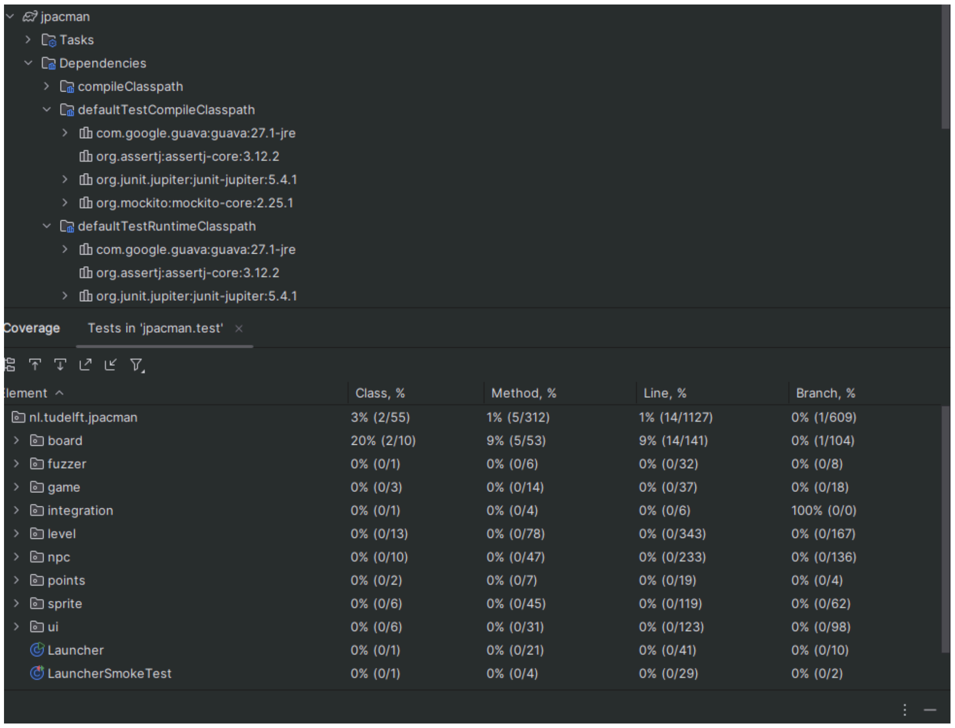
Task 1

Is the coverage good enough?
No!

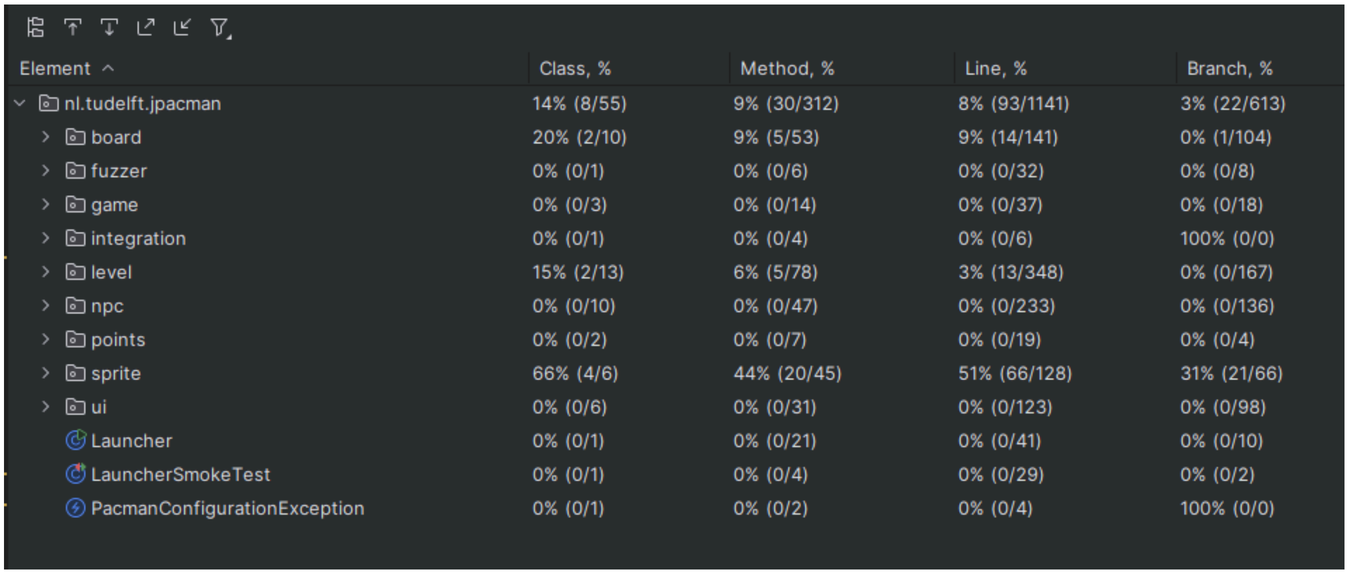


```
⌄ jpacman
  › Tasks
  ⌄ Dependencies
    › compileClasspath
    ⌄ defaultTestCompileClasspath
      › com.google.guava:guava:27.1-jre
        org.assertj:assertj-core:3.12.2
      › org.junit.jupiter:junit-jupiter:5.4.1
      › org.mockito:mockito-core:2.25.1
    ⌄ defaultTestRuntimeClasspath
      › com.google.guava:guava:27.1-jre
        org.assertj:assertj-core:3.12.2
      › org.junit.jupiter:junit-jupiter:5.4.1
```

