

Task 2.1 - Tests Files Created

The screenshot displays an IDE with three tabs: `withinBoardTest.java`, `SetDirectionTest.java`, and `createInkyTest.java`. Below the tabs, a 'Coverage: Tests in 'jpacman.test'' window shows a table with the following data:

Element	Class, %	Method, %	Line, %
nl	3% (4/110)	1% (10/624)	1% (28/2274)

The main editor shows the `testCreateInky()` method in `createInkyTest.java`:

```

@Test
void testCreateInky(){
    sprites = Mockito.mock(PacManSprites.class);
    ghost_factory = new GhostFactory(sprites);
    assertTrue(ghost_factory.createInky() != null);
}
    
```

On the right, a detailed coverage table is visible:

Element	Class, %	Method, %	Line, %
board	50% (10/20)	39% (42/107)	40% (118/295)
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	15% (4/26)	6% (10/156)	3% (26/700)
npc	40% (8/20)	12% (12/94)	6% (34/486)
points	0% (0/4)	0% (0/14)	0% (0/38)
sprite	83% (10/12)	44% (40/90)	52% (136/261)
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)

At the bottom, the 'Test Results' window shows:

```

Tests passed: 47 of 47 tests - 10 sec 286 ms
> Task :jacocoTestReport

Deprecated Gradle features were used in this build, making it incompatible with Gradle 6.0.
Use '--warning-mode all' to show the individual deprecation warnings.
See https://docs.gradle.org/5.3/userguide/command_line_interface.html#sec:command_line_warnings
BUILD SUCCESSFUL in 27s
9 actionable tasks: 4 executed, 5 up-to-date
11:58:51 PM: Execution finished 'test'.
    
```

Element	Missed Instructions	Cov.	Missed Branches	Cov.
nl.tudelft.jpacman.level	<div><div></div></div>	67%	<div><div></div></div>	58%
nl.tudelft.jpacman.npc.ghost	<div><div></div></div>	71%	<div><div></div></div>	55%
nl.tudelft.jpacman.ui	<div><div></div></div>	77%	<div><div></div></div>	47%
default	<div><div></div></div>	0%	<div><div></div></div>	0%
nl.tudelft.jpacman.board	<div><div></div></div>	86%	<div><div></div></div>	58%
nl.tudelft.jpacman.sprite	<div><div></div></div>	88%	<div><div></div></div>	62%
nl.tudelft.jpacman	<div><div></div></div>	69%	<div><div></div></div>	25%
nl.tudelft.jpacman.points	<div><div></div></div>	60%	<div><div></div></div>	75%
nl.tudelft.jpacman.game	<div><div></div></div>	87%	<div><div></div></div>	60%
nl.tudelft.jpacman.npc	<div><div></div></div>	100%		n/a
Total	1,204 of 4,694	74%	290 of 637	54%

Task 3:

The coverage results between IntelliJ and JaCoCo are vastly different. You can see that the JaCoCo tracks bytecode instruction and statement branches, while IntelliJ tracks each line. I find that JaCoCo's visualization is very helpful because it shows which statements have been covered and which have not been tested. I favor JaCoCo's coverage because it is a more specific version of the IntelliJ. I like the feature that highlights statements as well.

Task 4:

```
def test_from_dict(self):
    """ Test account dict """
    data = ACCOUNT_DATA[self.rand] # get a random account
    account = Account(**data)
    account.from_dict(data)
    result = account.to_dict()
    self.assertEqual(account.name, result["name"])

new *
def test_update(self):
    """ Test account update """
    data = ACCOUNT_DATA[self.rand] # get a random account
    account = Account(**data)
    with self.assertRaises(models.account.DataValidationError):
        account.update()
    account.create()
    account.update()
    result = account.to_dict()
    self.assertEqual(account.name, result["name"])
```

```
def test_delete(self):
    """ Test account deletion """
    data = ACCOUNT_DATA[self.rand] # get a random account
    account = Account(**data)
    account.create()
    account.delete()
    self.assertEqual(len(Account.all()), second: 0)

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

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 1.467s

Task 5:

Added test for POST and PUT.

```
def test_update_a_counter(self):
    """Test update counter. POST, PUT, GET"""
    # Create Counter doo
    self.setUp()
    result = self.client.post('/counters/doo')
    self.assertEqual(result.status_code, status.HTTP_201_CREATED)
    self.assertEqual(COUNTERS.get('doo'), second: 0)

    #Increment doo by 1
    result = self.client.put('/counters/doo')
    self.assertEqual(result.status_code, status.HTTP_200_OK)
    self.assertEqual(COUNTERS.get('doo'), second: 1)
```

```
- Test update counter. POST, PUT, GET (FAILED)

=====
FAIL: Test update counter. POST, PUT, GET
-----
Traceback (most recent call last):
  File "C:\Users\henry\Documents\GitHub\tdd\tests\test_counter.py", line 50, in test_update_a_counter
    self.assertEqual(result.status_code, status.HTTP_200_OK)
AssertionError: 405 != 200
```

Added PUT

```
new *
@app.route(rule: '/counters/<name>', methods=['PUT'])
def update_counter(name):
    """Update a counter"""
    app.logger.info(f"Request to update counter: {name}")
    global COUNTERS
    COUNTERS[name] = COUNTERS[name] + 1
    return {name: COUNTERS[name]}, status.HTTP_200_OK
```

- Test update counter. POST, PUT, GET

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 0.519s

Added GET and test for GET

```
new
@app.route(rule: '/counters/<name>', methods=['GET'])
def read_counter(name):
    """Read a counter"""
    app.logger.info(f"Request to read counter: {name}")
    global COUNTERS
    return {name: COUNTERS[name]}, status.HTTP_200_OK
```

```
#Read doo
result = self.client.get('/counters/doo')
self.assertEqual(result.status_code, status.HTTP_200_OK)
self.assertEqual(COUNTERS.get('doo'), second: 1)
```

- It should return an error for duplicates
- Test update counter. POST, PUT, GET

Name	Stmts	Miss	Cover	Missing
src\counter.py	20	0	100%	
src\status.py	6	0	100%	
TOTAL	26	0	100%	

Ran 3 tests in 0.512s