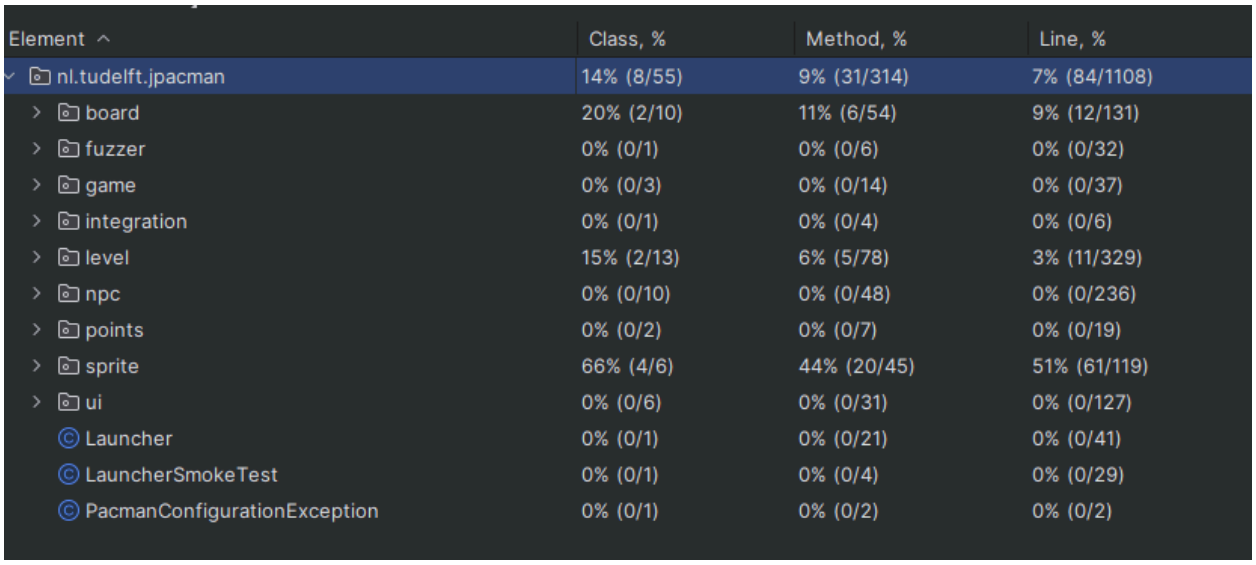
Kelvin Do
CS 472

Testing Lab Report

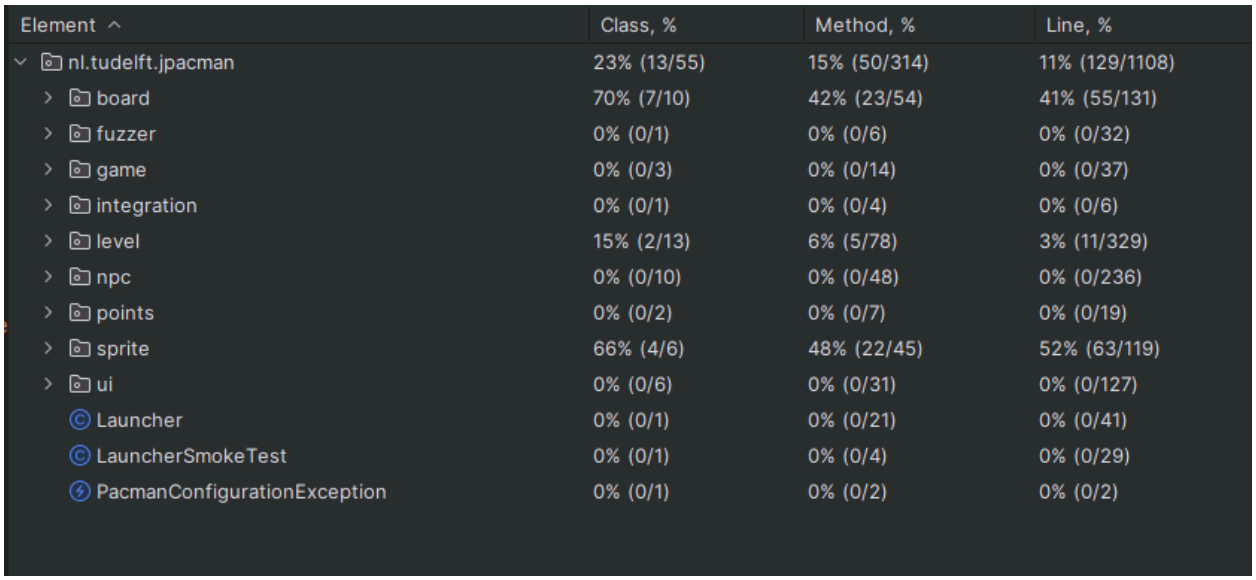
Forked Repos:
https://github.com/Haikp/jpacman
https://github.com/Haikp/test_coverage
https://github.com/Haikp/tdd

