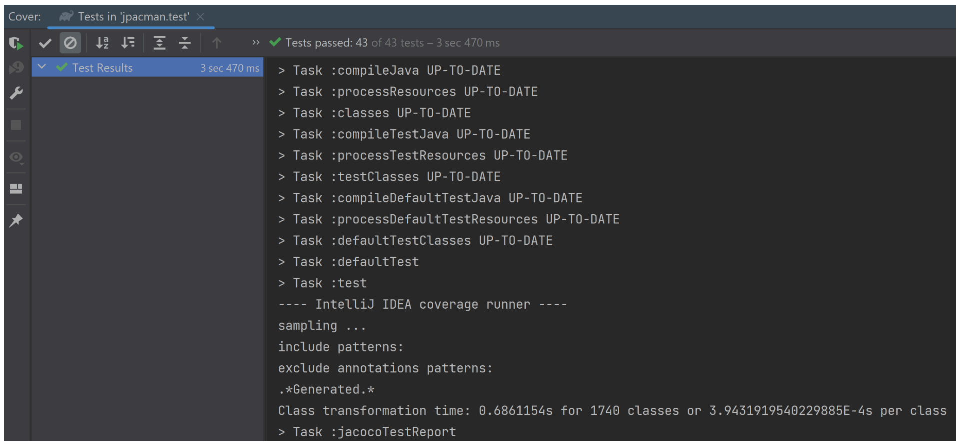
Brandon Cunningham
CS 472-1001
February 5th, 2024

Report - Testing Lab

Fork repository: https://github.com/barkangel/UNLV-S24-CS472-Group7

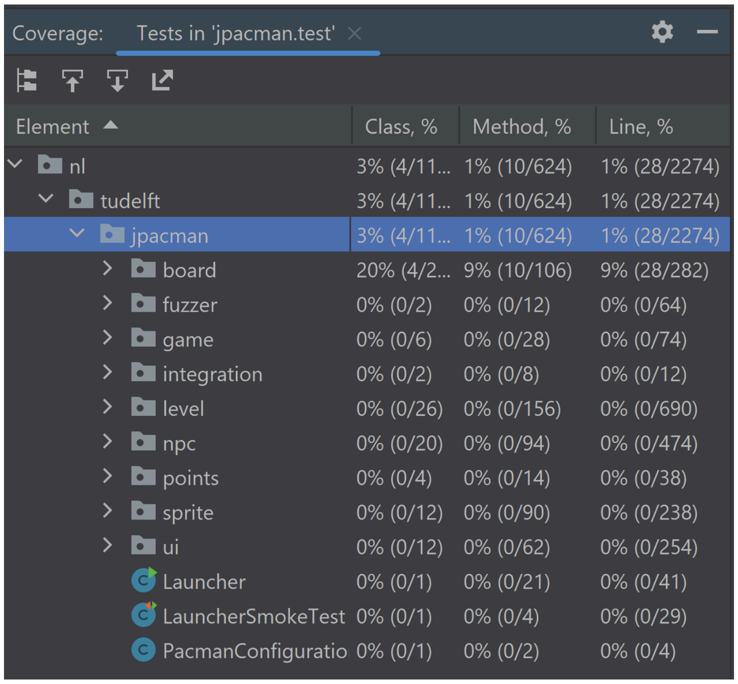
Task 1: JPacMan Test Coverage

Running ‘Tests’ in jpacman.test with Coverage”


Cover:
Tests in 'jpacman.test'
Tests passed: 43 of 43 tests – 3 sec 470 ms
Test Results 3 sec 470 ms

```
> Task :compileJava UP-TO-DATE
> Task :processResources UP-TO-DATE
> Task :classes UP-TO-DATE
> Task :compileTestJava UP-TO-DATE
> Task :processTestResources UP-TO-DATE
> Task :testClasses UP-TO-DATE
> Task :compileDefaultTestJava UP-TO-DATE
> Task :processDefaultTestResources UP-TO-DATE
> Task :defaultTestClasses UP-TO-DATE
> Task :defaultTest
> Task :test
---- IntelliJ IDEA coverage runner ----
sampling ...
include patterns:
exclude annotations patterns:
.*Generated.*
Class transformation time: 0.6861154s for 1740 classes or 3.9431919540229885E-4s per class
> Task :jacocoTestReport
```

