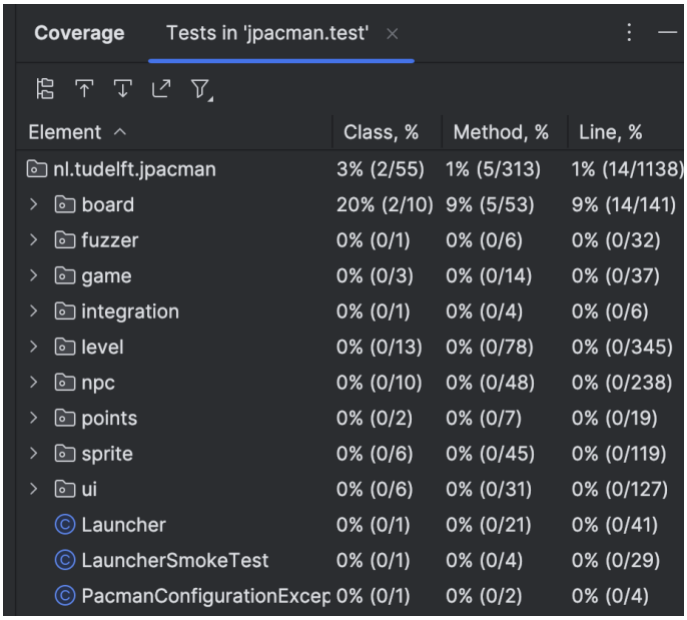
Unit Testing and Coverage Analysis in JPacman

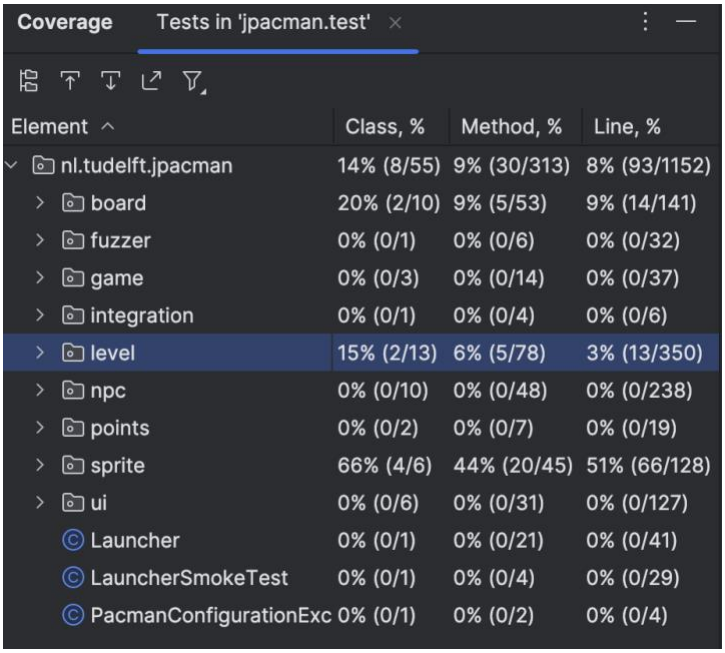
https://github.com/kzmaybe/2DRogueLikeUnityGame.git

Task 1



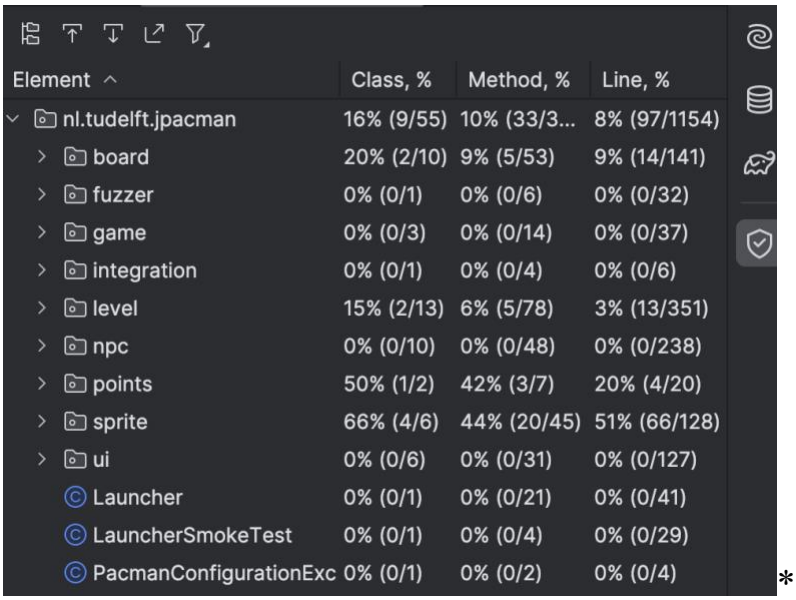
Ans: The unit test only covers 3% of the Class, 1% of the Method and Line, which is not good enough.