Task 1 Question -
There is not enough coverage. The image below shows that barely any of the methods created were tested, only including the sample test case of instantiating objects for the game.
Before tests were made:



Element	Class, %	Method, %	Line, %
nl.tudelft.jpacman	14% (8/55)	9% (31/314)	7% (84/1108)
board	20% (2/10)	11% (6/54)	9% (12/131)
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% (11/329)
npc	0% (0/10)	0% (0/48)	0% (0/236)
points	0% (0/2)	0% (0/7)	0% (0/19)
sprite	66% (4/6)	44% (20/45)	51% (61/119)
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% (0/2)

Task 2.1 and 3 Question(s) -
After tests were made:



Element	Class, %	Method, %	Line, %
nl.tudelft.jpacman	23% (13/55)	15% (50/314)	11% (129/1108)
board	70% (7/10)	42% (23/54)	41% (55/131)
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% (11/329)
npc	0% (0/10)	0% (0/48)	0% (0/236)
points	0% (0/2)	0% (0/7)	0% (0/19)
sprite	66% (4/6)	48% (22/45)	52% (63/119)
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% (0/2)

Element	Class, %	Method, %	Line, %
nl.tudelft.jpacman	23% (13/55)	15% (49/312)	12% (144/1154)
board	70% (7/10)	41% (22/53)	43% (63/144)
Board	100% (1/1)	71% (5/7)	66% (12/18)
BoardFactory	100% (3/3)	63% (7/11)	86% (25/29)
BoardFactoryTest	0% (0/1)	0% (0/6)	0% (0/18)
BoardTest	0% (0/1)	0% (0/3)	0% (0/3)
Direction	100% (1/1)	100% (4/4)	100% (11/11)
Square	100% (1/1)	50% (4/8)	47% (11/23)
SquareTest	0% (0/1)	0% (0/4)	0% (0/13)
Unit	100% (1/1)	20% (2/10)	13% (4/29)

Coverage visualized using JaCoCo:

nl.tudelft.jpacman.board

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
Unit		81%		58%	14	27	0	29	0	10	0	1
Square		79%		53%	13	22	1	23	0	8	0	1
Board		81%		56%	14	23	1	18	0	7	0	1
BoardFactory		94%		75%	3	11	0	19	0	5	0	1
Direction		100%		n/a	0	4	0	11	0	4	0	1
BoardFactory.Wall		100%		n/a	0	3	0	5	0	3	0	1
BoardFactory.Ground		100%		n/a	0	3	0	5	0	3	0	1
Total	77 of 581	86%	44 of 106	58%	44	93	2	110	0	40	0	7

1) Comparing the two images above, we can see that the outputs are fairly similar, however JaCoCo checks for percentage of missed branches, while IntelliJ does not.

2) The source code is very helpful as it serves as a visual representation of what still needs to be tested, which as though we should be aware of which methods to test, this helps as a reminder.

