Task 2

Coverage Tests in 'jpacman.test' ×			
電 不 Ţ ピ ∇,			
Element ^	Class	Method	Line, %
∨ 🖻 nl.tudelft.jpacman	14% (8	9% (30/3	8% (93/11
> 🗈 board	20% (2	9% (5/53)	9% (14/141)
> 🗈 fuzzer	0% (0/1)	0% (0/6)	0% (0/32)
> <b>i</b> game	0% (0/3)	0% (0/14)	0% (0/37)
> 🗈 integration	0% (0/1)	0% (0/4)	0% (0/6)
> 🗈 level	15% (2	6% (5/78)	3% (13/350)
>	0% (0/	0% (0/47)	0% (0/237)
> 🖻 points	0% (0/2)	0% (0/7)	0% (0/19)
> 🗈 sprite	66% (4	44% (20/	51% (66/1
>	0% (0/6)	0% (0/31)	0% (0/127)
© Launcher	0% (0/1)	0% (0/21)	0% (0/41)
© LauncherSmokeT	e: 0% (0/1)	0% (0/4)	0% (0/29)
PacmanConfigura	iti 0% (0/1)	0% (0/2)	0% (0/4)

Original Test Coverage

# Task 2.1 getDeltaX()

```
public class DirectionTest {
    /**
    * Do we get the correct delta when moving north?
    */
    @Test
    void testNorth() {
        Direction north = Direction.valueOf( name: "NORTH");
        assertThat(north.getDeltaY()).isEqualTo( expected: -1);
    }

@Test
    void testSouth() {
        Direction north = Direction.valueOf( name: "SOUTH");
        assertThat(north.getDeltaY()).isEqualTo( expected: 1);
    }

@Test
    void testEast() {
        Direction east = Direction.valueOf( name: "EAST");
        assertThat(east.getDeltaX()).isEqualTo( expected: 1);
    }

@Test
    void testWest() {
        Direction east = Direction.valueOf( name: "WEST");
        assertThat(east.getDeltaX()).isEqualTo( expected: -1);
    }
}
```

The first thing I decided to test was to try and look at the other directions for DirectionTest. I realized that it did not matter what value (aka the four directions) was ensured to increase test coverage rather than just the method. The board method coverage went from 9% to 11%, and the line percent went from 9% to 10%. I ensured the value gotten from getDeltaX() is equal to the east or west direction.

Task 2.1 start()

```
| Import | I
```

ackage nl.tudelft.jpacman.level;

Learning this new thing of using methods to increase test coverage, I decided to test start(), ensuring when a level is started in progress is true. This tests a number of things, creating level objects, retrieving board, ghost, and square, collision objects. With those, I started a new level, creating arrays, using the Lists class. Finally I started the level and ensured it is in progress aka true. The test coverage went up way more than the previous, the entire class percentage went up from 14% to 21%, and the board class percentage went up from 20% to 40%. The level coverage essentially doubled in each category, except in line going from 3% to 19%.

# Task 2.1 createBoard()

```
private Baceflactory factory;

20 usages
private Square s1;

15 usages
private Square s2;

27 est

15 usages
private Square s2;

28 est

15 usages
private Square s2;

28 est

29 est

20 usages
private Square s2;

29 est

20 usages
private Square s2;

29 est

20 usages
private Square s2;

20 est

20 usages
private Square s2;

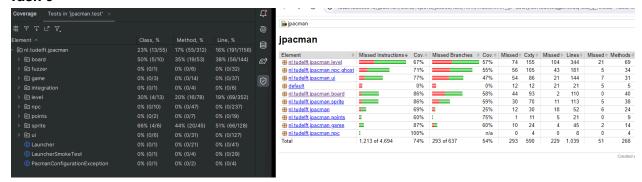
20 est

21 est

22 est BasicSquare();
factory - reareBasicSquare();
f
```

The final test I did was createBoard(). Learning from level, I realized I have to find all the elements to execute a function and execute that function. This is testing a bunch of methods such as creating a square, ensuring different placements of squares. The class test coverage had gone up from 21% to 23%. The board percents from 40% to 50% in class, 15% to 35% in method, 11 to 38% in lines.