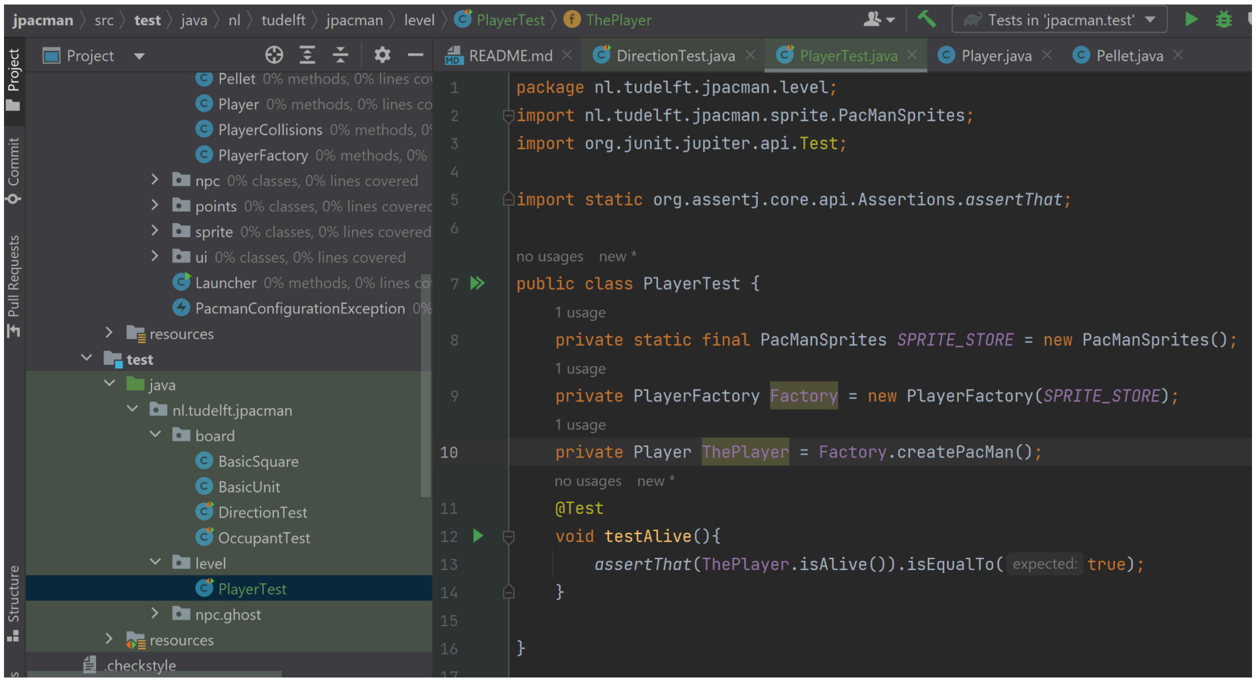
Coverage results:



Element	Class, %	Method, %	Line, %
nl	3% (4/11...	1% (10/624)	1% (28/2274)
tudelft	3% (4/11...	1% (10/624)	1% (28/2274)
jpacman	3% (4/11...	1% (10/624)	1% (28/2274)
board	20% (4/2...	9% (10/106)	9% (28/282)
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	0% (0/26)	0% (0/156)	0% (0/690)
npc	0% (0/20)	0% (0/94)	0% (0/474)
points	0% (0/4)	0% (0/14)	0% (0/38)
sprite	0% (0/12)	0% (0/90)	0% (0/238)
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)
PacmanConfiguratio	0% (0/1)	0% (0/2)	0% (0/4)

Task 2: Increasing Coverage on JPacMan

Added: level package, a test folder, PlayerTest class, and isAlive() test case



jpacman 〉 src 〉 test 〉 java 〉 nl 〉 tudelft 〉 jpacman 〉 level 〉 PlayerTest 〉 ThePlayer

Tests in 'jpacman.test'

Project

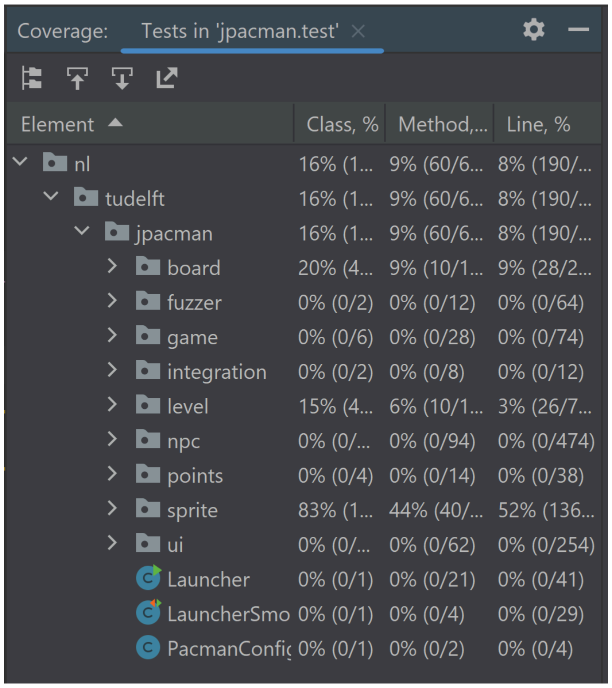
```
Pellet 0% methods, 0% lines co
Player 0% methods, 0% lines co
PlayerCollisions 0% methods, 0
PlayerFactory 0% methods, 0%
npc 0% classes, 0% lines covered
points 0% classes, 0% lines covered
sprite 0% classes, 0% lines covered
ui 0% classes, 0% lines covered
Launcher 0% methods, 0% lines co
PacmanConfigurationException 0%
resources
test
  java
    nl.tudelft.jpacman
      board
        BasicSquare
        BasicUnit
        DirectionTest
        OccupantTest
      level
        PlayerTest
      npc.ghost
  resources
.checkstyle
```

README.md | DirectionTest.java | PlayerTest.java | Player.java | Pellet.java

```java
package nl.tudelft.jpacman.level;
import nl.tudelft.jpacman.sprite.PacManSprites;
import org.junit.jupiter.api.Test;

import static org.assertj.core.api.Assertions.assertThat;

no usages   new *
public class PlayerTest {
    1 usage
    private static final PacManSprites SPRITE_STORE = new PacManSprites();
    1 usage
    private PlayerFactory Factory = new PlayerFactory(SPRITE_STORE);
    1 usage
    private Player ThePlayer = Factory.createPacMan();
    no usages   new *
    @Test
    void testAlive(){
        assertThat(ThePlayer.isAlive()).isEqualTo( expected: true);
    }

}
```

Building jpacman.test and running with coverage:

Noticeable differences: Sprite has 52% line coverage, compared to 0% before.



