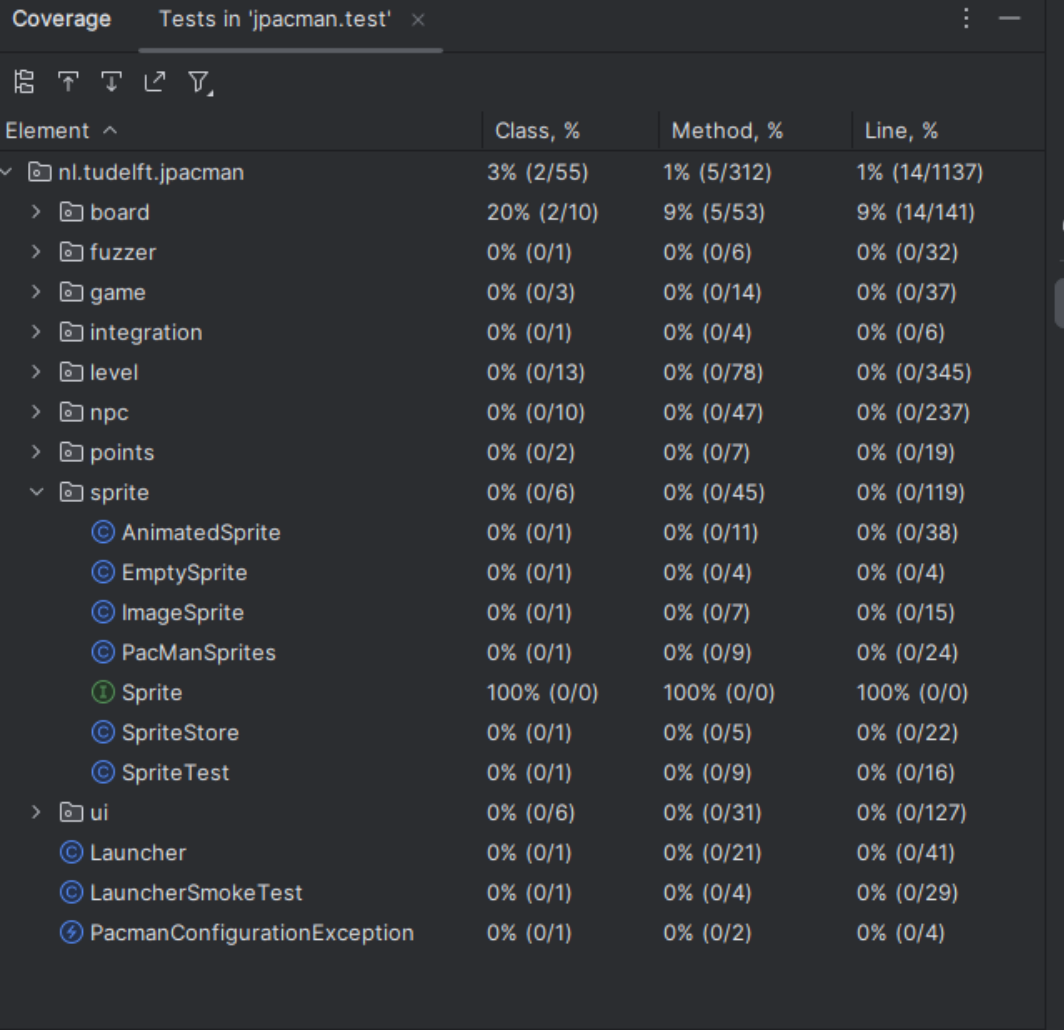


TASK 1

Photo below is the coverage without anything new added



The screenshot shows the Coverage tool in IntelliJ IDEA for the test suite 'jpacman.test'. The table displays coverage metrics for various classes and methods. Most classes have 0% coverage, while the 'Sprite' class is fully covered (100%).

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)
© AnimatedSprite	0% (0/1)	0% (0/11)	0% (0/38)
© EmptySprite	0% (0/1)	0% (0/4)	0% (0/4)
© ImageSprite	0% (0/1)	0% (0/7)	0% (0/15)
© PacManSprites	0% (0/1)	0% (0/9)	0% (0/24)
ⓘ Sprite	100% (0/0)	100% (0/0)	100% (0/0)
© SpriteStore	0% (0/1)	0% (0/5)	0% (0/22)
© SpriteTest	0% (0/1)	0% (0/9)	0% (0/16)
> 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)

- Is the coverage good enough?
 - No the coverage is not enough. Most classes and packages have 0% coverage.