#### Task 3



I got way more coverage in jcoco than intellej. It is strange because jcoco counts way more lines than intellej. For example, stop is counted fully in jcoco but not in intellej. My hypothesis is that jcoco tracks every line a code went over in a test, while intellej only tracks the methods used in the test code.

Example: My start() code had to eventually terminate the game, so jcoco tracks it, while in intellej I didnt explicitly stop() so it does not track it.

```
public void stop() {
                                                            * Stops or pauses this level, no lo
                                                  213.
                                                            * and stopping all NPCs.
                                                  214.
    synchronized (startStopLock) {
                                                  215.
        if (!isInProgress()) {
                                                           public void stop()
                                                  216.
                                                               synchronized (startStopLock) {
                                                  217.
                                                  218.
                                                                   if (!isInProgress()) {
                                                  219.
                                                                       return:
                                                  220.
        stopNPCs();
                                                  221.
                                                                   stopNPCs();
        inProgress = false:
                                                  222.
                                                                   inProgress = false;
                                                  223.
                                                  224.
                                                  225.
```

I find the source code visualization extremely useful on tracking uncovered branches, breaking down code by methods and each method by the line and if it was tested. I enjoy how a lot of color is in my face telling me what I have and have not tested

I much prefer jcoco over the intellej! The jacpacman allowed me to go line by line much easier than jacoco, as the entire lines are highlighted even showing red and yellow for the ones not yet done. However, I think I am suspicious of jcoco of its interestingly high coverage

### Task 4

```
Test Account Model
- Test creating multiple Accounts
- Test Account creation using known data
- Test account to delete
- Test account to find name
- Test account to fromdict
- Test the representation of an account
- Test account to update successfully
- Test account to updated FAIL
       Stmts Miss Cover Missing
Name
models\__init__.py 7 0 100%
models\account.py 40 0 100%
              47 0 100%
TOTAL
Ran 9 tests in 0.595s
0K
```

```
def test_find(self):
    """ Test account to find name"""
    data = ACCOUNT_DATA[self.rand] # get a random account
    account = Account(**data)
    account.create()
    test_account = Account.find(account.id)
    result = test_account.name
    self.assertEqual(result, account.name)
```

```
def test_update(self):
   data = ACCOUNT_DATA[self.rand] # get a random account
    account = Account(**data)
   account.create()
   testName = "name"
    account.name = testName
    account.update()
   new_account = Account.find(account.id)
    self.assertEqual(new_account.name, testName)
def test_update_fail(self):
   data = ACCOUNT_DATA[self.rand] # get a random account
   account = Account(**data)
    with self.assertRaises(DataValidationError):
        account.update()
def test_from_dict(self):
    """ Test account to fromdict """
   data = ACCOUNT_DATA[self.rand] # get a random account
   account = Account(**data)
   result = account.to_dict()
   test_account = Account(**data)
    test_account.from_dict(result)
    self.assertEqual(account.name, test_account.name)
    self.assertEqual(account.email, test_account.email)
    self.assertEqual(account.phone_number, test_account.phone_number)
    self.assertEqual(account.disabled, test_account.disabled)
    self.assertEqual(account.date_joined, test_account.date_joined)
```