Coverage: Tests in 'jpacman.test'

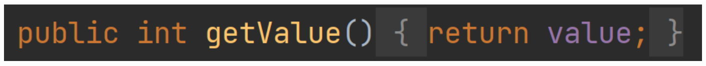
Element	Class, %	Method,...	Line, %
nl	16% (1...	9% (60/6...	8% (190/...
tudelft	16% (1...	9% (60/6...	8% (190/...
jpacman	16% (1...	9% (60/6...	8% (190/...
board	20% (4...	9% (10/1...	9% (28/2...
fuzzer	0% (0/2)	0% (0/12)	0% (0/64)
game	0% (0/6)	0% (0/28)	0% (0/74)
integration	0% (0/2)	0% (0/8)	0% (0/12)
level	15% (4...	6% (10/1...	3% (26/7...
npc	0% (0/...	0% (0/94)	0% (0/474)
points	0% (0/4)	0% (0/14)	0% (0/38)
sprite	83% (1...	44% (40/...	52% (136...
ui	0% (0/...	0% (0/62)	0% (0/254)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)
LauncherSmo	0% (0/1)	0% (0/4)	0% (0/29)
PacmanConfi	0% (0/1)	0% (0/2)	0% (0/4)

Task 2.1:

Added 4 test cases for the following:

1. **getValue()** in src/main/java/nl/tudelft/jpacman/level/Pellet.getValue

In Pellet.java:

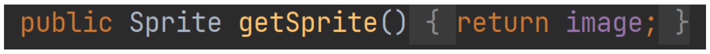


```
public int getValue() { return value; }
```

Implemented test case in PelletTest.java along with getSprite.

2. **getSprite()** in src/main/java/nl/tudelft/jpacman/level/Pellet.getSprite

In Pellet.java:



```
public Sprite getSprite() { return image; }
```

Implemented test case in PelletTest.java along with getValue in the level package in the test folder.

Code:

```java
package nl.tudelft.jpacman.level;
import nl.tudelft.jpacman.sprite.PacManSprites;
import nl.tudelft.jpacman.sprite.Sprite;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import static org.assertj.core.api.Assertions.assertThat;

public class PelletTest {
   private static final PacManSprites SPRITE_STORE = new PacManSprites();
   private Pellet pellet;

   @BeforeEach
   void setUp() {
       pellet = new Pellet(10, SPRITE_STORE.getPelletSprite());
   }

   @Test
   void shouldReturnValueOfPellet() {
       assertThat(pellet.getValue()).isEqualTo(10);
   }

   @Test
   void shouldReturnSpriteOfPellet() {
       Sprite expectedSprite = SPRITE_STORE.getPelletSprite();
       assertThat(pellet.getSprite()).isEqualTo(expectedSprite);
   }
}
```

3. **createClyde()** in src/main/java/nl/tudelft/jpacman/npc/ghost/GhostFactory.createClyde

In GhostFactory.java:

```
public Ghost createInky() { return new Inky(sprites.getGhostSprite(GhostColor.CYAN));
```

Implemented test case in GhostFactoryTest.java.

4. **createInky()** in src/main/java/nl/tudelft/jpacman/npc/ghost/GhostFactory.createInky

In GhostFactory.java:

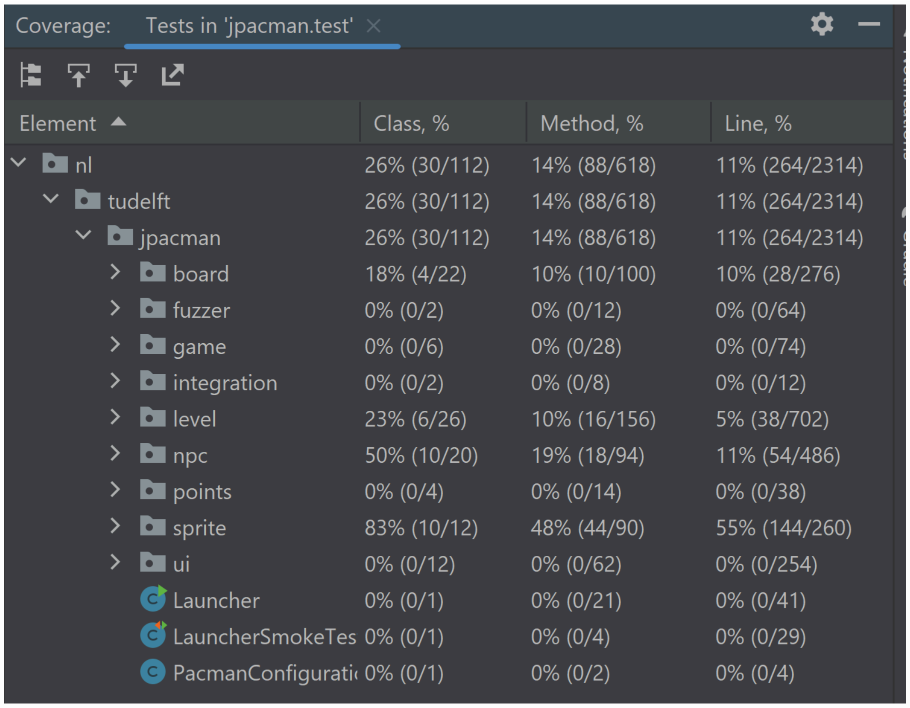
```
public Ghost createClyde() { return new Clyde(sprites.getGhostSprite(GhostColor.ORANGE))
```

Implemented test case in GhostFactoryTest.java in the npc.ghost package within the test folder.