TASK 2(2.1)

Photo below is the coverage with getPelletSprite() test.

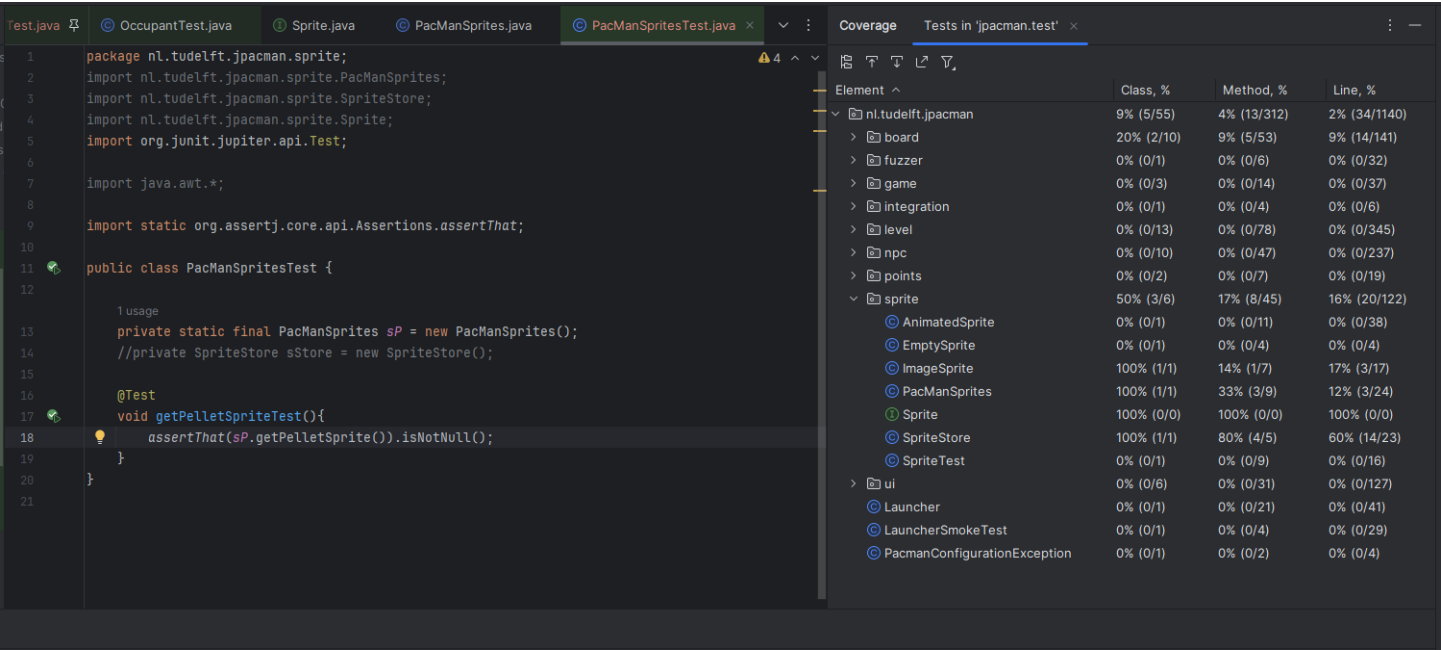


Photo below is the coverage with getSprites() test.