3) Between the two coverage visualizer, I prefer JaCoCo as it shows more information compared to IntelliJ's table of information. Knowing visually what I still need to check is very helpful when creating tests.

Task 4 -

```python
def test_from_dict(self):
    account = Account()
    dicttest = {
        "id": "0",
        "name": "test",
        "email": "smth@nothing.com",
        "phone_number": "1234567890",
        "disabled": True,
        "date_joined": 2002
    }

    account.from_dict(dicttest)

    self.assertEqual(int(account.id), int(0))
```

```python
    new *
    def test_create(self):
        account = Account()
        account.create()
        self.assertEqual(len(Account.all()),  second: 1)
```

```python
def test_update(self):
    try:
        account = Account()
        account.update()
    except:
        data = ACCOUNT_DATA[self.rand] # get a random account
        account = Account(**data)
        account.create()
        account.id = 1
        account.update()

    pullAccount = account.find(1)

    self.assertEqual(int(pullAccount.id), int(1))
```

```python
def test_delete(self):
    account = Account()
    account.create()
    account.delete()

    self.assertEqual(len(Account.all()), second: 0)
```

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

    findAccount = account.find(account.id)
    self.assertEqual(int(findAccount.id), int(account.id))
```

```
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 2.345s

OK

(venv) PS D:\CS\CS472\labs\test_coverage\test_coverage>
```

Task 5 -

```python
from unittest import TestCase

# we need to import the unit under test - counter
from src.counter import app

# we need to import the file that contains the status codes
from src import status

Koi
class CounterTest(TestCase):
    """Counter tests"""
```

Code above placed inside test_counter.py

```python
from flask import Flask
from src import status

app = Flask(__name__)

COUNTERS = {}
```

Code above placed inside counter.py

```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)
```

Code above placed inside test_counter.py

```
======================================================================
FAIL: It should create a counter
----------------------------------------------------------------------
Traceback (most recent call last):
  File "D:\CS\CS472\labs\forked_tdd\tdd\tests\test_counter.py", line 31, 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 0.228s

FAILED (failures=1)

(venv) PS D:\CS\CS472\labs\forked_tdd\tdd>
```

nosetest fails as expected

```python
COUNTERS = {}

# We will use the app decorator and create a route called slash counters.
# specify the variable in route <name>
# let Flask know that the only methods that is allowed to called
# on this function is "POST".
Koi *
@app.route( rule: '/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
```

Code above placed inside counter.py

```
(venv) PS D:\CS\CS472\labs\forked_tdd\tdd> nosetests

Counter tests
- It should create a counter

Name             Stmts   Miss  Cover   Missing
----------------------------------------------
src\counter.py      11      1    91%   18
src\status.py        6      0   100%
----------------------------------------------
TOTAL               17      1    94%
----------------------------------------------------------------------
Ran 1 test in 0.234s

OK

(venv) PS D:\CS\CS472\labs\forked_tdd\tdd>
```

nosetest passes as expected

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

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

Refactored code placed in test_counter.py

```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)
```

Code above placed in test_counter.py

```
(venv) PS D:\CS\CS472\labs\forked_tdd\tdd> nosetests

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

======================================================================
FAIL: It should return an error for duplicates
----------------------------------------------------------------------
Traceback (most recent call last):
  File "D:\CS\CS472\labs\forked_tdd\tdd\tests\test_counter.py", line 38, in test_duplicate_a_counter
    self.assertEqual(result.status_code, status.HTTP_409_CONFLICT)
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 0.236s

FAILED (failures=1)

(venv) PS D:\CS\CS472\labs\forked_tdd\tdd>
```

test_duplicate_a_counter fails

```python
@app.route( rule: '/counters/<name>', methods=['POST'])
def create_counter(name):
    """Create a counter"""
    app.logger.info(f"Request to create counter: {name}")
    global COUNTERS
    if name in COUNTERS:
        return {"Message":f"Counter {name} already exists"}, status.HTTP_409_CONFLICT
    COUNTERS[name] = 0
    return {name: COUNTERS[name]}, status.HTTP_201_CREATED
```

Added if statement inside create_counter method