Code:

```
package nl.tudelft.jpacman.npc.ghost;
import nl.tudelft.jpacman.board.BoardFactory;
import nl.tudelft.jpacman.npc.Ghost;
import nl.tudelft.jpacman.npc.ghost.GhostColor;
import nl.tudelft.jpacman.npc.ghost.GhostFactory;
import nl.tudelft.jpacman.sprite.PacManSprites;
import nl.tudelft.jpacman.board.Square;
import nl.tudelft.jpacman.level.LevelFactory;
import nl.tudelft.jpacman.level.MapParser;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import java.util.List;
import static org.assertj.core.api.Assertions.assertThat;

public class GhostFactoryTest {
   private GhostFactory ghostFactory;
   @BeforeEach
   void setUp() {
       ghostFactory = new GhostFactory(new PacManSprites()); }
   @Test
   void shouldCreateInky() {
       Ghost inky = ghostFactory.createInky();
       assertThat(inky).isNotNull();
       assertThat(isGhostValid(inky)).isTrue(); }
   @Test
   void shouldCreateClyde() {
       Ghost clyde = ghostFactory.createClyde();
       assertThat(clyde).isNotNull();
       assertThat(isGhostValid(clyde)).isTrue(); }
   private boolean isGhostValid(Ghost ghost) {
       // Implement this based on your actual Ghost class or interface
       return true;
   }
}
```

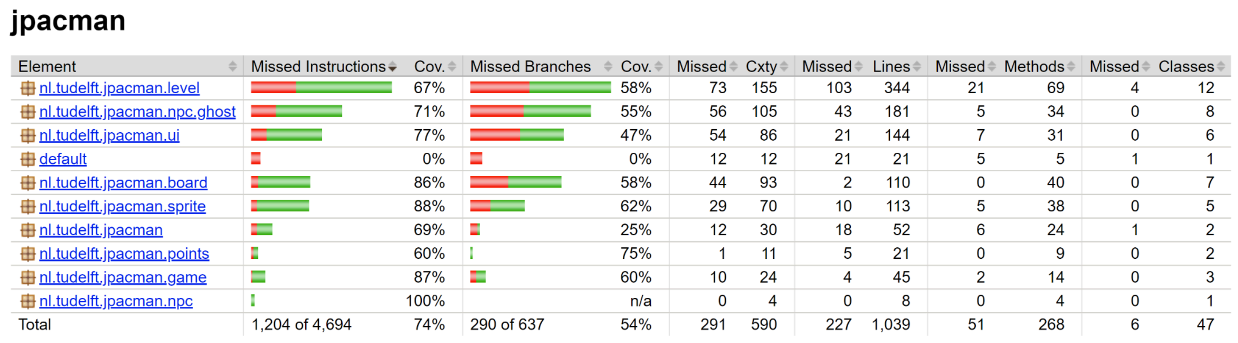
The coverage after these 4 test cases were introduced is below. Noticeable changes when comparing this coverage to Task 2 include: 11% total line coverage, compared to 8% in Task 2. 5% level line coverage compared to 3%, and 55% sprite coverage compared to 52% before.


Coverage: Tests in 'jpacman.test'

Element	Class, %	Method, %	Line, %
nl	26% (30/112)	14% (88/618)	11% (264/2314)
tudelft	26% (30/112)	14% (88/618)	11% (264/2314)
jpacman	26% (30/112)	14% (88/618)	11% (264/2314)
board	18% (4/22)	10% (10/100)	10% (28/276)
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	23% (6/26)	10% (16/156)	5% (38/702)
npc	50% (10/20)	19% (18/94)	11% (54/486)
points	0% (0/4)	0% (0/14)	0% (0/38)
sprite	83% (10/12)	48% (44/90)	55% (144/260)
ui	0% (0/12)	0% (0/62)	0% (0/254)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)
LauncherSmokeTes	0% (0/1)	0% (0/4)	0% (0/29)
PacmanConfigurati	0% (0/1)	0% (0/2)	0% (0/4)

Task 3: JaCoCo Report on JPacman

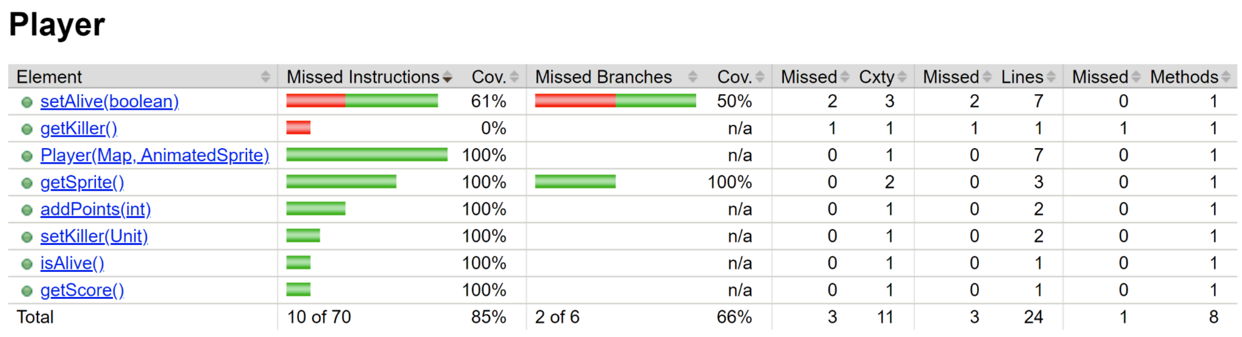
JPacMan total coverage by JaCoCo:



jpacman

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
nl.tudelft.jpacman.level		67%		58%	73	155	103	344	21	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		88%		62%	29	70	10	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	1,204 of 4,694	74%	290 of 637	54%	291	590	227	1,039	51	268	6	47

JPacMan > nl.tudelft.jpacman.level > Player total coverage by JaCoCo:



