Ace Aceron
Testing Assignment
9/25/2023

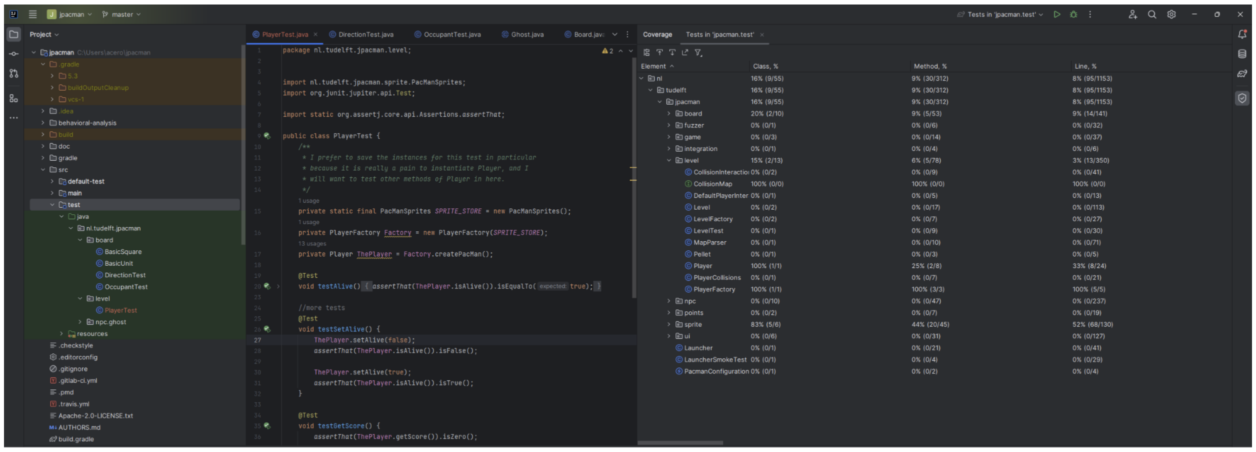
Link to fork repo: https://github.com/Hoodi3ac3/jpacman

Task 1:

Coverage is not good enough its only 1%

Task 2.1:

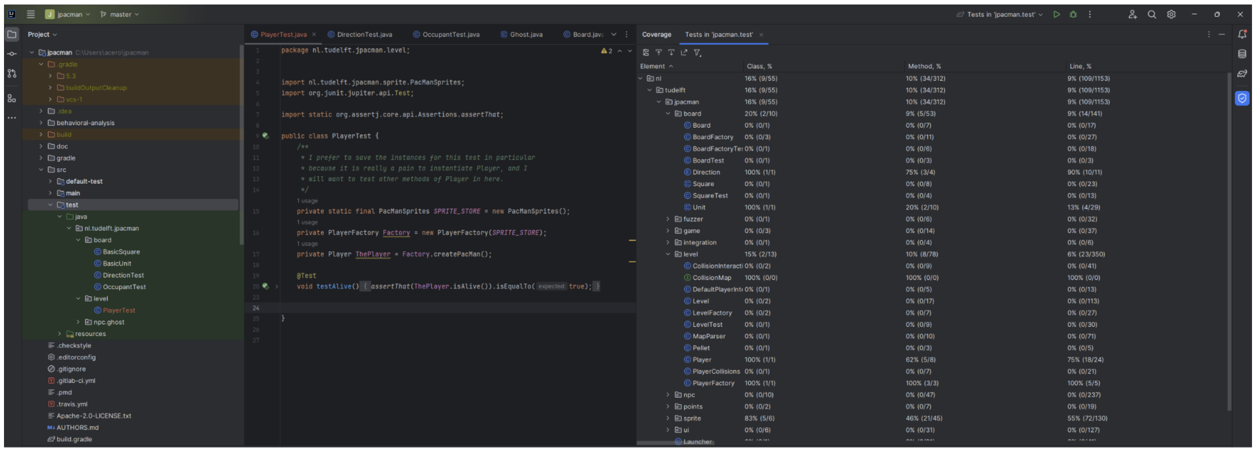
Before Adding Unit Tests



```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;

public class PlayerTest {
    /**
     * I prefer to save the instances for this test in particular
     * because it is really a pain to instantiate Player, and I
     * will want to test other methods of Player in here.
     */
    private static final PacManSprites SPRITE_STORE = new PacManSprites();
    private PlayerFactory Factory = new PlayerFactory(SPRITE_STORE);
    private Player ThePlayer = Factory.createPacMan();

    @Test
    void testAlive() { assertThat(ThePlayer.isAlive()).isEqualTo( expected: true); }

    //more tests
    @Test
    void testSetAlive() {
        ThePlayer.setAlive(false);
        assertThat(ThePlayer.isAlive()).isFalse();

        ThePlayer.setAlive(true);
        assertThat(ThePlayer.isAlive()).isTrue();
    }

    @Test
    void testGetScore() {
        assertThat(ThePlayer.getScore()).isZero();
```

```
Element                 Class, %      Method, %      Line, %
nl                      16% (9/55)    9% (30/312)    8% (95/1153)
  tudelft               16% (9/55)    9% (30/312)    8% (95/1153)
    jpacman             16% (9/55)    9% (30/312)    8% (95/1153)
      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)
        CollisionInteractio 0% (0/2)  0% (0/9)       0% (0/41)
        CollisionMap    100% (0/0)    100% (0/0)     100% (0/0)
        DefaultPlayerInter 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)    25% (2/8)      33% (8/24)
        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            83% (5/6)     44% (20/45)    52% (68/130)
      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)
      PacmanConfiguration 0% (0/1)    0% (0/2)       0% (0/4)
```