### Task 5

```
∨ 🗀 src
                               def test_update_a_counter(self):
       ≡ counter.py

≡ status.py

                                   result = self.client.post('/counters/update')
                                   self.assertEqual(result.status_code, status.HTTP_201_CREATED)

∨ □ tests

                                   self.assertEqual(result.json['update'], θ)
   > 🗀 venv
                                   result = self.client.put('/counters/update')
     coverage .
                                   self.assertEqual(result.status_code, status.HTTP_200_0K)
     M↓ README.md
                                   self.assertEqual(result.json['update'], 1)
     ≡ requirements.tx
     ≡ setup.cfg
                                   result = self.client.put('/counters/doesNotExist')
   self.assertEqual(result.status_code, status.HTTP_204_NO_CONTENT)
 > Properties and Con
Terminal
          Local ×
- It should return an error for duplicates
FAIL: It should update counter
Traceback (most recent call last):
 File "C:\Users\neonf\Documents\tdd\tdd\tests\test_counter.py", line 47, in test_update_a_counter
   self.assertEqual(result.status_code, status.HTTP_200_0K)
AssertionError: 405 != 200
------->>> begin captured logging << ----------
src.counter: INFO: Request to create counter: update
  ------>>> end captured logging << ------>>
              Stmts Miss Cover Missing
                 11 0 100%
6 0 100%
src\counter.py
src\status.py
```

Created test\_update\_a\_counter(self): that creates a counter, ensure it came back created, make sure it is baseline, updates, ensure successful code is returned, then make sure it is more than 0. If there is an item that does not exist there is an error I got RED phase, with the error AssertionError: 405 != 200

REFACTOR counter, by updating the counter to add one if it is found. If it is not found an error is returned
Went to GREEN

```
def test_read_a_counter(self):

✓ □ src

        ≡ counter.py
                                     result = self.client.post('/counters/read')

≡ status.py

                                     self.assertEqual(result.status_code, status.HTTP_201_CREATED)

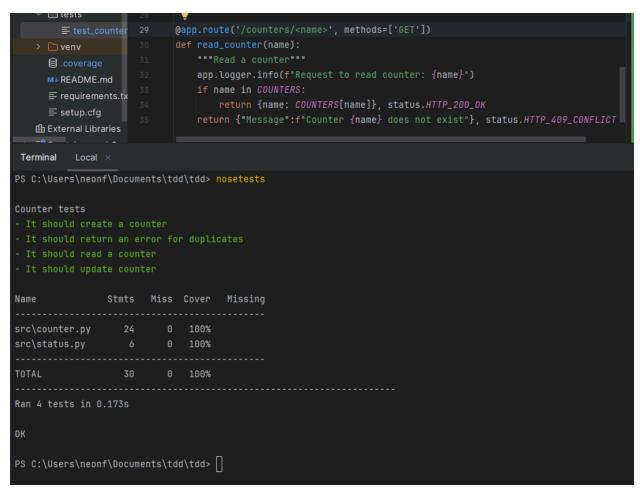
∨ □ tests

                                     result = self.client.get('/counters/read')
                                     self.assertEqual(result.status_code, status.HTTP_200_0K)
      result = self.client.get('/counters/doesNotExist')
      M↓ README.md
                                     self.assertEqual(result.status_code, status.HTTP_409_CONFLICT)
      ≡ requirements.tx
 Terminal Local ×
PS C:\Users\neonf\Documents\tdd\tdd> nosetests
Counter tests
 It should read a counter (FAILED)
Traceback (most recent call last):
   \begin{tabular}{ll} File & $\underline{C:\Users\setminus neonf\setminus Documents\setminus tdd\setminus test\_counter.py}^*, & line & 59, & in & test\_read\_a\_counter.py^*, \\ \end{tabular} 
    self.assertEqual(result.status_code, status.HTTP_200_0K)
AssertionError: 405 != 200
------ >> begin captured logging << ------
src.counter: INFO: Request to create counter: read
 Stmts Miss Cover Missing
src\status.py
                          0 100%
Ran 4 tests in 0.156s
FAILED (failures=1)
```

### **RED**

I created a read counter that creates a counter, then tries to get it successfully, then tries to get a counter unsuccessfully

I got error AssertionError: 405 != 200



# **REFACTOR**

I created a function that returns the counter if it is found, If it is not found, I return an error There i got GREEN