

**Git fork:** <https://github.com/YordyCruz/2DRogueLikeUnityGame/tree/main>

## **Task 1**

### **Question: Is the coverage good enough?**

- No, the coverage is not good, because the result shows low test coverage for all the class and methods.

Element ^	Class, %	Method, %	Line, %
▼  nl	3% (2/55)	1% (5/312)	1% (14/1137)
▼  tudelft	3% (2/55)	1% (5/312)	1% (14/1137)
▼  jpacman	3% (2/55)	1% (5/312)	1% (14/1137)
>  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	0% (0/13)	0% (0/78)	0% (0/345)
>  npc	0% (0/10)	0% (0/47)	0% (0/237)
>  points	0% (0/2)	0% (0/7)	0% (0/19)
>  sprite	0% (0/6)	0% (0/45)	0% (0/119)
>  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)
© PacmanConfigurationException	0% (0/1)	0% (0/2)	0% (0/4)

## Task 2:

### playerTest code:











```
1 package nl.tudelft.jpacman.level;
2
3 > import ...
12
13 /**
14  * Test class for Player.
15  */
16 new *
17 public class PlayerTest {
18     2 usages
19     private Player player;
20
21     new *
22     @BeforeEach
23     void setUp() {
24         // Initialize the necessary objects for Player
25         PacManSprites sprites = new PacManSprites();
26         PlayerFactory playerFactory = new PlayerFactory(sprites);
27         player = playerFactory.createPacMan();
28     }
29
30     new *
31     @Test
32     void testIsAliveInitially() {
33         // Player should be alive.
34         assertTrue(player.isAlive(), message: "Player should be alive.");
35     }
36 }
```

### Test Coverage:

Element ^	Class, %	Method, %	Line, %
▼ nl.tudelft.jpacman	14% (8/55)	9% (30/312)	8% (93/1151)
> 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% (13/350)
> npc	0% (0/10)	0% (0/47)	0% (0/237)
> points	0% (0/2)	0% (0/7)	0% (0/19)
> 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)
● PacmanConfigurationException	0% (0/1)	0% (0/2)	0% (0/4)

## Task 2.1:

**Methods:** implement test for getScore, addPoints, and setKiller

Element ^	Class, %	Method, %	Line, %
▼  nl.tudelft.jpacman	14% (8/55)	11% (36/312)	9% (108/1151)
>  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)	12% (10/78)	6% (24/350)
Ⓢ CollisionInteractionMap	0% (0/2)	0% (0/9)	0% (0/41)
Ⓢ CollisionMap	100% (0/0)	100% (0/0)	100% (0/0)
Ⓢ DefaultPlayerInteractionMap	0% (0/1)	0% (0/5)	0% (0/13)
Ⓢ Level	0% (0/2)	0% (0/17)	0% (0/113)
Ⓢ LevelFactory	0% (0/2)	0% (0/7)	0% (0/27)
Ⓢ LevelTest	0% (0/1)	0% (0/9)	0% (0/30)
Ⓢ MapParser	0% (0/1)	0% (0/10)	0% (0/71)
Ⓢ Pellet	0% (0/1)	0% (0/3)	0% (0/5)
Ⓢ Player	100% (1/1)	87% (7/8)	79% (19/24)
Ⓢ PlayerCollisions	0% (0/1)	0% (0/7)	0% (0/21)
Ⓢ PlayerFactory	100% (1/1)	100% (3/3)	100% (5/5)
>  npc	0% (0/10)	0% (0/47)	0% (0/237)
>  points	0% (0/2)	0% (0/7)	0% (0/19)
>  sprite	66% (4/6)	46% (21/45)	54% (70/128)
>  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)
Ⓢ PacmanConfigurationException	0% (0/1)	0% (0/2)	0% (0/4)

## Task 3:

### jpacman

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	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		77%		47%	54	86	21	144	7	31	0	6
default		0%		0%	12	12	21	21	5	5	1	1
nl.tudelft.jpacman.board		86%		58%	44	93	2	110	0	40	0	7
nl.tudelft.jpacman.sprite		86%		59%	30	70	11	113	5	38	0	5
nl.tudelft.jpacman		69%		25%	12	30	18	52	6	24	1	2
nl.tudelft.jpacman.points		60%		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		100%		n/a	0	4	0	8	0	4	0	1
Total	1,210 of 4,694	74%	293 of 637	54%	292	590	228	1,039	50	268	6	47

**Questions:** Are the coverage results from JaCoCo like the ones you got from IntelliJ in the last task? Why so or why not?

