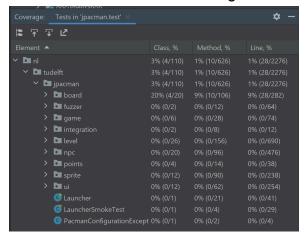
Yidong Fang CS 472 Lab Testing

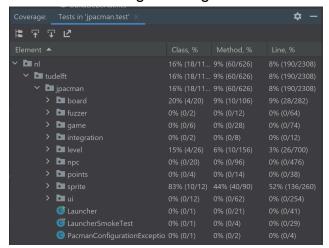
https://github.com/Thienguen/Barbell/tree/main/jpacman

Task 1 – JPacman Test Coverage



Note that coverage in this test is extremely bad, it covered very few lines.

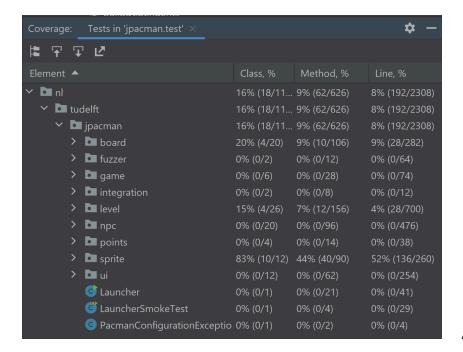
Task 2 – Increasing Coverage on JPacman



Task 2.1 testGetKiller:

```
@Test
void testGetKiller() {

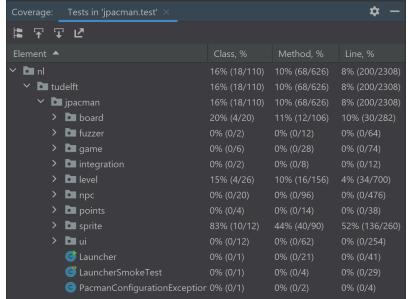
    if (player.isAlive())
        assertThat(player.getKiller()).isNull();
    else
        assertThat(player.getKiller() instanceof Unit).isEqualTo( expected: true);
}
```



After testGetKiller

testCreatePacMan:



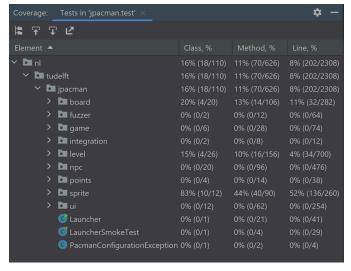


After testCreatePacMan

testGetDeltaX:

```
new*
@Test
void testGetDeltaX() {
    Direction dirN = Direction.NORTH;
    Direction dirW = Direction.WEST;
    Direction dirE = Direction.EAST;
    Direction dirS = Direction.SOUTH;

    assertThat(dirN.getDeltaX()).isEqualTo( expected: 0);
    assertThat(dirW.getDeltaX()).isEqualTo( expected: -1);
    assertThat(dirE.getDeltaX()).isEqualTo( expected: 1);
    assertThat(dirS.getDeltaX()).isEqualTo( expected: 0);
}
```



Task 3 – JaCoCo Report on JPacman jpacman

Element	Missed Instructions	Cov. \$	Missed Branches		Missed	Cxty	Missed =	Lines	Missed N	/lethods	Missed \$	Classes
nl.tudelft.jpacman.level		67%		57%	73	155	103	344	20	69	4	12
nl.tudelft.jpacman.npc.ghost		71%		55%	56	105	43	181	5	34	0	8
nl.tudelft.jpacman.ui		78%		47%	54	86	21	144	7	31	0	6
<u> </u>	=	0%	=	0%	12	12	21	21	5	5	1	1
nl.tudelft.jpacman.sprite		86%	_	59%	30	70	11	113	5	38	0	5
nl.tudelft.jpacman.board		86%		58%	44	93	2	110	0	40	0	7
nl.tudelft.jpacman	-	67%		25%	12	30	18	52	6	24	1	2
nl.tudelft.jpacman.points		59%	1	75%	1	11	5	21	0	9	0	2
nl.tudelft.jpacman.game		87%	-	60%	10	24	4	45	2	14	0	3
nl.tudelft.jpacman.npc	1	100%		n/a	0	4	0	8	0	4	0	1
Total	1,242 of 4,755	73%	293 of 637	54%	292	590	228	1,039	50	268	6	47

 The coverage results from JaCoCo has a more interactive and visual interface compared to gradlew results from IntelliJ. The biggest difference is that JaCoCo offers branch coverage visualization, which is more intuitive than just the line coverage percentage. The JaCoCo counts assembly-level instructions, and gradlew counts source code lines.

- I found the source code visualization from JaCoCo very helpful, it clearly tells me
 which branches are not covered and the source code displayed explains why.
- I prefer JaCoCo because it's the easiest way to achieve 100% coverage.

Task 4 – Working with Python Test Coverage

```
new*
def test_from_dict(self):
    accdata = ACCOUNT_DATA[len(ACCOUNT_DATA) - 1]
    acc = Account()
    acc.from_dict(accdata)
    for key, val in accdata.items():
        self.assertEqual(getattr(acc, key), val)

new*

def test_delete(
    data = ACCOUNT_DATA[self.cac, key), val)

acc.delete()
    self.assertE

data = ACCOUNT_DATA[self.rand]
    acc = Account(**data)
    acc.create()
    tmp = Account(**data)
    acc.update()
    self.assertEqual(acc.name, tmp.name)

acc.id = False
    with self.assertRaises(models.account.DataValidationError):
    acc.update()
    self.assertE
```

```
new*
def test_delete(self):
    data = ACCOUNT_DATA[self.rand]
    acc = Account(**data)
    acc.create()

    acc.delete()
    self.assertEqual(len(acc.all()), 0)

new*
def test_find(self):
    data = ACCOUNT_DATA[self.rand]
    acc = Account(**data)
    acc.create()
    data = ACCOUNT_DATA[len(ACCOUNT_DATA) - 1]
    account = Account(**data)
    account.create()

self.assertEqual(account.find(acc.id), acc)
```

Task 5 - TDD Test_update_a_counter

Test:

```
new *
def test_update_a_counter(self):
    self.setUp()
    returned = self.client.post('/counters/update')
    self.assertEqual(returned.status_code, status.HTTP_201_CREATED)

returned = self.client.put('/counters/update')
    self.assertEqual(returned.status_code, status.HTTP_200_OK)
    self.assertEqual(COUNTERS.get('update'), 1)
```

update_counter

```
new*

@app.route('/counters/<name>', methods=['PUT'])

def update_counter(name):
    app.logger.info(f"Request to update counter: {name}")
    COUNTERS[name] += 1
    return {name: COUNTERS[name]}, status.HTTP_200_OK
```

```
PS C:\Users\Aaron's\OneDrive\Documents\GitHub\tdd> nosetests

Counter tests

It should create a counter

It should return an error for duplicates

update a counter

Name Stmts Miss Cover Missing

src\counter.py 16 0 100%

src\status.py 6 0 100%

TOTAL 22 0 100%

Ran 3 tests in 0.172s
```

```
test_read__a_counter red
```

test_read__a_counter green/refactor

```
new *
@app.route('/counters/<name>', methods=['GET'])

def read_counter(name):
    app.logger.info(f"Request to read counter: {name}")
    return {name: COUNTERS[name]}, status.HTTP_200_0K
```