Coverage   Tests in 'jpacman.test' ×

lement ^	Class, %	Method, %	Line, %	Branch, %
nl.tudelft.jpacman	3% (2/55)	1% (5/312)	1% (14/1127)	0% (1/609)
› board	20% (2/10)	9% (5/53)	9% (14/141)	0% (1/104)
› fuzzer	0% (0/1)	0% (0/6)	0% (0/32)	0% (0/8)
› game	0% (0/3)	0% (0/14)	0% (0/37)	0% (0/18)
› integration	0% (0/1)	0% (0/4)	0% (0/6)	100% (0/0)
› level	0% (0/13)	0% (0/78)	0% (0/343)	0% (0/167)
› npc	0% (0/10)	0% (0/47)	0% (0/233)	0% (0/136)
› points	0% (0/2)	0% (0/7)	0% (0/19)	0% (0/4)
› sprite	0% (0/6)	0% (0/45)	0% (0/119)	0% (0/62)
› ui	0% (0/6)	0% (0/31)	0% (0/123)	0% (0/98)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)	0% (0/10)
LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)	0% (0/2)

Task 2



Element ^	Class, %	Method, %	Line, %	Branch, %
⌄ nl.tudelft.jpacman	14% (8/55)	9% (30/312)	8% (93/1141)	3% (22/613)
› board	20% (2/10)	9% (5/53)	9% (14/141)	0% (1/104)
› fuzzer	0% (0/1)	0% (0/6)	0% (0/32)	0% (0/8)
› game	0% (0/3)	0% (0/14)	0% (0/37)	0% (0/18)
› integration	0% (0/1)	0% (0/4)	0% (0/6)	100% (0/0)
› level	15% (2/13)	6% (5/78)	3% (13/348)	0% (0/167)
› npc	0% (0/10)	0% (0/47)	0% (0/233)	0% (0/136)
› points	0% (0/2)	0% (0/7)	0% (0/19)	0% (0/4)
› sprite	66% (4/6)	44% (20/45)	51% (66/128)	31% (21/66)
› ui	0% (0/6)	0% (0/31)	0% (0/123)	0% (0/98)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)	0% (0/10)
LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)	0% (0/2)
PacmanConfigurationException	0% (0/1)	0% (0/2)	0% (0/4)	100% (0/0)

Task 2.1

Code snippets:

```java
/**
* The level under test.
*/
private Level level;

/**
* An NPC on this level.
*/
private final Ghost ghost = mock(Ghost.class);

/**
* Starting position 1.
*/
private final Square square1 = mock(Square.class);

/**
* Starting position 2.
*/
private final Square square2 = mock(Square.class);

/**
* The board for this level.
*/
private final Board board = mock(Board.class);

private SinglePlayerGame game;

/**
* The collision map.
*/
private final CollisionMap collisions = mock(CollisionMap.class);

/**
* Sets up the level with the default board, a single NPC and a starting
* square.
*/
@BeforeEach
void setUp() {
   final long defaultInterval = 100L;
   level = new Level(board, Lists.newArrayList(ghost), Lists.newArrayList(
       square1, square2), collisions);
   when(ghost.getInterval()).thenReturn(defaultInterval);
}
```

```
@Test
void testAddObserver(){
    level.start();

    level.addObserver(game);
    // note added a getter in level class to test this method
   // size should be 1
   assert(level.getObservers().size() == 1);
}
```