Task 2



*added PlayerTest

Task 2.1



Element	Class, %	Method, %	Line, %
nl.tudelft.jpacman	16% (9/55)	10% (33/3...	8% (97/1154)
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% (13/351)
npc	0% (0/10)	0% (0/48)	0% (0/238)
points	50% (1/2)	42% (3/7)	20% (4/20)
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)

*added Points test

```
new *
@Test
void testCollidedWithAGhost() {
    pointCalculator.collidedWithAGhost(player, ghost);
    // Verify no points are added
    verify(player, never()).addPoints(anyInt());
}
new *
@Test
void testConsumedAPellet() {
    when(pellet.getValue()).thenReturn( t: 10);
    pointCalculator.consumedAPellet(player, pellet);
    // Verify points are added correctly
    verify(player).addPoints(10);
}
new *
@Test
void testPacmanMoved() {
    pointCalculator.pacmanMoved(player, Direction.NORTH);
    // Verify no points are added
    verify(player, never()).addPoints(anyInt());
}
```

Test Methods Added:

- testCollidedWithAGhost: Ensures no points are added when colliding with a ghost.

- testConsumedAPellet: Confirms that points are added correctly when a pellet is consumed.
- testPacmanMoved: Vertifies that no points are added when Pacman moves.

Task 3

- Ans1:
  The coverage results from the JaCoCo report indicate a 73% instruction coverage, in contrast to the 8%-line coverage observed in the IntelliJ report from Task 2.1. The superior coverage reported by JaCoCo can be attributed to the broader scope of testing it encompasses. In contrast, the tests implemented during Task 2.1 were limited to only four functions, which causes a large portion of the codebase remains untested.

- Ans2

  The source code visualization provided by JaCoCo for uncovering branches was indeed helpful. This feature allows for an at-a-glance assessment of which code paths have been exercised by the test suite and which have not, visually distinguishing between executed and missed branches. Such direct feedback within the context of the source code is crucial for identifying untested parts of the codebase, enabling targeted improvements in test coverage.

- Ans3
  I prefer JaCoCo's report because it provides a comprehensive visual representation of the coverage through charts. Additionally, JaCoCo offers the advantage of displaying the source code with annotations that clearly indicate which sections of code have been covered by tests, enhancing the overall utility of the coverage analysis.

Task 4

Before adding tests:

```
PS C:\Users\Kevin\Desktop\lab\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 0.430s

OK
```

Added the provided test in the document:

```
PS C:\Users\Kevin\Desktop\lab\test_coverage> nosetests

Test Account Model
- Test creating multiple Accounts
- Test Account creation using known data
- 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     11    72%   34-35, 45-48, 52-54, 74-75
--------------------------------------------------
TOTAL                   47     11    77%
----------------------------------------------------------------------
Ran 4 tests in 0.456s

OK
```

Implement the test:

```python
#my code
def test_from_dict(self):
    """Test setting attributes from a dictionary"""
    account = Account()
    data = {"name": "Foo", "email": "foo@example.com"}
    account.from_dict(data)
    self.assertEqual(account.name, "Foo")
    self.assertEqual(account.email, "foo@example.com")

#
@staticmethod
def create_and_return_account_with_id():
    """
    """
    new_account = Account(name="Test User", email="testuser@example.com")
    db.session.add(new_account)
    db.session.commit()
    return new_account

def test_update_success(self):
    """Test successful account update"""
    account = self.create_and_return_account_with_id()
    account.name = "New Name"
    account.update()
    updated_account = Account.find(account.id)
    self.assertEqual(updated_account.name, "New Name")

def test_update_no_id(self):
    """Test update with no ID"""
    account = Account(name="Foo")
    with self.assertRaises(DataValidationError):
        account.update()

def test_delete(self):
    """Test deleting an account"""
    account = self.create_and_return_account_with_id()
    account_id = account.id
    account.delete()
    self.assertIsNone(Account.find(account_id))

def test_find_existing_account(self):
    """Test finding an existing account by ID"""
    account = self.create_and_return_account_with_id()
    found_account = Account.find(account.id)
    self.assertEqual(found_account, account)

def test_find_non_existing_account(self):
    """Test finding a non-existing account"""
    self.assertIsNone(Account.find(99999))
```

Final result:

```
PS C:\Users\Kevin\Desktop\lab\test_coverage> nosetests

Test Account Model
- Test creating multiple Accounts
- Test Account creation using known data
- Test deleting an account
- Test finding an existing account by ID
- Test finding a non-existing account
- Test setting attributes from a dictionary
- Test the representation of an account
- Test account to dict
- Test update with no ID
- Test successful account update

Name                 Stmts   Miss  Cover   Missing
--------------------------------------------------
models\__init__.py       7      0   100%
models\account.py       40      0   100%
--------------------------------------------------
TOTAL                   47      0   100%
----------------------------------------------------------------------
Ran 10 tests in 0.515s

OK
```