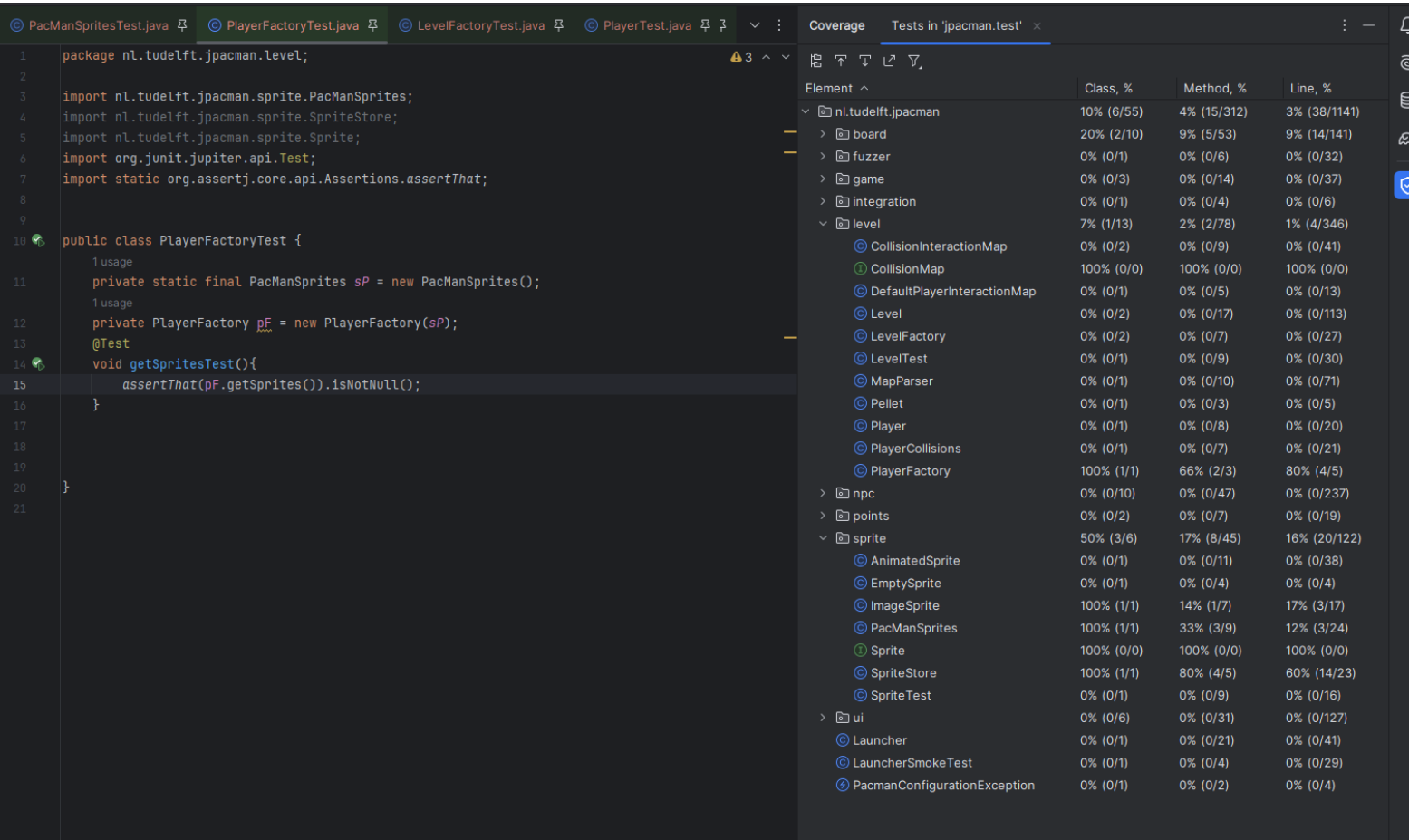


Photo below is the coverage with createGhost test.

LevelFactoryTest.java

PlayerTest.java

DirectionTest.java

OccupantTest.java

LevelFactory.java

PlayerFactory.java

Pac

```
1 package nl.tudelft.jpacman.level;
2
3 import nl.tudelft.jpacman.npc.ghost.GhostFactory;
4 import nl.tudelft.jpacman.points.DefaultPointCalculator;
5 import nl.tudelft.jpacman.points.PointCalculator;
6 import nl.tudelft.jpacman.points.PointCalculatorLoader;
7 import nl.tudelft.jpacman.sprite.PacManSprites;
8 import nl.tudelft.jpacman.points.PointCalculator;
9 import nl.tudelft.jpacman.sprite.SpriteStore;
10 import nl.tudelft.jpacman.sprite.Sprite;
11 import org.junit.jupiter.api.Test;
12 import static org.assertj.core.api.Assertions.assertThat;
13
14 public class LevelFactoryTest {
15     private static final PacManSprites pF = new PacManSprites();
16     private static final GhostFactory gF = new GhostFactory(pF);
17     private final PointCalculator pC = new DefaultPointCalculator();
18     private LevelFactory lf = new LevelFactory(pF, gF, pC);
19
20     @Test
21     void createGhostTest(){
22         assertThat(lf.createGhost()).isNotNull();
23     }
24 }
25
26
```