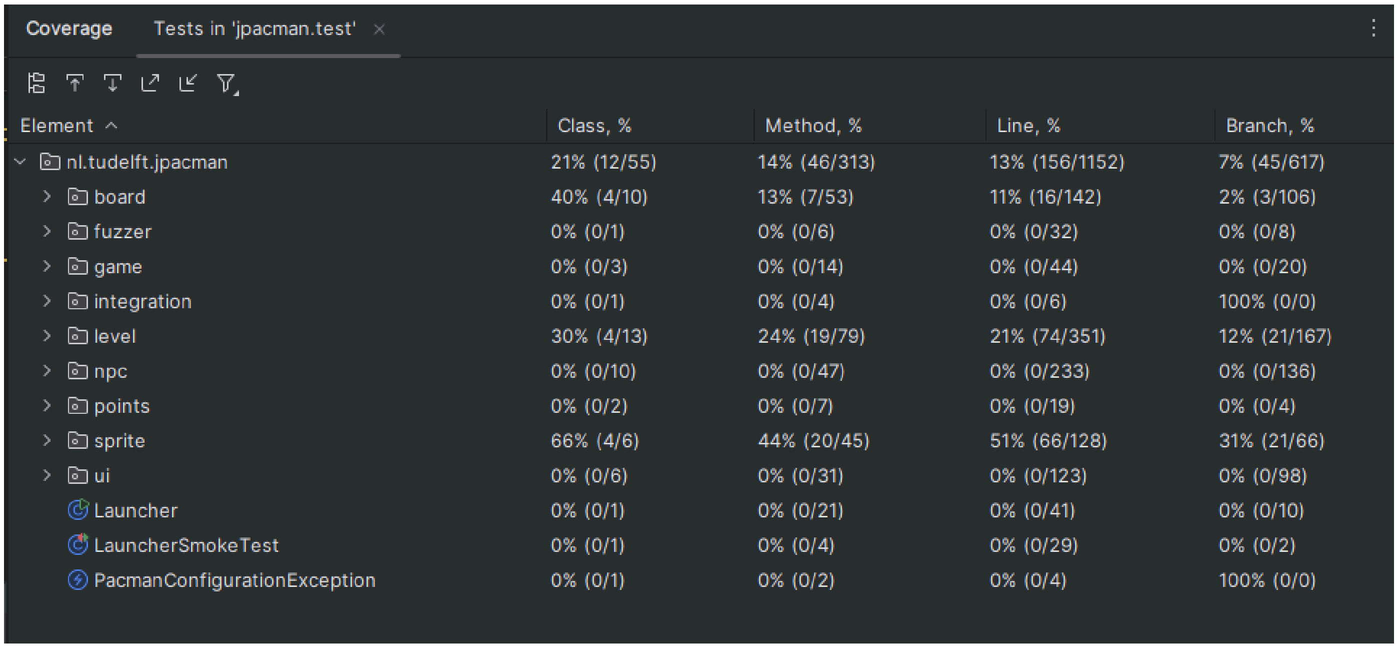
```
@Test
void testRemoveObserver(){
       level.start();

    level.addObserver(game);
    level.removeObserver(game);
    // note added a getter in level class to test this method
    // observers should be empty
   assert((level.getObservers().isEmpty()));
}
```

```
@Test
void testRemainingPellets(){
    level.start();
    // number of remaining pellets should be 0 for this level
   assert((level.remainingPellets() == 0));
}
```