```
(venv) PS D:\CS\CS472\labs\forked_tdd\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 0.239s

OK

(venv) PS D:\CS\CS472\labs\forked_tdd\tdd>
```

test_duplicate_a_counter now passes

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

In function created above to update a counter, it takes two paths, whether or not it is able to find an existing counter or not.

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

 Koi
def test_false_counter_update(self):
    """It should return an error for nonexistent counter updates"""
    result = self.client.put('/counters/update')
    self.assertEqual(result.status_code, status.HTTP_404_NOT_FOUND)
```

The two methods above tests both possible cases

```python
@app.route( rule: '/counters/<name>', methods=['GET'])
def fetch_counter(name):
    app.logger.info(f"Request to fetch counter: {name}")
    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
```

The function to fetch a counter follows a similar code structure to the update counter, except it does not increment the value.

```
(venv) PS D:\CS\CS472\labs\forked_tdd\tdd> nosetests

Counter tests
- It should create a counter
- It should return an error for duplicates
- It should return an error for nonexistent counter updates
- It should return an error for false fetches
- It should fetch a 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 0.240s

OK

(venv) PS D:\CS\CS472\labs\forked_tdd\tdd>
```

As shown above, nosetests passes with 100% cover.

Final files -
test_counter.py

```python
class CounterTest(TestCase):
    """Counter tests"""

    Koi
    def setUp(self):
        self.client = app.test_client()

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

    Koi
    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)

    Koi
    def test_update_a_counter(self):
        """It should update a counter"""
        result = self.client.post('/counters/update')
        self.assertEqual(result.status_code, status.HTTP_201_CREATED)
        prevValue = result.json.get('update')
        result = self.client.put('/counters/update')
        self.assertEqual(result.status_code, status.HTTP_200_OK)
        self.assertEqual(result.json.get('update'), prevValue + 1)

    Koi
    def test_false_counter_update(self):
        """It should return an error for nonexistent counter updates"""
        result = self.client.put('/counters/update')
        self.assertEqual(result.status_code, status.HTTP_404_NOT_FOUND)

    Koi
    def test_fetch_a_counter(self):
        """It should fetch a counter"""
        result = self.client.post('counters/fetch')
        self.assertEqual(result.status_code, status.HTTP_201_CREATED)
        fetchedCounter = self.client.get('counters/fetch')
        self.assertEqual(fetchedCounter.status_code, status.HTTP_200_OK)
        self.assertEqual(result.json.get('fetch'), fetchedCounter.json.get('fetch'))

    Koi
    def test_false_fetch_counter(self):
        """It should return an error for false fetches"""
        fetchedCounter = self.client.get('counters/fetch')
        self.assertEqual(fetchedCounter.status_code, status.HTTP_404_NOT_FOUND)
```

counter.py

```python
from flask import Flask
from src import status

app = Flask(__name__)

COUNTERS = {}

# We will use the app decorator and create a route called slash counters.
# specify the variable in route <name>
# let Flask know that the only methods that is allowed to called
# on this function is "POST".
Koi
@app.route( rule: '/counters/<name>', methods=['POST'])
def create_counter(name):
    """Create a counter"""
    app.logger.info(f"Request to create counter: {name}")
    global COUNTERS
    if name in COUNTERS:
        return {"Message":f"Counter {name} already exists"}, status.HTTP_409_CONFLICT
    COUNTERS[name] = 0
    return {name: COUNTERS[name]}, status.HTTP_201_CREATED

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

Koi
@app.route( rule: '/counters/<name>', methods=['GET'])
def fetch_counter(name):
    app.logger.info(f"Request to fetch counter: {name}")
    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
```