Alec Him

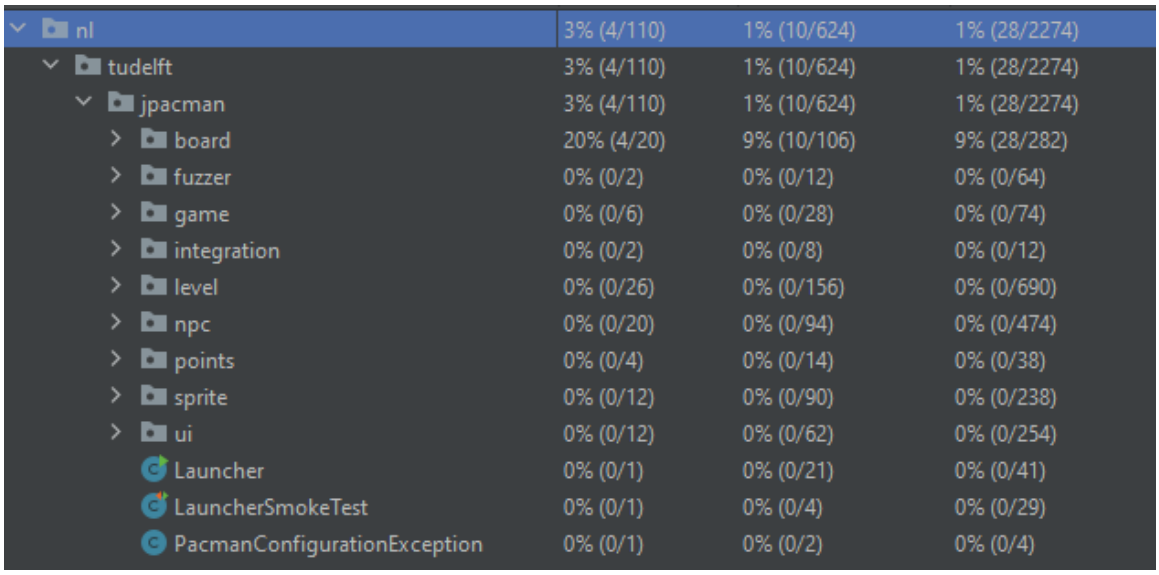
CS 472 - 1001

Due by: 2/5/24

GitHub Repo for JPacman: https://github.com/Redblaze74/jpacman

---

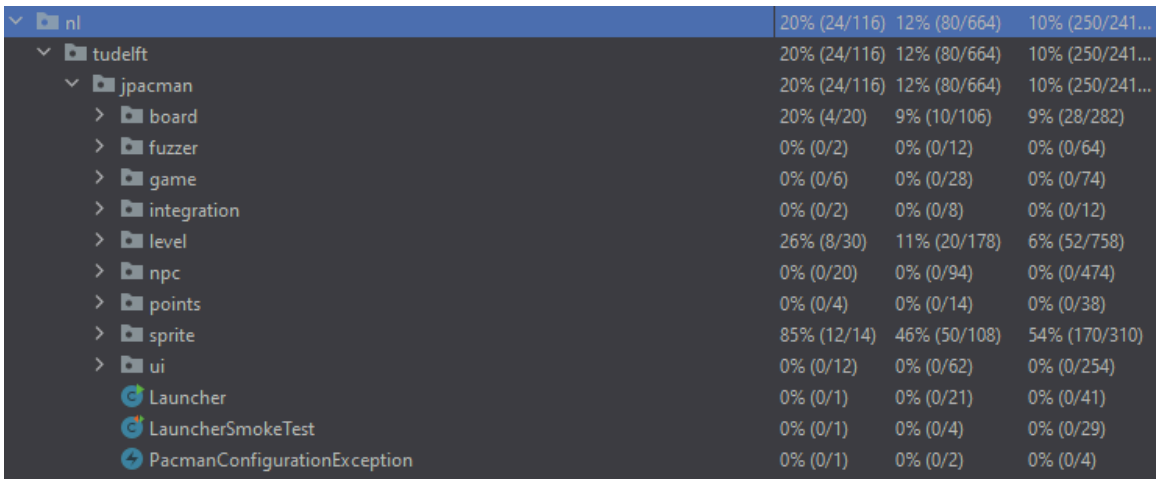
## Task 1 - JPacman Test Coverage



| | | | |
|---|---|---|---|
| nl | 3% (4/110) | 1% (10/624) | 1% (28/2274) |
| tudelft | 3% (4/110) | 1% (10/624) | 1% (28/2274) |
| jpacman | 3% (4/110) | 1% (10/624) | 1% (28/2274) |
| board | 20% (4/20) | 9% (10/106) | 9% (28/282) |
| fuzzer | 0% (0/2) | 0% (0/12) | 0% (0/64) |
| game | 0% (0/6) | 0% (0/28) | 0% (0/74) |
| integration | 0% (0/2) | 0% (0/8) | 0% (0/12) |
| level | 0% (0/26) | 0% (0/156) | 0% (0/690) |
| npc | 0% (0/20) | 0% (0/94) | 0% (0/474) |
| points | 0% (0/4) | 0% (0/14) | 0% (0/38) |
| sprite | 0% (0/12) | 0% (0/90) | 0% (0/238) |
| ui | 0% (0/12) | 0% (0/62) | 0% (0/254) |
| 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?
    - Considering the main nl folder is 3% for class coverage and 1% for both method and line coverage, no it is not good enough.

## Task 2 - Increasing Coverage on JPacman



| | | | |
|---|---|---|---|
| nl | 20% (24/116) | 12% (80/664) | 10% (250/241... |
| tudelft | 20% (24/116) | 12% (80/664) | 10% (250/241... |
| jpacman | 20% (24/116) | 12% (80/664) | 10% (250/241... |
| board | 20% (4/20) | 9% (10/106) | 9% (28/282) |
| fuzzer | 0% (0/2) | 0% (0/12) | 0% (0/64) |
| game | 0% (0/6) | 0% (0/28) | 0% (0/74) |
| integration | 0% (0/2) | 0% (0/8) | 0% (0/12) |
| level | 26% (8/30) | 11% (20/178) | 6% (52/758) |
| npc | 0% (0/20) | 0% (0/94) | 0% (0/474) |
| points | 0% (0/4) | 0% (0/14) | 0% (0/38) |
| sprite | 85% (12/14) | 46% (50/108) | 54% (170/310) |
| ui | 0% (0/12) | 0% (0/62) | 0% (0/254) |
| 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

- Google Sheets
- | | | |
  |---|---|---|
  | 11 | Alec Him | src/main/java/nl/tudelft/jpacman/level/LevelFactory.createLevel |
  | 12 | Alec Him | src/main/java/nl/tudelft/jpacman/board/BoardFactory.createBoard |