Coverage

Tests in 'jpacman.test'

nl.tudelft.jpacman

board

fuzzer

game

integration

level

CollisionInteractionMap

CollisionMap

DefaultPlayerInteractionMap

Level

LevelFactory

LevelTest

MapParser

Pellet

Player

PlayerCollisions

PlayerFactory

npc

points

sprite

AnimatedSprite

EmptySprite

ImageSprite

PacManSprites

Sprite

SpriteStore

SpriteTest

ui

Launcher

LauncherSmokeTest

PacmanConfigurationException

Class, %

Method, %

Line, %

nl.tudelft.jpacman

board

fuzzer

game

integration

level

CollisionInteractionMap

CollisionMap

DefaultPlayerInteractionMap

Level

LevelFactory

LevelTest

MapParser

Pellet

Player

PlayerCollisions

PlayerFactory

npc

points

sprite

AnimatedSprite

EmptySprite

ImageSprite

PacManSprites

Sprite

SpriteStore

SpriteTest

ui

Launcher

LauncherSmokeTest

PacmanConfigurationException

21% (12/55)

11% (35/312)

9% (109/1156)

20% (2/10)

9% (5/53)

9% (14/141)

0% (0/1)

0% (0/6)

0% (0/32)

0% (0/3)

0% (0/14)

0% (0/37)

0% (0/1)

0% (0/4)

0% (0/6)

15% (2/13)

5% (4/78)

4% (14/348)

0% (0/2)

0% (0/9)

0% (0/41)

100% (0/0)

100% (0/0)

100% (0/0)

0% (0/1)

0% (0/5)

0% (0/13)

0% (0/2)

0% (0/17)

0% (0/113)

50% (1/2)

28% (2/7)

34% (10/29)

0% (0/1)

0% (0/9)

0% (0/30)

0% (0/1)

0% (0/10)

0% (0/71)

0% (0/1)

0% (0/3)

0% (0/5)

0% (0/1)

0% (0/8)

0% (0/20)

0% (0/1)

0% (0/7)

0% (0/21)

100% (1/1)

66% (2/3)

80% (4/5)

40% (4/10)

12% (6/47)

6% (17/243)

0% (0/2)

0% (0/7)

0% (0/20)

66% (4/6)

44% (20/45)

50% (64/128)

100% (1/1)

36% (4/11)

34% (15/44)

0% (0/1)

0% (0/4)

0% (0/4)

100% (1/1)

85% (6/7)

76% (13/17)

100% (1/1)

55% (5/9)

58% (14/24)

100% (0/0)

100% (0/0)

100% (0/0)

100% (1/1)

100% (5/5)

95% (22/23)

0% (0/1)

0% (0/9)

0% (0/16)

0% (0/6)

0% (0/31)

0% (0/127)

0% (0/1)

0% (0/21)

0% (0/41)

0% (0/1)

0% (0/4)

0% (0/29)

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	<div></div>	67%	<div></div>	57%	74	155	104	344	21	69	4	12
nl.tudelft.jpacman.npc.ghost	<div></div>	71%	<div></div>	55%	56	105	43	181	5	34	0	8
nl.tudelft.jpacman.ui	<div></div>	77%	<div></div>	47%	54	86	21	144	7	31	0	6
default	<div></div>	0%	<div></div>	0%	12	12	21	21	5	5	1	1
nl.tudelft.jpacman.board	<div></div>	86%	<div></div>	58%	44	93	2	110	0	40	0	7
nl.tudelft.jpacman.sprite	<div></div>	86%	<div></div>	59%	30	70	11	113	5	38	0	5
nl.tudelft.jpacman	<div></div>	69%	<div></div>	25%	12	30	18	52	6	24	1	2
nl.tudelft.jpacman.points	<div></div>	60%	<div></div>	75%	1	11	5	21	0	9	0	2
nl.tudelft.jpacman.game	<div></div>	87%	<div></div>	60%	10	24	4	45	2	14	0	3
nl.tudelft.jpacman.npc	<div></div>	100%	<div></div>	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?