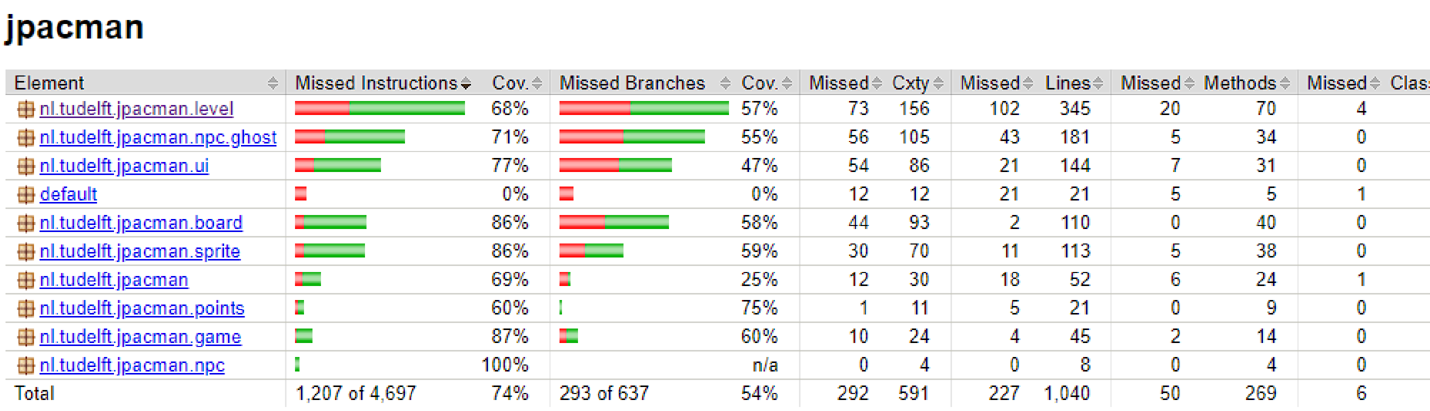
Task 3:

Questions:

Are the coverage results from JaCoCo similar to the ones you got from IntelliJ in the last task?



Element	Class, %	Method, %	Line, %	Branch, %
nl.tudelft.jpacman	21% (12/55)	14% (46/313)	13% (156/1152)	7% (45/617)
board	40% (4/10)	13% (7/53)	11% (16/142)	2% (3/106)
fuzzer	0% (0/1)	0% (0/6)	0% (0/32)	0% (0/8)
game	0% (0/3)	0% (0/14)	0% (0/44)	0% (0/20)
integration	0% (0/1)	0% (0/4)	0% (0/6)	100% (0/0)
level	30% (4/13)	24% (19/79)	21% (74/351)	12% (21/167)
npc	0% (0/10)	0% (0/47)	0% (0/233)	0% (0/136)
points	0% (0/2)	0% (0/7)	0% (0/19)	0% (0/4)
sprite	66% (4/6)	44% (20/45)	51% (66/128)	31% (21/66)
ui	0% (0/6)	0% (0/31)	0% (0/123)	0% (0/98)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)	0% (0/10)
LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)	0% (0/2)
PacmanConfigurationException	0% (0/1)	0% (0/2)	0% (0/4)	100% (0/0)



jpacman

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Clas
nl.tudelft.jpacman.level	🟥🟩	68%	🟥🟩	57%	73	156	102	345	20	70	4	
nl.tudelft.jpacman.npc.ghost	🟥🟩	71%	🟥🟩	55%	56	105	43	181	5	34	0	
nl.tudelft.jpacman.ui	🟥🟩	77%	🟥🟩	47%	54	86	21	144	7	31	0	
default	🟥	0%	🟥	0%	12	12	21	21	5	5	1	
nl.tudelft.jpacman.board	🟥🟩	86%	🟥🟩	58%	44	93	2	110	0	40	0	
nl.tudelft.jpacman.sprite	🟥🟩	86%	🟥🟩	59%	30	70	11	113	5	38	0	
nl.tudelft.jpacman	🟥🟩	69%	🟥	25%	12	30	18	52	6	24	1	
nl.tudelft.jpacman.points	🟥🟩	60%	🟩	75%	1	11	5	21	0	9	0	
nl.tudelft.jpacman.game	🟩	87%	🟥🟩	60%	10	24	4	45	2	14	0	
nl.tudelft.jpacman.npc	🟩	100%		n/a	0	4	0	8	0	4	0	
Total	1,207 of 4,697	74%	293 of 637	54%	292	591	227	1,040	50	269	6	

The code coverage looks like it is different.

Why so or why not?
Well JaCoCo seems to be a lot more specific. You can go to a specific method in a class and see the coverage and missed branches.

Did you find helpful the source code visualization from JaCoCo on uncovered branches?
Yes, it is helpful. And I like the UI better.