  | 13 | Alec Him | src/main/java/nl/tudelft/jpacman/game/GameFactory.createSinglePlayerGame |
- Before Unit Tests
  - | | | | |
    |---|---|---|---|
    | LevelFactory | 0% (0/2) | 0% (0/7) | 0% (0/27) |
  - | | | | |
    |---|---|---|---|
    | BoardFactory | 0% (0/3) | 0% (0/11) | 0% (0/27) |
    | BoardFactoryTest | 0% (0/1) | 0% (0/6) | 0% (0/18) |
  - | | | | |
    |---|---|---|---|
    | GameFactory | 0% (0/1) | 0% (0/3) | 0% (0/4) |
- After Unit Tests
  - | | | | |
    |---|---|---|---|
    | LevelFactory | 50% (1/2) | 28% (2/7) | 27% (8/29) |
    - Instantiated PacManSprites, GhostFactory, and PointCalculator to create LevelFactory
    - Mocked a board and created two array lists of ghost and square to call the method createLevel
    - Asserted that the createLevel is not null
    - ```
      /*
       * - LevelFactory(PacManSprites, GhostFactory, PointCalculator)
       *   > Instantiations
       *     - PacManSprites
       *     - PacManSprites => GhostFactory
       *     - PointCalculatorLoader => PointCalculator
       */
      2 usages
      private static final PacManSprites pms = new PacManSprites();
      1 usage
      private final GhostFactory ghsFac = new GhostFactory(pms);
      1 usage
      private final PointCalculatorLoader pcl = new PointCalculatorLoader();
      1 usage
      private final PointCalculator pc = pcl.load();
      1 usage
      private final LevelFactory lvlFac = new LevelFactory(pms, ghsFac, pc);
      ```
      ```

```java
/*
 * - createLevel(Board, List<Ghost>, List<Square>)
 *   > Mock Board, Create Lists using ArrayList
 */
1 usage
private final Board board = mock(Board.class);
1 usage
private final List<Ghost> ghosts = new ArrayList<>();
1 usage
private final List<Square> startPositions = new ArrayList<>();

no usages  new *
@Test
void testCreateLvl(){
    assertThat(lvlFac.createLevel(board, ghosts, startPositions)).isNotNull();
}
}
```