JaCoCo and the Coverage test from IntelliJ are a lot different from each other. The JaCoCo tests cover missed instruction and branches while IntelliJ covers line, method and class coverage. JaCoCo also has a missed Methods and Classes column but it does not feature a percentage like IntelliJ.

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

Yes I did find it useful. It is a detailed way of figuring out what branches you have missed. It is nice to be able to click through each package and class and figure out what exactly you are missing.

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

I would prefer JaCoCo over IntelliJ. I feel as though JaCoCo has a more in-depth report than IntelliJ's one, and I like how you are able to click through the packages and classes, to the point where it even highlights which portions of the code are missed.

TASK 4

First function

```
def test_from_dict(self):
    """ Test account from dict """
    data = ACCOUNT_DATA[self.rand]
    account = Account()
    account.from_dict(data)
    self.assertEqual(account.name, data["name"])
```

```
C:\Users\ania\OneDrive\Desktop\UNLV\SeniorYear\SpringSem\CS472\test_coverage>python3 -m nose
```

```
Test Account Model
- Test creating multiple Accounts
- Test Account creation using known data
- Test account from dict
- Test the representation of an account
- Test account to dict
```

Name	Stmts	Miss	Cover	Missing
models__init__.py	7	0	100%	
models\account.py	40	9	78%	45-48, 52-54, 74-75
TOTAL	47	9	81%	

```
Ran 5 tests in 0.440s
```

```
OK
```

Second function

```
def test_update(self):
    """ Test updating an account """
    data = ACCOUNT_DATA[self.rand]
    account = Account()
    account.create()
    account.name = "Foo"
    account.update({})
    self.assertEqual(account.name, "Foo")
    account.id = 0
    with self.assertRaises(DataValidationError):
        account.update()
```

```
C:\Users\ania\OneDrive\Desktop\UNLV\SeniorYear\SpringSem\CS472\test_coverage>python3 -m nose
```

Test Account Model

- Test creating multiple Accounts
- Test Account creation using known data
- Test account from dict
- 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	5	88%	52-54, 74-75
TOTAL	47	5	89%	

Ran 6 tests in 0.449s

OK

Third function

```
def test_delete(self):
    """ Test deleting an account """
    data = ACCOUNT_DATA[self.rand]
    account = Account(**data)
    account.create()
    account.delete()
    self.assertEqual(len(Account.all()), 0)
```

```
C:\Users\ania\OneDrive\Desktop\UNLV\SeniorYear\SpringSem\CS472\test_coverage>python3 -m nose
```

Test Account Model

- Test creating multiple Accounts
- Test Account creation using known data
- Test deleting an account
- Test account from dict
- 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	2	95%	74-75
TOTAL	47	2	96%	

Ran 7 tests in 0.443s

OK

```
C:\Users\ania\OneDrive\Desktop\UNLV\SeniorYear\SpringSem\CS472\test_coverage>
```

Fourth function

```
def test_find(self):
    """ Test finding an account """
    data = ACCOUNT_DATA[self.rand]
    account = Account(**data)
    account.create()
    result = Account.find(account.id)
    self.assertEqual(account.id, result.id)
    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])
```

```
C:\Users\ania\OneDrive\Desktop\UNLV\SeniorYear\SpringSem\CS472\test_coverage>python3 -m nose
```

```
Test Account Model
- Test creating multiple Accounts
- Test Account creation using known data
- Test deleting an account
- Test finding an account
- Test account from dict
- 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 0.443s
```

```
OK
```

```
C:\Users\ania\OneDrive\Desktop\UNLV\SeniorYear\SpringSem\CS472\test_coverage>
```