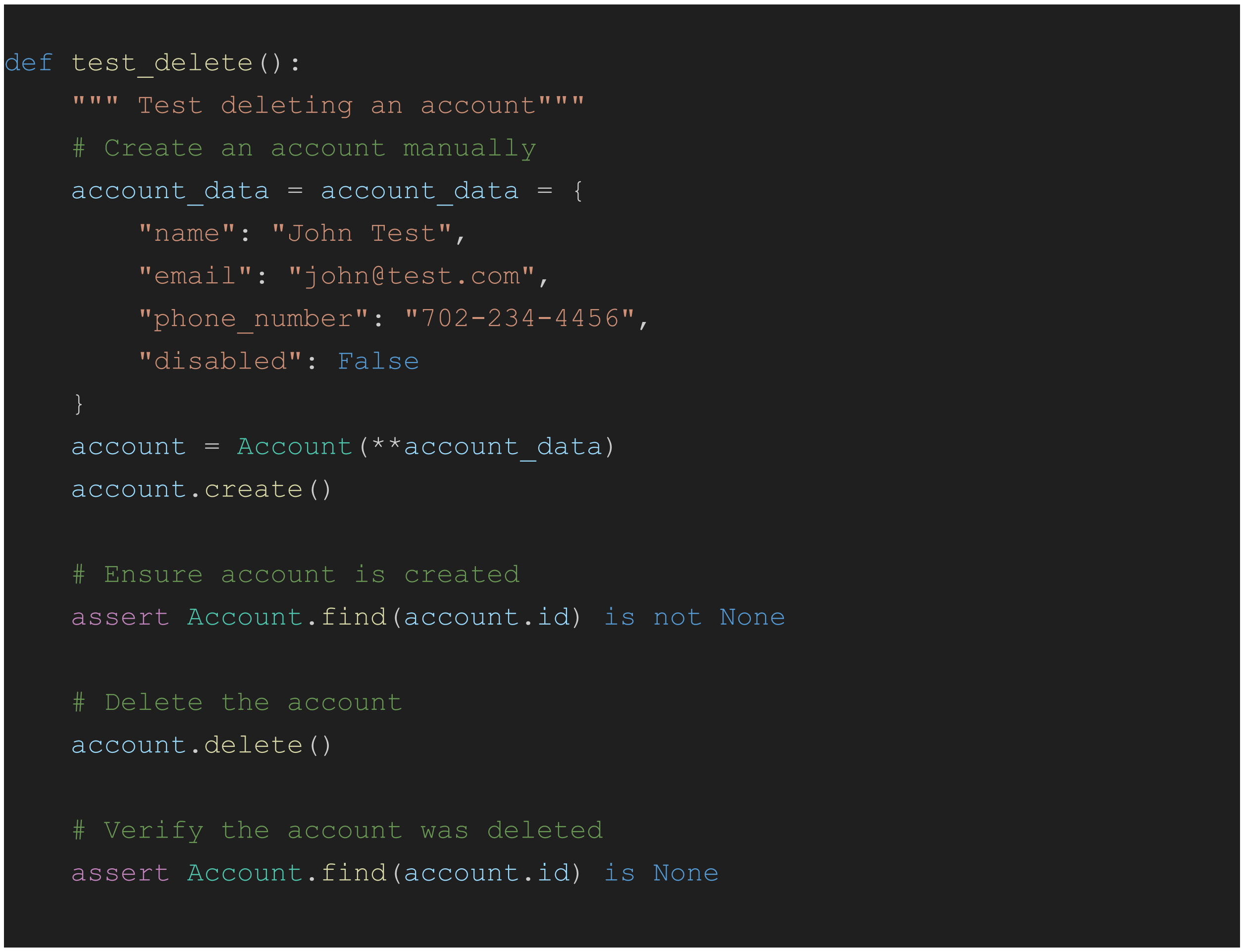
Which visualization did you prefer and why? IntelliJ's coverage window or JaCoCo?
I Prefer JaCoCo because the UI and visualization is better with the red and green bars and being able to see missed branches as well as being able to see specific method coverage.

