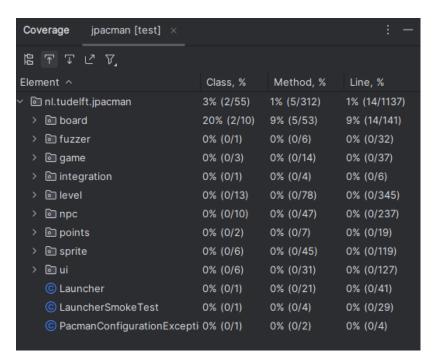
# **Unit Testing Report**

# **Table of Contents**

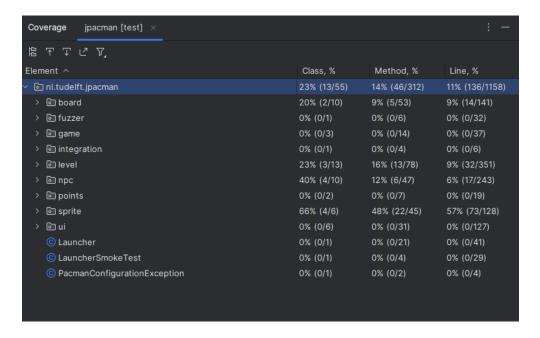
Unit Testing Report		1
Task 2.1: Java unit tes	ts	2
Task 3: JaCoCo		4
Task 4: Python		4
Task 5: TDD		6
Red Phase: Writing a	failing test according to requirements	6
Green Phase: Writing	minimum amount of code to pass test	6
Refactoring: Cleaning	up code with the assurance of our test	7
Exceptions and Errors	Encountered with nosetests	7

#### Task 2.1: Java unit tests

Original coverage before adding code.



Coverage after adding my unit tests. Improvement in level, npc and sprite packages



#### I decided to work on these Unit Tests

Kamil Dusejovsky	src/main/java/nl/tudelft/jpacman/level/Player.setAlive
Kamil Dusejovsky	src/main/java/nl/tudelft/jpacman/level/Player.setKiller
Kamil Dusejovsky	src/main/java/nl/tudelft/jpacman/level/Player.addPoints
Kamil Dusejovsky	src/main/java/nl/tudelft/jpacman/level/Pellet.getValue
Kamil Dusejovsky	src/main/java/nl/tudelft/jpacman/level/Pellet.getSprite

Here is a code snippet of tests in package Pellet for methods getValue and getSprite

```
public class PelletTest {
    Sprite newSprite = new EmptySprite();
    Pellet newPellet = new Pellet(10, newSprite);

    @Test
    void testGetValue() {
        assertThat(newPellet.getValue()).isEqualTo(10);
    }

    @Test
    void testGetSprite() {
        assertThat(newPellet.getSprite()).isEqualTo(newSprite);
    }
}
```

and here is snippet for package Player method addPoints

```
@Test
void testAddPoints() {
    assertThat(object_player.getScore()).isEqualTo(0);
    object_player.addPoints(10);
    assertThat(object_player.getScore()).isEqualTo(10);
}
```

# Task 3: JaCoCo

JaCoCo coverage is slightly different from IntelliJ beause JaCoCo work on byteode level.
 IntelliJ interprets some code structures differently from JaCoCo

Here is an example of missed branch in getSprite() I missed the false branch of if statement

```
111.
        }
112.
        @Override
113.
114.
115. �
        public Sprite getSprite() {
         if (isAlive())
116. return sprites.get(getDirection());
117.
        return deathSprite;
118.
119.
      }
120.
121.
```

#### **Player**

Element	Missed Instructions	Cov. \$	Missed Branches	Cov. \$	Missed \$	Cxty \$	Missed	Lines \$	Missed \$	Methods *
<ul><li>getSprite()</li></ul>		76%		50%	1	2	1	3	0	1
<ul> <li>Player(Map_AnimatedSprite)</li> </ul>		100%		n/a	0	1	0	7	0	1
<ul><li>setAlive(boolean)</li></ul>		100%		100%	0	3	0	7	0	1
<ul><li>addPoints(int)</li></ul>		100%		n/a	0	1	0	2	0	1
<ul><li>setKiller(Unit)</li></ul>		100%		n/a	0	1	0	2	0	1
<ul><li>isAlive()</li></ul>		100%		n/a	0	1	0	1	0	1
<ul><li>getKiller()</li></ul>		100%		n/a	0	1	0	1	0	1
getScore()		100%		n/a	0	1	0	1	0	1
Total	3 of 70	95%	1 of 6	83%	1	11	1	24	0	8

- 2. The source code visualization from JaCoCo is very helpful as example showed above.
- 3. I prefer JaCoCo because it makes it easier to see where my tests missed lines of code.