- 

| | | | |
|---|---|---|---|
| BoardFactory | 33% (1/3) | 27% (3/11) | 58% (17/29) |
| BoardFactoryTest | 100% (1/1) | 100% (2/2) | 100% (7/7) |

  - Instantiated PacManSprites to create BoardFactory
  - Instantiated two Squares as BasicSquares() to create the 2D Square needed to call the method createBoard
  - Asserted that createBoard is not null

```java
/*
 * - BoardFactory(PacManSprites)
 * - createBoard(Square[][])
 */
1 usage
private static final PacManSprites pms = new PacManSprites();
1 usage
private final BoardFactory boardFac = new BoardFactory(pms);
1 usage
private final Square s1 = new BasicSquare();
1 usage
private final Square s2 = new BasicSquare();

no usages  new *
@Test
void testCreateBoard(){
    assertThat(boardFac.createBoard(new Square[][]{{s1}, {s2}})).isNotNull();
}
```

- 

| | | | |
|---|---|---|---|
| GameFactory | 100% (1/1) | 66% (2/3) | 80% (4/5) |

  - Instantiated PacManSprites and PlayerFactory to create GameFactory

- Mocked Level, and instantiated PointCalculator to call the method createSinglePlayerGame
- Asserted that createSinglePlayerGame is not null
- 

```
/*
 * - GameFactory(PlayerFactory)
 *    > PlayerFactory
 *       - PacManSprites => PlayerFactory
 */
1 usage
private final static PacManSprites pms = new PacManSprites();
1 usage
private final PlayerFactory factory = new PlayerFactory(pms);
1 usage
private final GameFactory gameFac = new GameFactory(factory);
```

- 

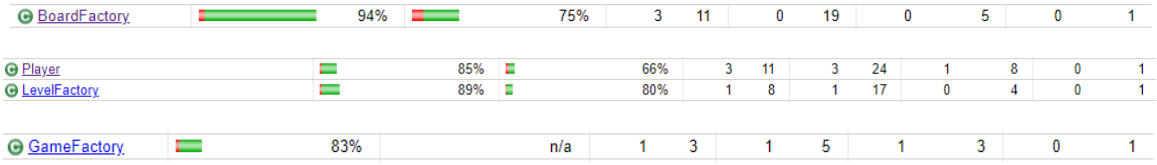
```
/*
 * - createSinglePlayerGame(Level, PointCalculator)
 *    > Mock Level
 *    > PointCalculatorLoader => PointCalculator
 */
1 usage
private final Level lvl = mock(Level.class);
1 usage
private final PointCalculatorLoader pcl = new PointCalculatorLoader();
1 usage
private final PointCalculator pc = pcl.load();

no usages  new *
@Test
void testCreateSinglePlay(){
    assertThat(gameFac.createSinglePlayerGame(lvl, pc)).isNotNull();
}
```

Task 3 - JaCoCo Report on JPacman

- Are the coverage results from JaCoCo similar to the ones you got from IntelliJ in the last task? Why so or why not?
    - Comparing the methods of my test cases I had done in Task 2.1, it is different. In the JaCoCo one the number of lines and methods are different compared to the one in IntelliJ. In the IntelliJ, my BoardFactory has 1/3 classes, 3/11 methods, and

17/29 lines while in JaCoCo 1 class, 5 methods, and 19 lines; it is the same story as the other classes.

- 


- Did you find helpful the source code visualization from JaCoCo on uncovered branches?
  - It looks nice having a visual gauge on what I may be missing or covered from my test cases so I believe it's a useful feature.
- Which visualization did you prefer and why? IntelliJ's coverage window or JaCoCo's report?
  - I preferred IntelliJ's coverage, having a list I could view alongside code that updates as I run, allowing me to access the code and see what has been covered and what hasn't. The JaCoCo looks nice and has many features but I believe the IntelliJ's built-in coverage is better than the JaCoCo's in my opinion.