Code: https://github.com/caesarx26/jpacman

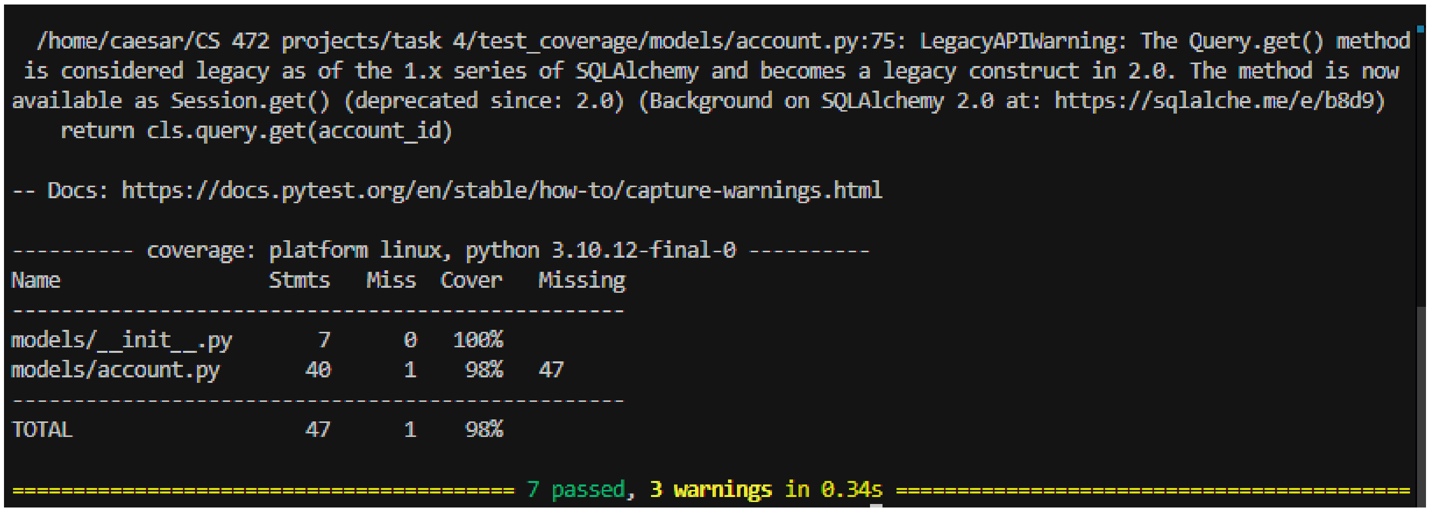
Task 4:

I added these tests:

```python
def test_from_dict():
    """ Test Account creation from a dictionary """
    rand = randrange(0, len(ACCOUNT_DATA))  # Generate a random index
    data = ACCOUNT_DATA[rand]  # get a random account
    account = Account()
    account.from_dict(data)
    account.create()

    # Check that attributes were set correctly
    assert account.name == data.get("name")
    assert account.email == data.get("email")
    assert account.phone_number == data.get("phone_number")
    assert account.disabled == data.get("disabled")


def test_update():
    """ Test updating an account"""
    # Create an account manually
    account_data = {
        "name": "John Test",
        "email": "john@test.com",
        "phone_number": "702-234-4456",
        "disabled": False
    }
    account = Account(**account_data)
    account.create()

    # Update the account
    account.name = "Update"
    account.update()

    # Verify the account was updated
    updated_account = Account.find(account.id)
    assert updated_account.name == "Update"
```



```python
def test_delete():
    """ Test deleting an account"""
    # Create an account manually
    account_data = account_data = {
        "name": "John Test",
        "email": "john@test.com",
        "phone_number": "702-234-4456",
        "disabled": False
    }
    account = Account(**account_data)
    account.create()

    # Ensure account is created
    assert Account.find(account.id) is not None

    # Delete the account
    account.delete()

    # Verify the account was deleted
    assert Account.find(account.id) is None
```



```
  /home/caesar/CS 472 projects/task 4/test_coverage/models/account.py:75: LegacyAPIWarning: The Query.get() method
 is considered legacy as of the 1.x series of SQLAlchemy and becomes a legacy construct in 2.0. The method is now
available as Session.get() (deprecated since: 2.0) (Background on SQLAlchemy 2.0 at: https://sqlalche.me/e/b8d9)
    return cls.query.get(account_id)

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html

---------- coverage: platform linux, python 3.10.12-final-0 ----------
Name                   Stmts   Miss  Cover   Missing
----------------------------------------------------
models/__init__.py         7      0   100%
models/account.py         40      1    98%   47
----------------------------------------------------
TOTAL                     47      1    98%

========================================== 7 passed, 3 warnings in 0.34s ==========================================
```