# Task 4: Python

Here is the original test coverage.

Here is the final test coverage.

```
(hw2) UNLV-Student@LON-2G07469 test_coverage % nosetests
  Test creating multiple Accounts
  Test Account creation using known data
  Test account deletion
  Test account search
  Test dict to account
  Test the representation of an account
  Test account update
                   Stmts Miss Cover Missing
Name
models/__init__.py 7 0 100%
models/account.py 40 0 100%
models/account.py 40 0
TOTAL
                     47 0 100%
Ran 8 tests in 0.181s
0K
(hw2) UNLV-Student@LON-2G07469 test_coverage %
```

#### Code snippets:

```
def test_from_dict(self):
    """ Test dict to account """
    data = ACCOUNT_DATA[self.rand]
    account = Account ("*data)
    data = {
        'name': 'Spider man',
        'email': 'spider@man.com',
        'phone_number': '9996669990',
        'disabled': False,
        'date_joined': '2011-01-01'
}
account.from_dict(data)
self.assertEqual(account.name, 'Spider man')
self.assertEqual(account.email, 'spider@man.com')
self.assertEqual(account.disabled, False)
self.assertEqual(account.disabled, False)
self.assertEqual(account.disabled, False)
self.assertEqual(account.date_joined, '2011-01-01')

def test_update(self):
    """ Test account update"""
    data = ACCOUNT_DATA[self.rand]
    account.id = Iiilil
try:
        account.update()
        except DataValidationError:
        self.fail("DataValidationError in account update")
        account.id = None
        exception = False
try:
        account.update()
except DataValidationError:
        exception = True
        self.assertEqual(exception, True)

def test_delete(self):
    """ Test account deletion"""
        data = ACCOUNT_DATA[self.rand]
        account.oreate()
        account.oreate()
        account.oreate()
        account.ercate()
        account.ercate()
```

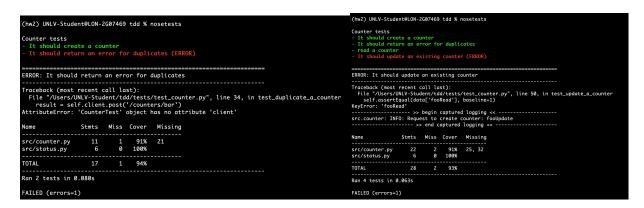
## Kamil Dusejovsky

```
def test_find(self):
    """ Test account search"""
    data = ACCOUNT_DATA[self.rand]
    account = Account(**data)
    account.create()
    accountFound = Account.find(account.id)
    print(accountFound)
    print(account.id)
    self.assertEqual(account.name, accountFound.name)
```

#### Task 5: TDD

In this phase we focus on Test Driven Development which is separated by 3 phases.

# Red Phase: Writing a failing test according to requirements



## Green Phase: Writing minimum amount of code to pass test.

#### Refactoring: Cleaning up code with the assurance of our test

```
@app.route('/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

@app.route('/counters/<name>', methods=['PUT'])
def update_counter(name):
    if name not in COUNTERS:
        return {"Message": f"Counter {name} doesnt exists"},

status.HTTP_404_NOT_FOUND
    COUNTERS[name] += 1
    return {name: COUNTERS[name]}, status.HTTP_200_OK

@app.route('/counters/<name>', methods=['GET'])
def read_counter(name):
    if name not in COUNTERS:
        return {"Message": f"Counter {name} doesnt exists"},

status.HTTP_404_NOT_FOUND
    return {name: COUNTERS[name]}, status.HTTP_200_OK
```

#### Exceptions and Errors Encountered with nosetests

- AssertionError: 404! =201
  - o 404 http code NOT FOUND returned, but test expected 201 CREATED
- AttributeError: 'CounterTest' object has no attribute 'client'
  - O During RED phase error occurred due to no code in counter.py
- AssertionError: 409! = 201
  - o 409 CONFLICT http code returned but test expected 21 CREATED
- AttributeError: 'FlaskClient' object has no attribute 'update\_counter'
  - o During RED phase error occurred due to no code in counter.py