does not exist, return error
        COUNTERS[name] += 1     # increment counter by 1
        return {name: COUNTERS[name]}, status.HTTP_200_OK
```

```
(.venv) kameluna@Kaitlyns-MacBook-Pro tdd % nosetests

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

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 0.145s

OK
```

Now that I've implemented the code, my tests pass and I am in the **Green** phase. However, I want to Refactor my code to include an error message for when the counter does not exist.

```
@app.route('/counters/<name>', methods=['PUT'])
def update_counter(name):
    """Update a counter"""
    app.logger.info(f"Request to update counter: {name}")
    if name in COUNTERS:    # if counter does not exist, return error
        COUNTERS[name] += 1     # increment counter by 1
        return {name: COUNTERS[name]}, status.HTTP_200_OK
    else:
        return {"Message":f"Counter {name} does not exist"},
status.HTTP_404_NOT_FOUND
```

I also have to include a test case for full coverage.

```
def test_update_a_bad_counter(self):
    """It should return an error for non-existent counter"""
    updateTest = self.client.put('/counters/test')
    self.assertEqual(updateTest.status_code, status.HTTP_404_NOT_FOUND)
```

```
(.venv) kameluna@Kaitlyns-MacBook-Pro tdd % nosetests

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

Name             Stmts   Miss  Cover   Missing
------------------------------------------------
src/counter.py      18      0   100%
src/status.py        6      0   100%
------------------------------------------------
TOTAL               24      0   100%
----------------------------------------------------------------------
Ran 4 tests in 0.140s

OK

(.venv) kameluna@Kaitlyns-MacBook-Pro tdd %
```

Now I move on to my next task.

```
def test_get_a_counter(self):
    """It should get a counter"""
    result = self.client.post('/counters/newTest')
    self.assertEqual(result.status_code, status.HTTP_201_CREATED)
    value = self.client.get('/counters/newTest')
    self.assertEqual(value.status_code, status.HTTP_200_OK)
    self.assertEqual(value.json['newTest'], 0)
```

```
(.venv) kameluna@Kaitlyns-MacBook-Pro tdd % nosetests

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

======================================================================
FAIL: It should get a counter
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/kameluna/Documents/GitHub/tdd/tests/test_counter.py", line 56, in test_get_a_counter
    self.assertEqual(value.status_code, status.HTTP_200_OK)
AssertionError: 405 != 200
-------------------- >> begin captured logging << --------------------
src.counter: INFO: Request to create counter: newTest
--------------------- >> end captured logging << ---------------------

Name             Stmts   Miss  Cover   Missing
------------------------------------------------
src/counter.py      17      0   100%
src/status.py        6      0   100%
------------------------------------------------
TOTAL               23      0   100%
----------------------------------------------------------------------
Ran 4 tests in 0.145s

FAILED (failures=1)
```

Here I have included my test code for the GET request, where I:

- Create a new counter
- Validate its status code
- Get the counter
- Validate its status code
- Validate that the counter is equal to 0

Since I have only implemented the test code and not the code itself, I am still in the **Red** phase.

```python
@app.route('/counters/<name>', methods=['GET'])
def get_counter(name):
    """Get a counter"""
    app.logger.info(f"Request to get counter: {name}")
    if name in COUNTERS:    # return counter and 200 status message if found
        return {name: COUNTERS[name]}, status.HTTP_200_OK
```

```
(.venv) kameluna@Kaitlyns-MacBook-Pro tdd % nosetests

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

Name             Stmts   Miss  Cover   Missing
----------------------------------------------
src/counter.py      23      0   100%
src/status.py        6      0   100%
----------------------------------------------
TOTAL               29      0   100%
----------------------------------------------------------------------
Ran 5 tests in 0.145s

OK
```

Now my tests pass and I am in the **Green** phase. However, I still want to do the same and output an error if the counter is not found, so I **Refactor** my code and write a test case for it.

```
@app.route('/counters/<name>', methods=['GET'])
def get_counter(name):
   """Get a counter"""
   app.logger.info(f"Request to get counter: {name}")
   if name in COUNTERS:    # return counter and 200 status message if found
       return {name: COUNTERS[name]}, status.HTTP_200_OK
   else:
       return {"Message":f"Counter {name} does not exist"},
status.HTTP_404_NOT_FOUND
```

```
(.venv) kameluna@Kaitlyns-MacBook-Pro tdd % nosetests

Counter tests
- It should create a counter
- It should return an error for duplicates
- It should return an error for getting non-existent counter
- It should get a counter
- It should return an error for updating non-existent counter
- It should update a counter

Name             Stmts   Miss  Cover   Missing
------------------------------------------------
src/counter.py      24      0   100%
src/status.py        6      0   100%
------------------------------------------------
TOTAL               30      0   100%
----------------------------------------------------------------------
Ran 6 tests in 0.157s

OK
```

Now I am back in the **Green** phase.

Link to repo: https://github.com/voxelit/2DRogueLikeUnityGame/tree/main