Player

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
setAlive(boolean)		61%		50%	2	3	2	7	0	1
getKiller()		0%		n/a	1	1	1	1	1	1
Player(Map, AnimatedSprite)		100%		n/a	0	1	0	7	0	1
getSprite()		100%		100%	0	2	0	3	0	1
addPoints(int)		100%		n/a	0	1	0	2	0	1
setKiller(Unit)		100%		n/a	0	1	0	2	0	1
isAlive()		100%		n/a	0	1	0	1	0	1
getScore()		100%		n/a	0	1	0	1	0	1
Total	10 of 70	85%	2 of 6	66%	3	11	3	24	1	8

1. **Are the coverage results from JaCoCo similar to the ones you got from IntelliJ in the last task? Why so or why not?**

No, the coverage results are not similar. IntelliJ coverage shows only ~26% coverage, where JaCoCo shows around ~67% coverage. A difference in configuration because one method of coverage may collect more or less data. Maybe one coverage method filters our certain files and ignores ghost files, while the other includes them, and that could go for any other type of file or redundancy.

2. **Did you find helpful the source code visualization from JaCoCo on uncovered branches?**

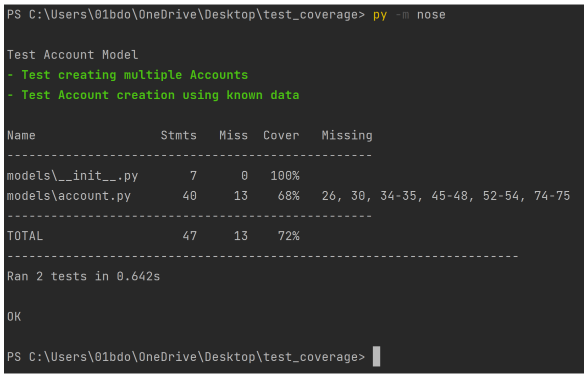
Yes, I found the source code visualization helpful on missed branches. That feature adds another level of depth to file coverage and testing that is very useful to determine the quality of the program.

3. **Which visualization did you prefer and why? IntelliJ's coverage window or JaCoCo's report?**

I preferred the visualization of the IntelliJ coverage window because the data was right within the IDE, and I directly saw the effects of my test file inclusion by seeing the % 's go up after every run test.

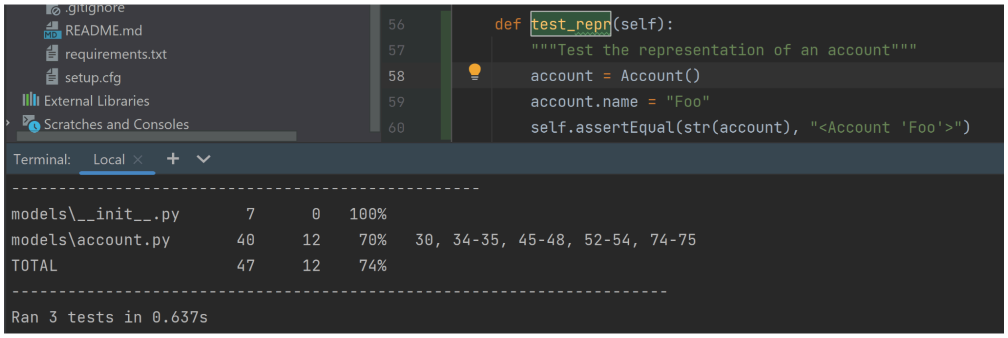
Task 4: Working with Python Test Coverage

Note: nosetests did not work for me, so I used pynose and pytest



```
PS C:\Users\01bdo\OneDrive\Desktop\test_coverage> py -m nose

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 0.642s

OK

PS C:\Users\01bdo\OneDrive\Desktop\test_coverage>
```

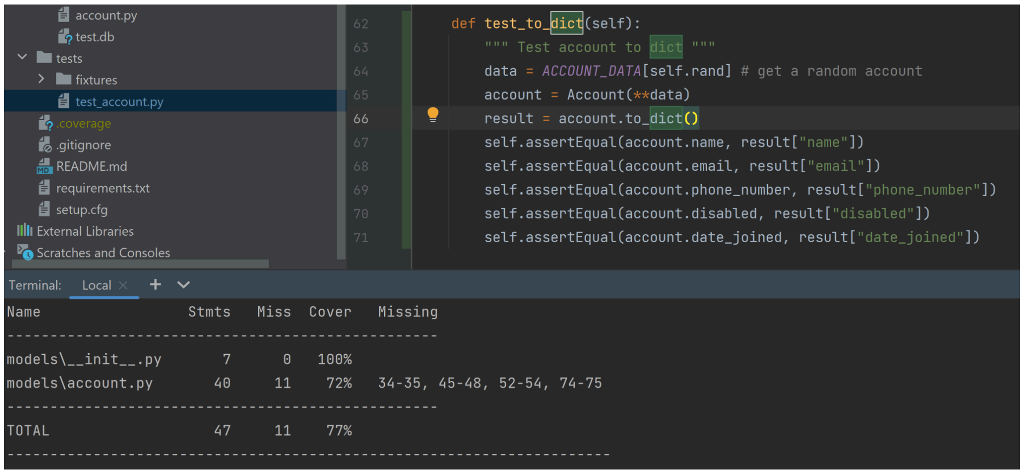
After adding test_repr: 74%



```python
    def test_repr(self):
        """Test the representation of an account"""
        account = Account()
        account.name = "Foo"
        self.assertEqual(str(account), "<Account 'Foo'>")
```

```
--------------------------------------------------
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 0.637s
```