IntelliJ and JaCoCo vary for each other, JaCoCo shows what percent and how many are missing and is easier to read to understand what needs to be tested. While IntelliJ is easier to run due to its integrated with the coding environment it provides enough information to understand if the coverage for a specific test. Overall I prefer using JaCoCo interface since it more detail about finding branches and better at reporting missed branches

## Task 4:

### Original

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 0.689s				
OK				

## Def test\_repr(self)

```
- Test Account creation using known data
- Test the representation of an account

Name                Stmts  Miss  Cover   Missing
-----
models\__init__.py    7      0   100%
models\account.py    40     12    70%   30, 34-35, 45-48, 52-54, 74-75
-----
TOTAL                 47     12    74%
-----

Ran 3 tests in 0.555s

OK
```

## Def test\_to\_dict

```
- 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 0.537s

OK

PS C:\Users\Yordy\IdeaProjects\test_coverage>
```

## Def test\_from\_dict(self)

```
Test Account Model
- Test creating multiple Accounts
- Test Account creation using known data
- from dict
- 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      9    78%   45-48, 52-54, 74-75
-----
TOTAL                 47      9    81%
-----

Ran 5 tests in 0.549s

OK
```

**All test coverage to 100% includes** - update\_success, update\_account, delete\_account, find\_account,

```
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 account attributes from the dictionary
- Test the representation of an account
- Test account to dict
- Test updating an account without an ID raises `ValidationError`
- Test updating the account successfully

Name	Stmts	Miss	Cover	Missing
models\__init__.py	7	0	100%	
models\account.py	40	0	100%	
TOTAL	47	0	100%	

```
Ran 9 tests in 0.590s
```

```
OK
```

```
PS C:\Users\Yordy\IdeaProjects\test_coverage>
```

## Task 5: - TDD

### Test\_counter.py

```
def test_update_a_counter(self):  
    """It should update a counter"""  
    # Create a counter named 'update'  
    create_result = self.client.post('/counters/update')  
    self.assertEqual(create_result.status_code, status.HTTP_201_CREATED)  
  
    # Set the value of counter 'update'  
    get_result = self.client.get('/counters/update')  
    initial_value = get_result.json['update']  
  
    # Update the counter 'update'  
    update_result = self.client.put('/counters/update')  
    self.assertEqual(update_result.status_code, status.HTTP_200_OK)  
  
    # Retrieve the updated value of the counter 'update'  
    get_updated_result = self.client.get('/counters/update')  
    updated_value = get_updated_result.json['update']  
  
    # Check the counter value is +1  
    self.assertEqual(updated_value, initial_value + 1)
```

This test method checks whether the counter name can be created

```
def test_read_counter(self):  
    """It should read a counter"""  
    result = self.client.post('/counters/example')  
    self.assertEqual(result.status_code, status.HTTP_201_CREATED)  
    result = self.client.get('/counters/example')  
    self.assertEqual(result.status_code, status.HTTP_200_OK)  
    self.assertEqual(result.json['example'], 0)
```

Test method that verifies the creation and reading 'example' work after creation.



Counters.py:

update\_counter(name)/read\_counter – green phase

```
@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

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

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

**Create a counter** – given a name and initialize its value to 0

**Return errors for duplicates** – if Counter already exist returns a conflict error

**Update the counter** – Increments the value by 1 if exists, and if it does not exist do not perform.

**Read the counter** – Retrieves the current value with the name, if it does not exist do not perform the action.



## All task completed:

Nosetest passes with 100%

```
Create Counter tests
```

- It should create a counter
- It should return an error for duplicates
- It should read a counter
- It should update a counter

Name	Stmts	Miss	Cover	Missing
src\counters.py	22	0	100%	
src\status.py	6	0	100%	
TOTAL	28	0	100%	

```
Ran 4 tests in 0.228s
```

```
OK
```

```
PS C:\Users\Yordy\IdeaProjects\tdd>
```