Task 4 - Working with Python Test Coverage

- Run nosetests and produce coverage report

```
PS E:\Programming Projects\CS472\Software Testing\Python Test Coverage\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 1.796s

OK
```

- Ensure 26 is now covered

```
PS E:\Programming Projects\CS472\Software Testing\Python Test Coverage\test_coverage> nosetests

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

Name                 Stmts   Miss  Cover   Missing
--------------------------------------------------
models\__init__.py       7      0   100%
models\account.py       40     12    70%   30, 34-35, 45-48, 52-54, 74-75
--------------------------------------------------
TOTAL                   47     12    74%
----------------------------------------------------------------------
Ran 3 tests in 1.573s

OK
```

- Ensure 30 is now covered

```
PS E:\Programming Projects\CS472\Software Testing\Python Test Coverage\test_coverage> nosetests

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

Name                 Stmts   Miss  Cover   Missing
--------------------------------------------------
models\__init__.py       7      0   100%
models\account.py       40     11    72%   34-35, 45-48, 52-54, 74-75
--------------------------------------------------
TOTAL                   47     11    77%
----------------------------------------------------------------------
Ran 4 tests in 1.671s

OK
```

- Getting Coverage to 100%

```
PS E:\Programming Projects\CS472\Software Testing\Python Test Coverage\test_coverage> nosetests

Test Account Model
- Test creating multiple Accounts
- Test Account creation using known data
- Test removing an Account from the database
- Test Account to dict
- Test creating an Account from a dictionary
- Test the representation of an Account
- Test updating an Account in the database
- Test updating an Account with an empty ID

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 2.568s

OK
```

```python
def test_from_dict(self):
    """Test creating an Account from a dictionary"""
    data = ACCOUNT_DATA[self.rand] # get a random account
    account = Account()
    account.from_dict(data)
    self.assertEqual(account.name, data["name"])
    self.assertEqual(account.email, data["email"])
    self.assertEqual(account.phone_number, data["phone_number"])
    self.assertEqual(account.disabled, data["disabled"])

def test_update(self):
    """Test updating an Account in the database"""
    data = ACCOUNT_DATA[self.rand] # get a random account
    account = Account(**data)
    account.create()
    account.name = "John Smith"
    account.update()
    updated_account = Account.find(account.id)
    self.assertEqual(updated_account.name, "John Smith")
```

```python
def test_update_with_empty_id(self):
    """Test updating an Account with an empty ID"""
    data = ACCOUNT_DATA[self.rand] # get a random account
    account = Account(**data)
    with self.assertRaises(DataValidationError) as context:
        account.update()
    self.assertIn("Update called with empty ID field", str(context.exception))

def test_delete(self):
    """Test removing an Account from the database"""
    data = ACCOUNT_DATA[self.rand] # get a random account
    account = Account(**data)
    account.create()
    self.assertEqual(len(Account.all()), 1)
    account.delete()
    self.assertEqual(len(Account.all()), 0)
```

Task 5 - TDD

- ModuleNotFoundError

```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

class CounterTest(TestCase):
    """Counter Tests"""
```

```
ModuleNotFoundError: No module named 'src.counter'

Name             Stmts   Miss  Cover   Missing
------------------------------------------------
src\status.py        6      6     0%   2-7
------------------------------------------------
TOTAL                6      6     0%
------------------------------------------------------------------------------

FAILED (errors=1)
```

- ImportError

```
ImportError: cannot import name 'app' from 'src.counter' (E:\Programming Projects\CS472\Software Testing\tdd\src\counter.py)

Name            Stmts   Miss  Cover   Missing
-----------------------------------------------
src\counter.py      0      0   100%
-----------------------------------------------
TOTAL               0      0   100%
----------------------------------------------------------------------
Ran 1 test in 0.020s

FAILED (errors=1)
```

- Import Flask (No Error)

```python
from flask import Flask

app = Flask(__name__)
```

```
Name             Stmts   Miss  Cover   Missing
------------------------------------------------
src\counter.py       2      0   100%
src\status.py        6      0   100%
------------------------------------------------
TOTAL                8      0   100%
------------------------------------------------------------------------------
Ran 0 tests in 0.220s

OK
```

- AssertionError

```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)
```

```
Counter Tests
- It should create a counter (FAILED)

======================================================================
FAIL: It should create a counter
----------------------------------------------------------------------
Traceback (most recent call last):
  File "E:\Programming Projects\CS472\Software Testing\tdd\tests\test_counter.py", line 28, in test_create_a_counter
    self.assertEqual(result.status_code, status.HTTP_201_CREATED)
AssertionError: 404 != 201
```

- Create First Endpoint (Green)

```python
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
```

```
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 0.244s

OK
```

- Added setUp (Refactor) and duplicate_a_counter (Red)