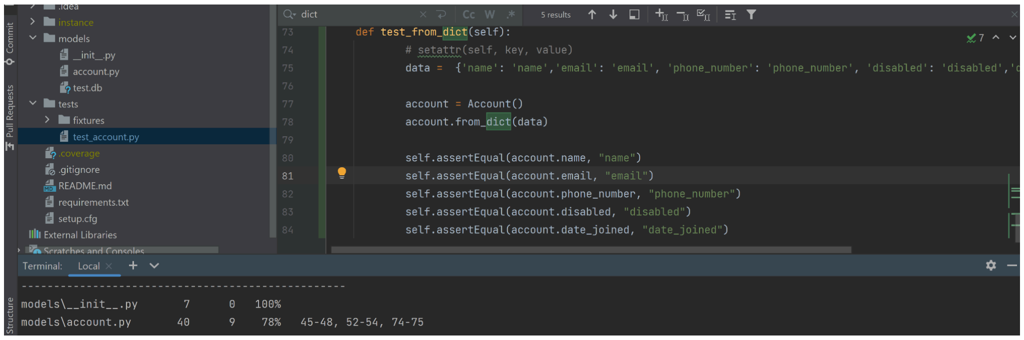
After adding test_to_dict: 77%



```python
    def test_to_dict(self):
        """ Test account to dict """
        data = ACCOUNT_DATA[self.rand] # get a random account
        account = Account(**data)
        result = account.to_dict()
        self.assertEqual(account.name, result["name"])
        self.assertEqual(account.email, result["email"])
        self.assertEqual(account.phone_number, result["phone_number"])
        self.assertEqual(account.disabled, result["disabled"])
        self.assertEqual(account.date_joined, result["date_joined"])
```

```
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%
----------------------------------------------------------------------
```

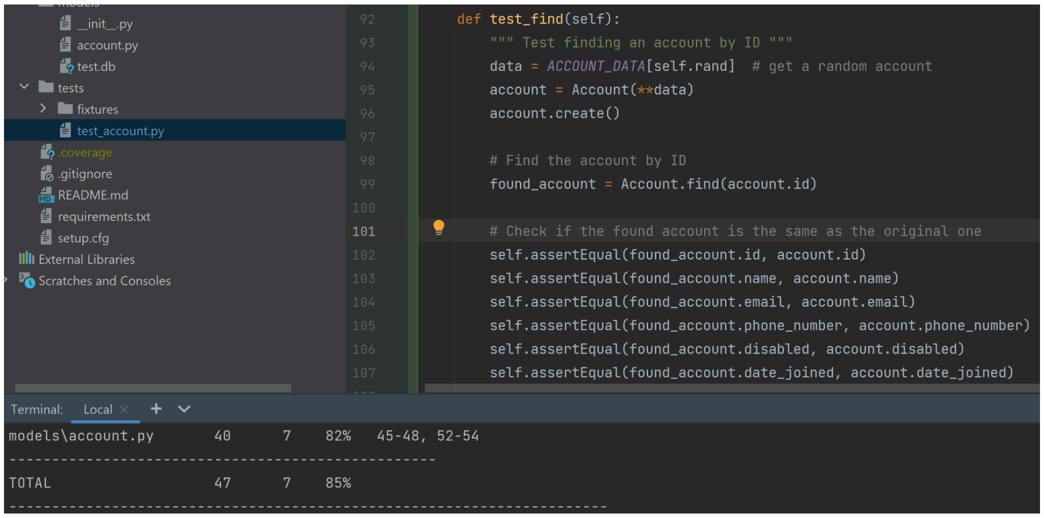
After adding test_from_dict: 78%



```python
    def test_from_dict(self):
        # setattr(self, key, value)
        data =  {'name': 'name','email': 'email', 'phone_number': 'phone_number', 'disabled': 'disabled','

        account = Account()
        account.from_dict(data)

        self.assertEqual(account.name, "name")
        self.assertEqual(account.email, "email")
        self.assertEqual(account.phone_number, "phone_number")
        self.assertEqual(account.disabled, "disabled")
        self.assertEqual(account.date_joined, "date_joined")
```

```
--------------------------------------------------
models\__init__.py       7      0   100%
models\account.py       40      9    78%   45-48, 52-54, 74-75
```

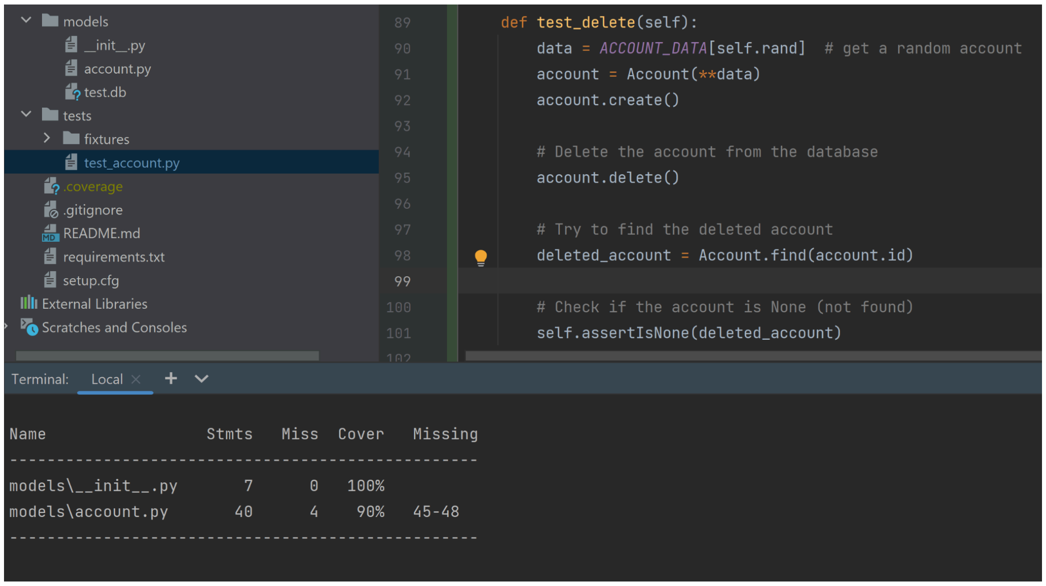
After adding test_find: 85%



```python
    def test_find(self):
        """ Test finding an account by ID """
        data = ACCOUNT_DATA[self.rand]  # get a random account
        account = Account(**data)
        account.create()

        # Find the account by ID
        found_account = Account.find(account.id)

        # Check if the found account is the same as the original one
        self.assertEqual(found_account.id, account.id)
        self.assertEqual(found_account.name, account.name)
        self.assertEqual(found_account.email, account.email)
        self.assertEqual(found_account.phone_number, account.phone_number)
        self.assertEqual(found_account.disabled, account.disabled)
        self.assertEqual(found_account.date_joined, account.date_joined)
```

```
models\account.py       40      7    82%   45-48, 52-54
---------------------------------------------------
TOTAL                   47      7    85%
-----------------------------------------------------------------------
```