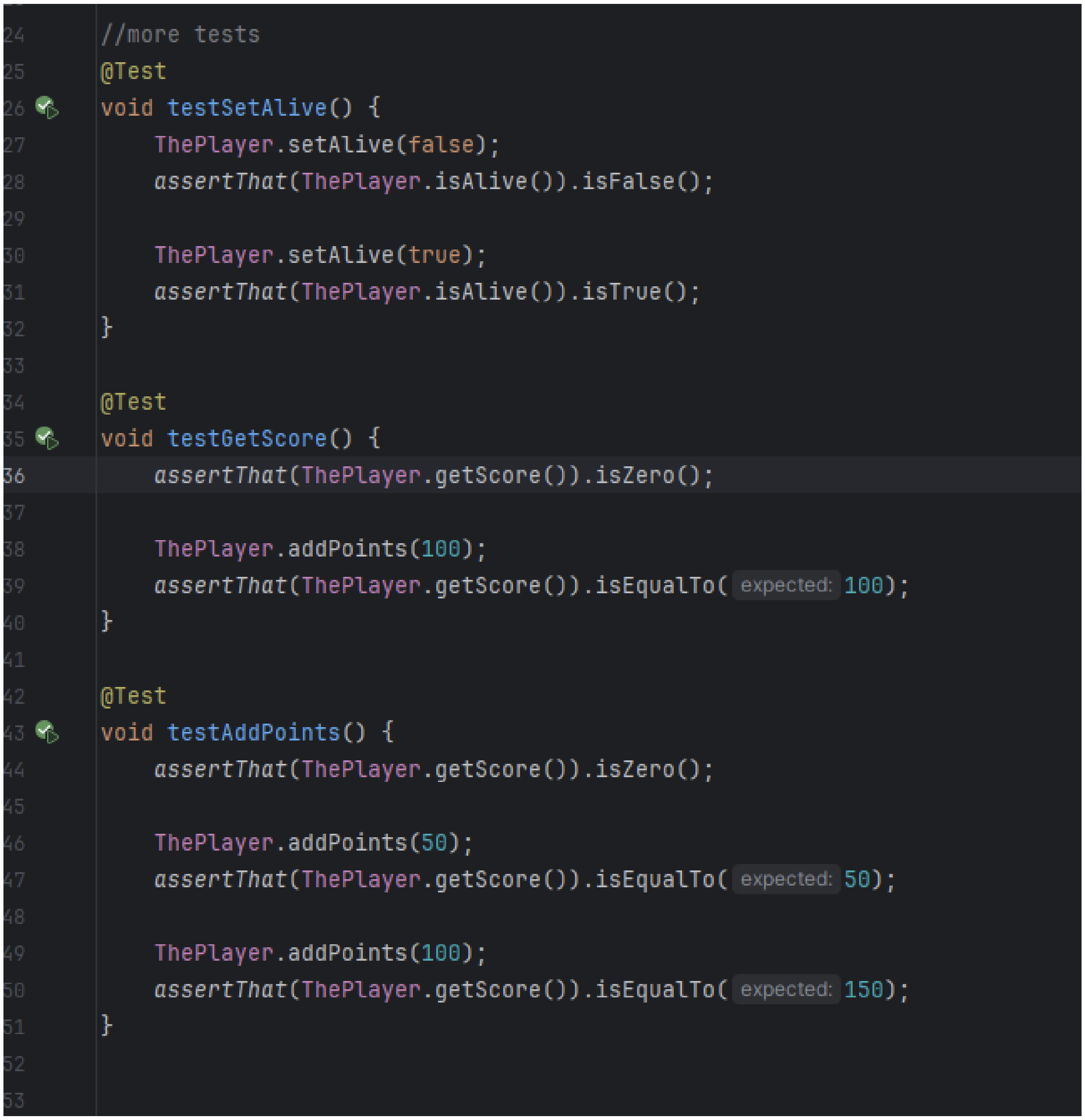
After Adding Unit Tests



```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;

public class PlayerTest {
    /**
     * I prefer to save the instances for this test in particular
     * because it is really a pain to instantiate Player, and I
     * will want to test other methods of Player in here.
     */
    private static final PacManSprites SPRITE_STORE = new PacManSprites();
    private PlayerFactory Factory = new PlayerFactory(SPRITE_STORE);
    private Player ThePlayer = Factory.createPacMan();

    @Test
    void testAlive() { assertThat(ThePlayer.isAlive()).isEqualTo( expected: true); }


}
```

```
Element                 Class, %      Method, %      Line, %
nl                      16% (9/55)    10% (34/312)   9% (109/1153)
  tudelft               16% (9/55)    10% (34/312)   9% (109/1153)
    jpacman             16% (9/55)    10% (34/312)   9% (109/1153)
      board             20% (2/10)    9% (5/53)      9% (14/141)
        Board           0% (0/1)      0% (0/7)       0% (0/17)
        BoardFactory    0% (0/3)      0% (0/11)      0% (0/27)
        BoardFactoryTe: 0% (0/1)      0% (0/6)       0% (0/18)
        BoardTest       0% (0/1)      0% (0/3)       0% (0/3)
        Direction       100% (1/1)    75% (3/4)      90% (10/11)
        Square          0% (0/1)      0% (0/8)       0% (0/23)
        SquareTest      0% (0/1)      0% (0/4)       0% (0/13)
        Unit            100% (1/1)    20% (2/10)     13% (4/29)
      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)    10% (8/78)     6% (23/350)
        CollisionInteracti 0% (0/2)   0% (0/9)       0% (0/41)
        CollisionMap    100% (0/0)    100% (0/0)     100% (0/0)
        DefaultPlayerInt 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)    62% (5/8)      75% (18/24)
        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            83% (5/6)     46% (21/45)    55% (72/130)
      ui                0% (0/6)      0% (0/31)      0% (0/127)
      Launcher
```