TASK 5

Next two photos are photos of changes in test_counter.py and its resulting output

```
def test_update_a_counter(self):
    """It should update a counter and increase its value by 1"""
    create_result = self.client.post('/counters/dan')
    self.assertEqual(create_result.status_code, 201)
    baseline_value = create_result.json['dan']
    update_result = self.client.put('/counters/dan')
    self.assertEqual(update_result.status_code, 200)
    updated_value = update_result.json['dan']
    self.assertEqual(updated_value, baseline_value + 1)
```

```
C:\Users\ania\OneDrive\Desktop\UNLV\SeniorYear\SpringSem\CS472\tdd>python3 -m nose
```

```
Counter tests
- It should create a counter
- It should return an error for duplicates
- It should update a counter and increase its value by 1 (FAILED)
```

```
FAIL: It should update a counter and increase its value by 1
```

```
Traceback (most recent call last):
```

```
File "C:\Users\ania\OneDrive\Desktop\UNLV\SeniorYear\SpringSem\CS472\tdd\tests\test_counter.py", line 47, in test_update_a_counter
    self.assertEqual(update_result.status_code, 200)
```

```
AssertionError: 405 != 200
```

```
>> begin captured logging <<
```

```
src.counter: INFO: Request to create counter: dan
```

```
>> 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 0.169s
```

```
FAILED (failures=1)
```

Next two photos are photos of changes in counter.py and its resulting output

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

```
C:\Users\ania\OneDrive\Desktop\UNLV\SeniorYear\SpringSem\CS472\tdd>python3 -m nose
```

```
Counter tests
- It should create a counter
- It should return an error for duplicates
- It should update a counter and increase its value by 1
```

Name	Stmts	Miss	Cover	Missing
src\counter.py	18	1	94%	28
src\status.py	6	0	100%	
TOTAL	24	1	96%	

```
Ran 3 tests in 0.166s
```

```
OK
```

```
C:\Users\ania\OneDrive\Desktop\UNLV\SeniorYear\SpringSem\CS472\tdd>
```

Next two photos are photos after changes made to test_counter.py in order to write a function to test getting value of the counter and its resulting output.

```
def test_get_a_counter(self):
    """It should return the value of the counter"""
    create_result = self.client.post('/counters/alex')
    self.assertEqual(create_result.status_code, 201)
    get_result = self.client.get('/counters/alex')
    self.assertEqual(get_result.status_code, 200)
    self.assertEqual(get_result.json['alex'], 0)
```

```
C:\Users\ania\OneDrive\Desktop\UNLV\SeniorYear\SpringSem\CS472\tdd>python3 -m nose
```

```
Counter tests
- It should create a counter
- It should return an error for duplicates
- It should return the value of the counter (FAILED)
- It should update a counter and increase its value by 1
```

```
FAIL: It should return the value of the counter
```

```
Traceback (most recent call last):
  File "C:\Users\ania\OneDrive\Desktop\UNLV\SeniorYear\SpringSem\CS472\tdd\tests\test_counter.py", line 56, in test_get_a_counter
    self.assertEqual(get_result.status_code, 200)
AssertionError: 405 != 200

>> begin captured logging <<
src.counter: INFO: Request to create counter: alex
>> end captured logging <<
```

Name	Stmts	Miss	Cover	Missing
src\counter.py	18	1	94%	28
src\status.py	6	0	100%	
TOTAL	24	1	96%	

```
Ran 4 tests in 0.163s
```

```
FAILED (failures=1)
```

```
C:\Users\ania\OneDrive\Desktop\UNLV\SeniorYear\SpringSem\CS472\tdd>
```

Next two photos are after changes made to counter.py in order to fix the error

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

```
C:\Users\ania\OneDrive\Desktop\UNLV\SeniorYear\SpringSem\CS472\tdd>python3 -m nose
```

```
Counter tests
```

- It should create a counter
- It should return an error for duplicates
- It should return the value of the counter
- It should update a counter and increase its value by 1

Name	Stmts	Miss	Cover	Missing
------	-------	------	-------	---------

src\counter.py	24	2	92%	28, 38
src\status.py	6	0	100%	

TOTAL	30	2	93%	
-------	----	---	-----	--

```
Ran 4 tests in 0.159s
```

```
OK
```

```
C:\Users\ania\OneDrive\Desktop\UNLV\SeniorYear\SpringSem\CS472\tdd>
```