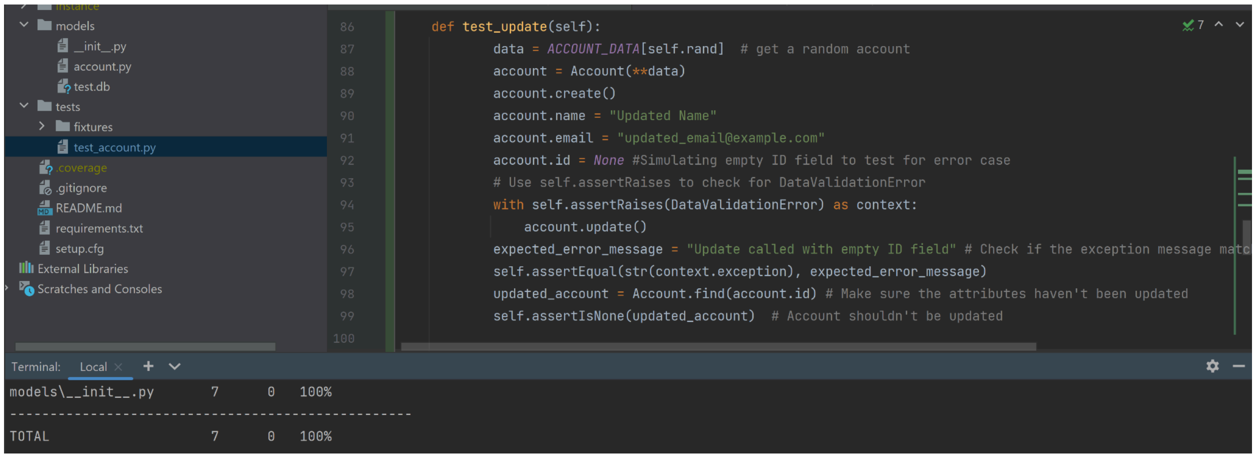
After adding test_delete: 90%



```python
    def test_delete(self):
        data = ACCOUNT_DATA[self.rand]  # get a random account
        account = Account(**data)
        account.create()

        # Delete the account from the database
        account.delete()

        # Try to find the deleted account
        deleted_account = Account.find(account.id)

        # Check if the account is None (not found)
        self.assertIsNone(deleted_account)
```

```
Name                  Stmts   Miss  Cover   Missing
---------------------------------------------------
models\__init__.py        7      0   100%
models\account.py        40      4    90%   45-48
---------------------------------------------------
```

After adding test_update: 100%

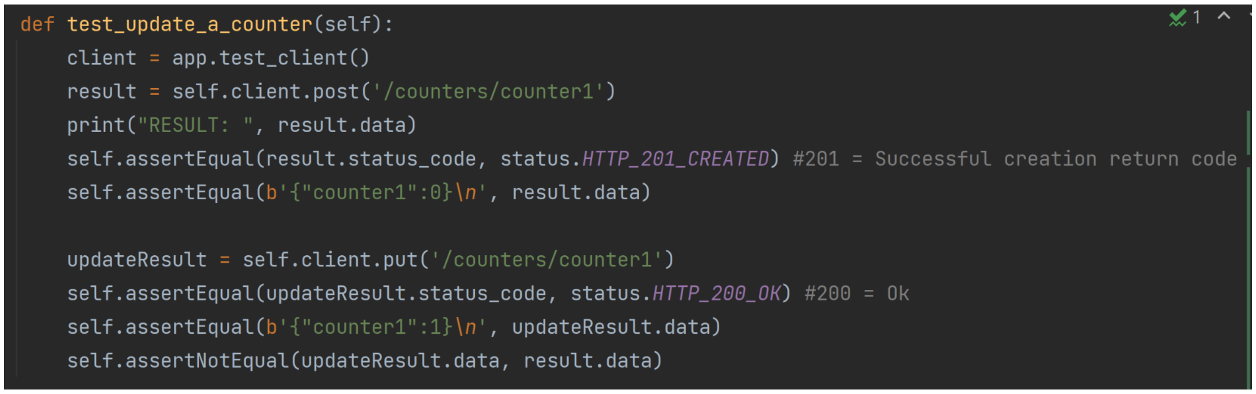


```python
    def test_update(self):
            data = ACCOUNT_DATA[self.rand]  # get a random account
            account = Account(**data)
            account.create()
            account.name = "Updated Name"
            account.email = "updated_email@example.com"
            account.id = None #Simulating empty ID field to test for error case
            # Use self.assertRaises to check for DataValidationError
            with self.assertRaises(DataValidationError) as context:
                account.update()
            expected_error_message = "Update called with empty ID field" # Check if the exception message matc
            self.assertEqual(str(context.exception), expected_error_message)
            updated_account = Account.find(account.id) # Make sure the attributes haven't been updated
            self.assertIsNone(updated_account)  # Account shouldn't be updated
```

```
models\__init__.py       7      0   100%
---------------------------------------------------
TOTAL                    7      0   100%
```