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

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)
```

```
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 "E:\Programming Projects\CS472\Software Testing\tdd\tests\test_counter.py", line 38, in test_duplicate_a_counter
    self.assertEqual(result.status_code, status.HTTP_409_CONFLICT)
AssertionError: 201 != 409
```

- Refactored counter.py (Green)

```python
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
```

```
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 0.235s

OK
```

- Create test_update_a_counter (Red)

```python
def test_update_a_counter(self):
    """It should update the counter"""
    # 1. Create
    result = self.client.post('/counters/updateCount')
    self.assertEqual(result.status_code, status.HTTP_201_CREATED)
    # 2. Ensure Success
    baseline = self.client.get('/counters/updateCount')
    self.assertEqual(baseline.status_code, status.HTTP_200_OK)
    # 3. Check counter value as baseline
    baselineValue = baseline.json['updateCount']
    # 4. Call to Update counter
    update = self.client.put('/counters/updateCount')
    self.assertEqual(update.status_code, status.HTTP_200_OK)
    # 5. Ensure Success
    update = self.client.get('counters/updateCount')
    self.assertEqual(update.status_code, status.HTTP_200_OK)
    # 6. Check counter value is one more than baseline
    update = update.json['updateCount']
    self.assertEqual(update, baselineValue + 1)
```

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

======================================================================
FAIL: It should update the counter
----------------------------------------------------------------------
Traceback (most recent call last):
  File "E:\Programming Projects\CS472\Software Testing\tdd\tests\test_counter.py", line 48, in test_update_a_counter
    self.assertEqual(baseline.status_code, status.HTTP_200_OK)
AssertionError: 405 != 200
```

- Create update_counter (Red b/c need read_a_counter)

```python
@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
```

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

======================================================================
FAIL: It should update the counter
----------------------------------------------------------------------
Traceback (most recent call last):
  File "E:\Programming Projects\CS472\Software Testing\tdd\tests\test_counter.py", line 47, in test_update_a_counter
    self.assertEqual(baseline.status_code, status.HTTP_200_OK)
AssertionError: 405 != 200
```

- Create test_read_a_counter (Red)

```python
def test_read_a_counter(self):
    """It should read the counter value"""
    result = self.client.post('counters/readCount')
    self.assertEqual(result.status_code, status.HTTP_201_CREATED)
    read = self.client.get('/counters/readCount')
    self.assertEqual(read.status_code, status.HTTP_200_OK)
    value = read.json['readCount']
    self.assertEqual(value, 0)
```

```
Counter Tests
- It should create a counter
- It should return an error for duplicates
- It should read the counter value (FAILED)
- It should update the counter (FAILED)

======================================================================
FAIL: It should read the counter value
----------------------------------------------------------------------
Traceback (most recent call last):
  File "E:\Programming Projects\CS472\Software Testing\tdd\tests\test_counter.py", line 65, in test_read_a_counter
    self.assertEqual(read.status_code, status.HTTP_200_OK)
AssertionError: 405 != 200
---------------------- >> begin captured logging << ----------------------
src.counter: INFO: Request to create counter: foobar
----------------------- >> end captured logging << -----------------------

======================================================================
FAIL: It should update the counter
----------------------------------------------------------------------
Traceback (most recent call last):
  File "E:\Programming Projects\CS472\Software Testing\tdd\tests\test_counter.py", line 47, in test_update_a_counter
    self.assertEqual(baseline.status_code, status.HTTP_200_OK)
AssertionError: 405 != 200
```

- Create read_a_counter (Green) [Refactored to get 100% cover with not exist test cases]

```python
@app.route('/counters/<name>', methods=['GET'])
def read_a_counter(name):
    """Get a Counter Value"""
    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
```

```python
def test_read_a_counter_no_name(self):
    """It should attempt to read a counter that does not exist"""
    result = self.client.get('/counters/falseCount')
    self.assertEqual(result.status_code, status.HTTP_404_NOT_FOUND)

def test_update_a_counter_no_name(self):
    """It should attempt to update a counter that does not exist"""
    result = self.client.put('/counters/falseCount')
    self.assertEqual(result.status_code, status.HTTP_404_NOT_FOUND)
```

```
Counter Tests
- It should create a counter
- It should return an error for duplicates
- It should read the counter value
- It should attempt to read a counter that does not exist
- It should update the counter
- It should attempt to update a counter that does not exist

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 0.245s

OK
```