Task 5

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

Name           Stmts   Miss  Cover   Missing
--------------------------------------------
src\status.py      6      6     0%   2-7
--------------------------------------------
TOTAL              6      6     0%
----------------------------------------------------------------------
Ran 1 test in 0.010s

FAILED (errors=1)
```

```
ImportError: cannot import name 'app' from 'src.counter' (

Name              Stmts   Miss  Cover   Missing
-----------------------------------------------
src\counter.py        0      0   100%
-----------------------------------------------
TOTAL                 0      0   100%
----------------------------------------------------------------------
Ran 1 test in 0.009s

FAILED (errors=1)
```

```
PS C:\Users\Kevin\Desktop\lab\tdd> nosetests

Counter tests
- It should create a counter (FAILED)

======================================================================
FAIL: It should create a counter
----------------------------------------------------------------------
Traceback (most recent call last):
  File "C:\Users\Kevin\Desktop\lab\tdd\tests\test_counter.py", line 27, 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      2      0   100%
src\status.py       6      0   100%
-----------------------------------------------
TOTAL               8      0   100%
----------------------------------------------------------------------
Ran 1 test in 0.163s
```

```
PS C:\Users\Kevin\Desktop\lab\tdd> nosetests

Counter tests
- It should create a counter

Name              Stmts   Miss  Cover   Missing
-------------------------------------------------
src\counter.py        2      0   100%
src\status.py         6      0   100%
-------------------------------------------------
TOTAL                 8      0   100%
-------------------------------------------------------------
Ran 1 test in 0.168s

OK
```

```
PS C:\Users\Kevin\Desktop\lab\tdd> nosetests

Counter
- 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 "C:\Users\Kevin\Desktop\lab\tdd\tests\test_counter.py", line 53, 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      2      0   100%
src\status.py       6      0   100%
-----------------------------------------------
TOTAL               8      0   100%
----------------------------------------------------------------------
Ran 2 tests in 0.165s

FAILED (failures=1)
```

```
PS C:\Users\Kevin\Desktop\lab\tdd> nosetests

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

Name              Stmts   Miss  Cover   Missing
-------------------------------------------------
src\counter.py        2      0   100%
src\status.py         6      0   100%
-------------------------------------------------
TOTAL                 8      0   100%
-------------------------------------------------------------
Ran 2 tests in 0.169s

OK
```

Added test_update_a_counter

```
PS C:\Users\Kevin\Desktop\lab\tdd> nosetests

Counter
- 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 "C:\Users\Kevin\Desktop\lab\tdd\tests\test_counter.py", line 68, in test_update_a_counter
    self.assertEqual(update_response.status_code, status.HTTP_200_OK)
AssertionError: 405 != 200
-------------------- >> begin captured logging << --------------------
src.counter: INFO: Request to create counter: testcounter
--------------------- >> end captured logging << ---------------------

Name             Stmts   Miss  Cover   Missing
------------------------------------------------
src\counter.py       2      0   100%
src\status.py        6      0   100%
------------------------------------------------
TOTAL                8      0   100%
----------------------------------------------------------------------
Ran 3 tests in 0.164s

FAILED (failures=1)
```

Added `update_counter(name)`

```
PS C:\Users\Kevin\Desktop\lab\tdd> nosetests

Counter
- 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       2      0   100%
src\status.py        6      0   100%
------------------------------------------------
TOTAL                8      0   100%
----------------------------------------------------------------------
Ran 3 tests in 0.166s

OK
```

```python
def test_update_a_counter(self):
    """It should update a counter"""
    # Create a new counter
    create_response = self.client.post('/counters/testcounter')
    self.assertEqual(create_response.status_code, status.HTTP_201_CREATED)

    # Get baseline counter value
    baseline_value = create_response.json['testcounter']

    # Update the counter
    update_response = self.client.put('/counters/testcounter')
    self.assertEqual(update_response.status_code, status.HTTP_200_OK)

    # Check that the counter value has increased by 1
    updated_value = update_response.json['testcounter']
    self.assertEqual(updated_value, baseline_value + 1)
```

```python
@app.route('/counters/<name>', methods=['PUT'])
def update_counter(name):
    """Update a counter"""
    global COUNTERS
    if name not in COUNTERS:
        # If the counter doesn't exist, return a 404 Not Found error
        return {"Message": f"Counter {name} does not exist"}, status.HTTP_404_NOT_FOUND

    # Increment the counter by 1
    COUNTERS[name] += 1

    # Return the updated counter value and a 200 OK status
    return {name: COUNTERS[name]}, status.HTTP_200_OK
```