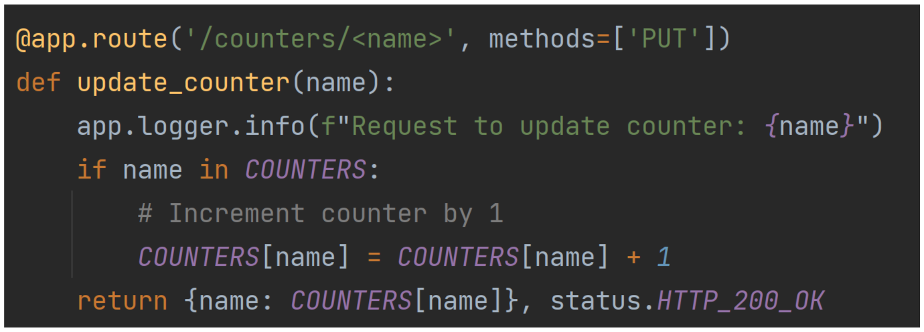
Task 5: TDD

Creating test_update_a_counter(self) in test_counter.py:



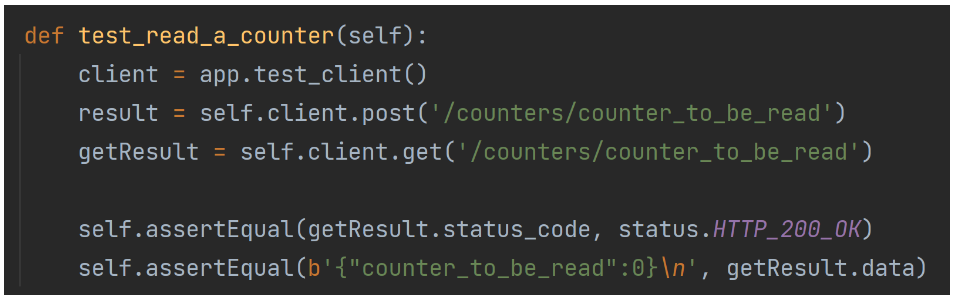
```python
def test_update_a_counter(self):
    client = app.test_client()
    result = self.client.post('/counters/counter1')
    print("RESULT: ", result.data)
    self.assertEqual(result.status_code, status.HTTP_201_CREATED) #201 = Successful creation return code
    self.assertEqual(b'{"counter1":0}\n', result.data)

    updateResult = self.client.put('/counters/counter1')
    self.assertEqual(updateResult.status_code, status.HTTP_200_OK) #200 = Ok
    self.assertEqual(b'{"counter1":1}\n', updateResult.data)
    self.assertNotEqual(updateResult.data, result.data)
```

We are now in **RED** stage, because our new test case fails.
Creating update_counter(name) in counter.py:



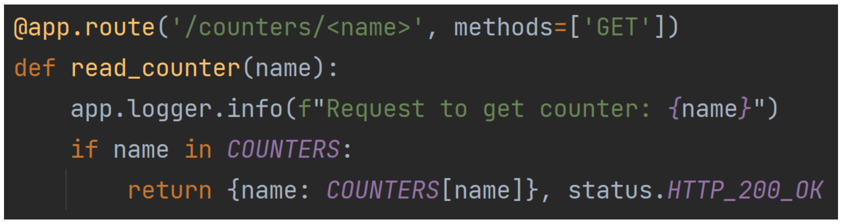
```python
@app.route('/counters/<name>', methods=['PUT'])
def update_counter(name):
    app.logger.info(f"Request to update counter: {name}")
    if name in COUNTERS:
        # Increment counter by 1
        COUNTERS[name] = COUNTERS[name] + 1
    return {name: COUNTERS[name]}, status.HTTP_200_OK
```

We are now in **GREEN** stage, because we have written minimum amount of code to pass test.
Creating a test case to read a counter:



```python
def test_read_a_counter(self):
    client = app.test_client()
    result = self.client.post('/counters/counter_to_be_read')
    getResult = self.client.get('/counters/counter_to_be_read')

    self.assertEqual(getResult.status_code, status.HTTP_200_OK)
    self.assertEqual(b'{"counter_to_be_read":0}\n', getResult.data)
```

Implementing read counter actual case:



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

Exceptions I encountered while doing Task 5: TDD:

AssertionError: 404 !=201 - Occurred when first writing counter, and /counters endpoint wasn't found
AssertionError: 201 != 409 - Occurred during refactor, when a second counter with same name was created.
HTTP_409_CONFLICT - Occurred during refactor, when ran twice after attaining the exception AssertionError 201 != 409.
ModuleNotFoundError - When I ran nose after writing a test case for counter in test_counter.py, this is because there was no module to refer the test to in counter.py
ImportError - When I was writing the module for counter.py that's referenced by test_counter.py, and it happened because we didn't import the flask application. It was fixed by importing flask at the top of the counter.py file.