Task 5:

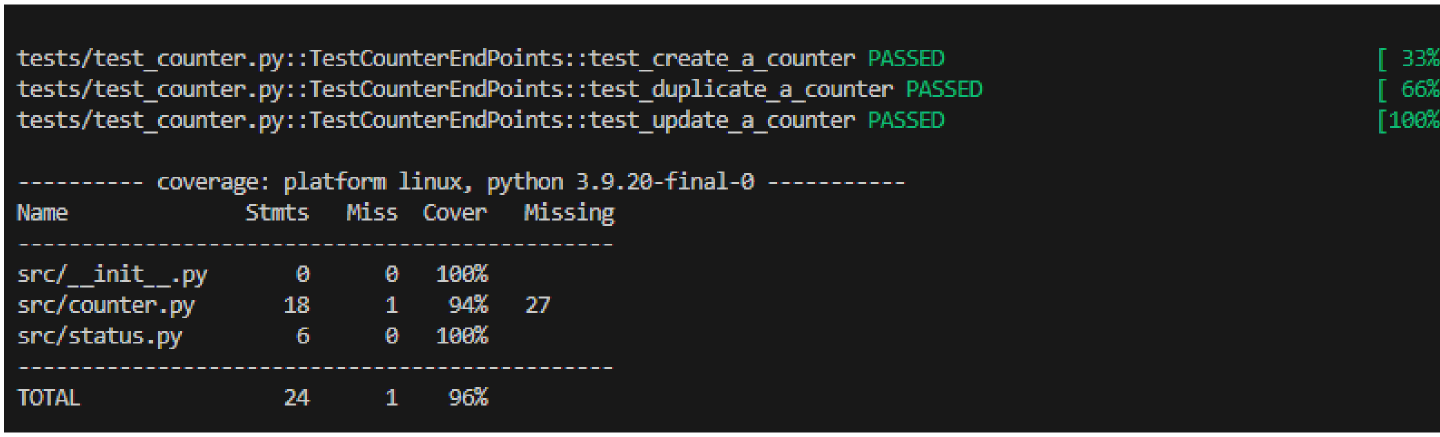
Added this test:

```
def test_update_a_counter(self, client):
        """test should update a counter value"""
        # Create the counter
        result = client.post('/counters/zen')
        assert result.status_code == status.HTTP_201_CREATED

        # Check the initial counter value
        initial_value = result.json['zen']

        assert initial_value == 0

        # Update the counter
        result = client.put('/counters/zen')
        assert result.status_code == status.HTTP_200_OK

        # Check the updated counter value
        updated_value = result.json['zen']
        assert updated_value == initial_value + 1
```

And added this method in counter for the put request:

```
@app.route('/counters/<name>', methods=['PUT'])
def update_counter(name):
    """update counter by 1"""
    app.logger.info(f"Request to update counter by 1: {name}")
    global COUNTERS
    if name not in COUNTERS:
        return {"Message":f"Counter {name} does not exist"},
status.HTTP_404_NOT_FOUND
    COUNTERS[name] += 1
    return {name: COUNTERS[name]}, status.HTTP_200_OK
```

The test failed at first because the method was not implemented but it also failed when I first implemented the post request because I initially copied the create_counter method and forgot to change the status code returned. There is also a check if the name is not found but this is not checked.



```
tests/test_counter.py::TestCounterEndPoints::test_create_a_counter PASSED                                           [ 33%
tests/test_counter.py::TestCounterEndPoints::test_duplicate_a_counter PASSED                                        [ 66%
tests/test_counter.py::TestCounterEndPoints::test_update_a_counter PASSED                                           [100%

---------- coverage: platform linux, python 3.9.20-final-0 -----------
Name               Stmts   Miss  Cover   Missing
-----------------------------------------------
src/__init__.py        0      0   100%
src/counter.py        18      1    94%   27
src/status.py          6      0   100%
-----------------------------------------------
TOTAL                 24      1    96%
```

My fork: https://github.com/caesarx26/CS472TeamsRepo