The Tests I did



```
//more tests
@Test
void testSetAlive() {
    ThePlayer.setAlive(false);
    assertThat(ThePlayer.isAlive()).isFalse();

    ThePlayer.setAlive(true);
    assertThat(ThePlayer.isAlive()).isTrue();
}

@Test
void testGetScore() {
    assertThat(ThePlayer.getScore()).isZero();

    ThePlayer.addPoints(100);
    assertThat(ThePlayer.getScore()).isEqualTo( expected: 100);
}

@Test
void testAddPoints() {
    assertThat(ThePlayer.getScore()).isZero();

    ThePlayer.addPoints(50);
    assertThat(ThePlayer.getScore()).isEqualTo( expected: 50);

    ThePlayer.addPoints(100);
    assertThat(ThePlayer.getScore()).isEqualTo( expected: 150);
}
```

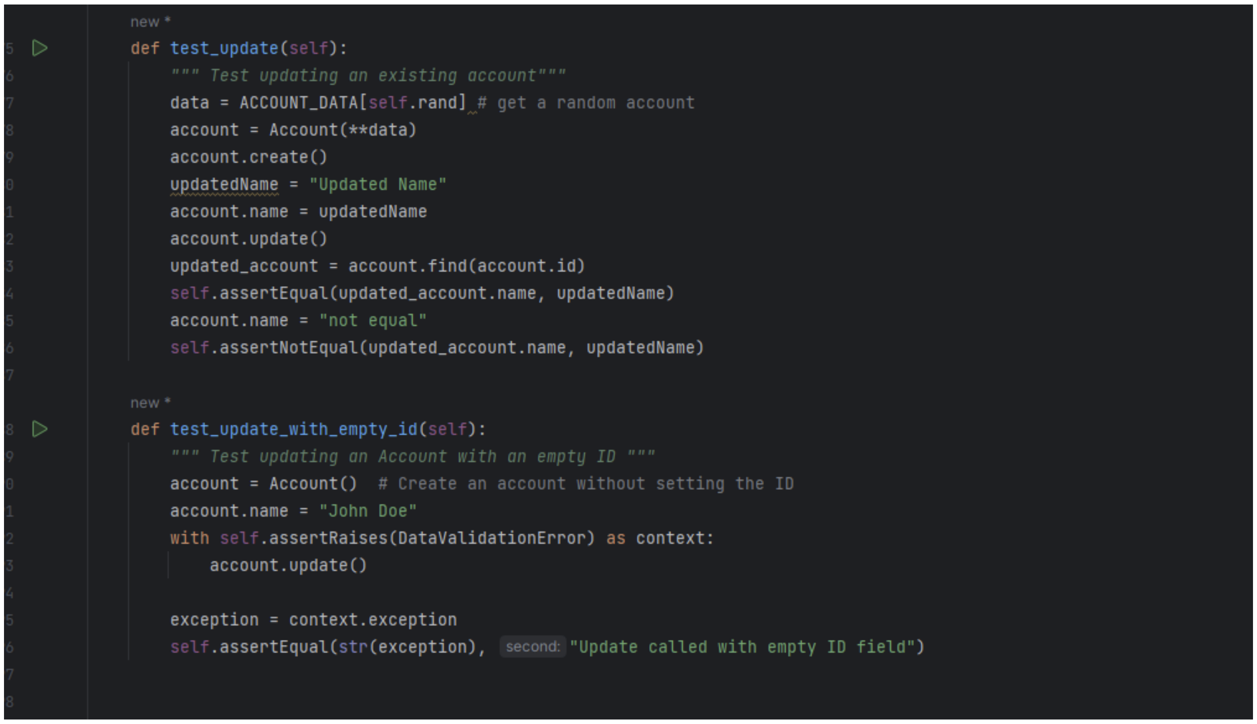
It is clear that the coverage for methods in level/player class increased after adding the unit tests.

Task 3:

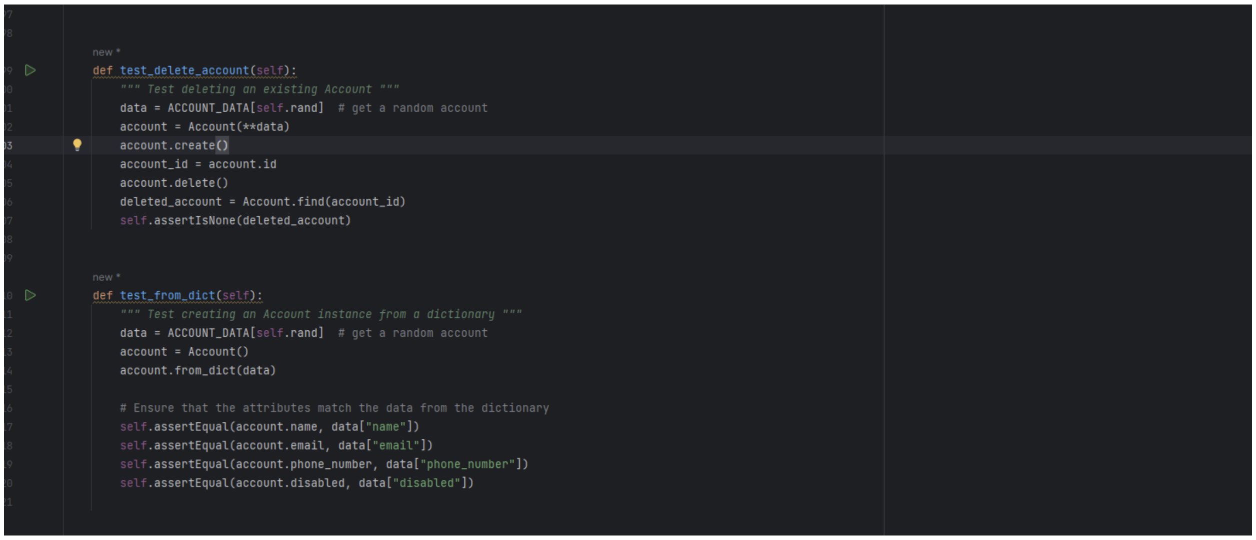
1. There are differences between using JaCoco instead of IntelliJ because they use different mechanisms to collect coverage data.
2. Yes, the source code visualization from JaCoCo on uncovered branches can be extremely helpful. It provides detailed insights into which branches of the code are covered (or partially covered) by tests and which branches are not.

3. I like IntelliJ better because it provides real-time feedback within the IDE during development and testing.

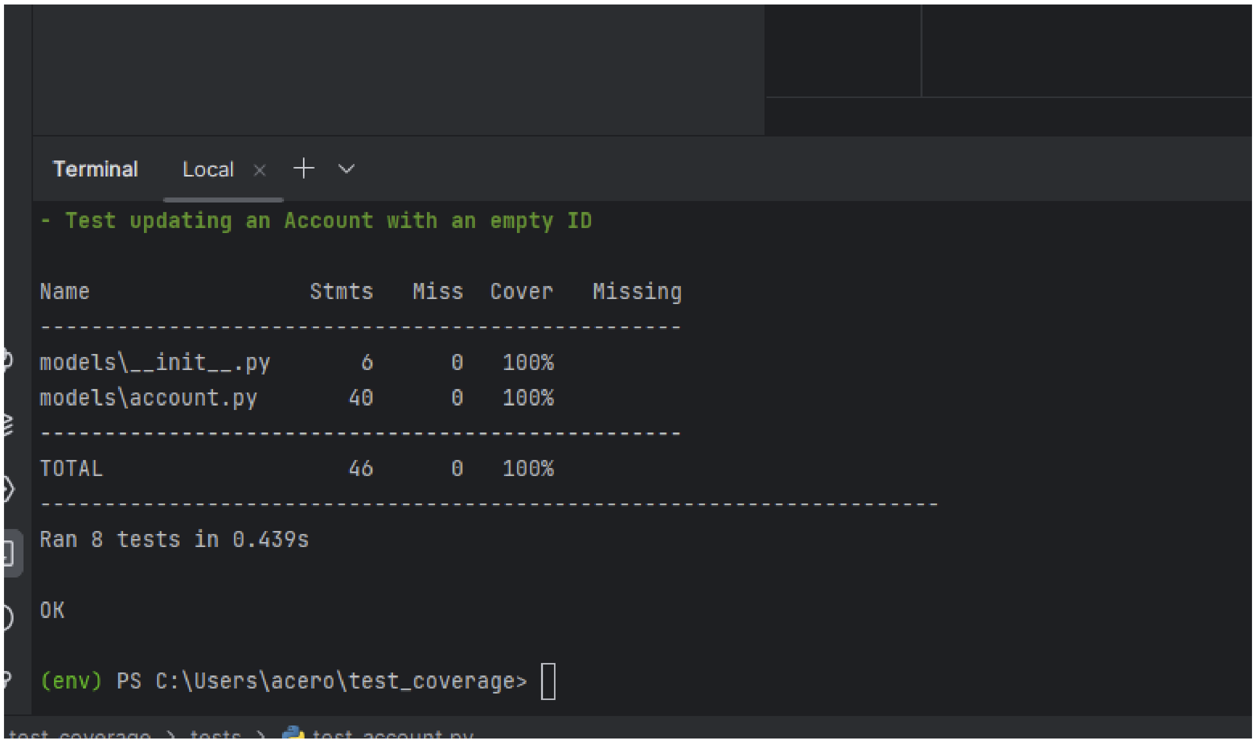
Task 4:



```python
new *
def test_update(self):
    """ Test updating an existing account"""
    data = ACCOUNT_DATA[self.rand] # get a random account
    account = Account(**data)
    account.create()
    updatedName = "Updated Name"
    account.name = updatedName
    account.update()
    updated_account = account.find(account.id)
    self.assertEqual(updated_account.name, updatedName)
    account.name = "not equal"
    self.assertNotEqual(updated_account.name, updatedName)

new *
def test_update_with_empty_id(self):
    """ Test updating an Account with an empty ID """
    account = Account()  # Create an account without setting the ID
    account.name = "John Doe"
    with self.assertRaises(DataValidationError) as context:
        account.update()

    exception = context.exception
    self.assertEqual(str(exception),  second: "Update called with empty ID field")
```



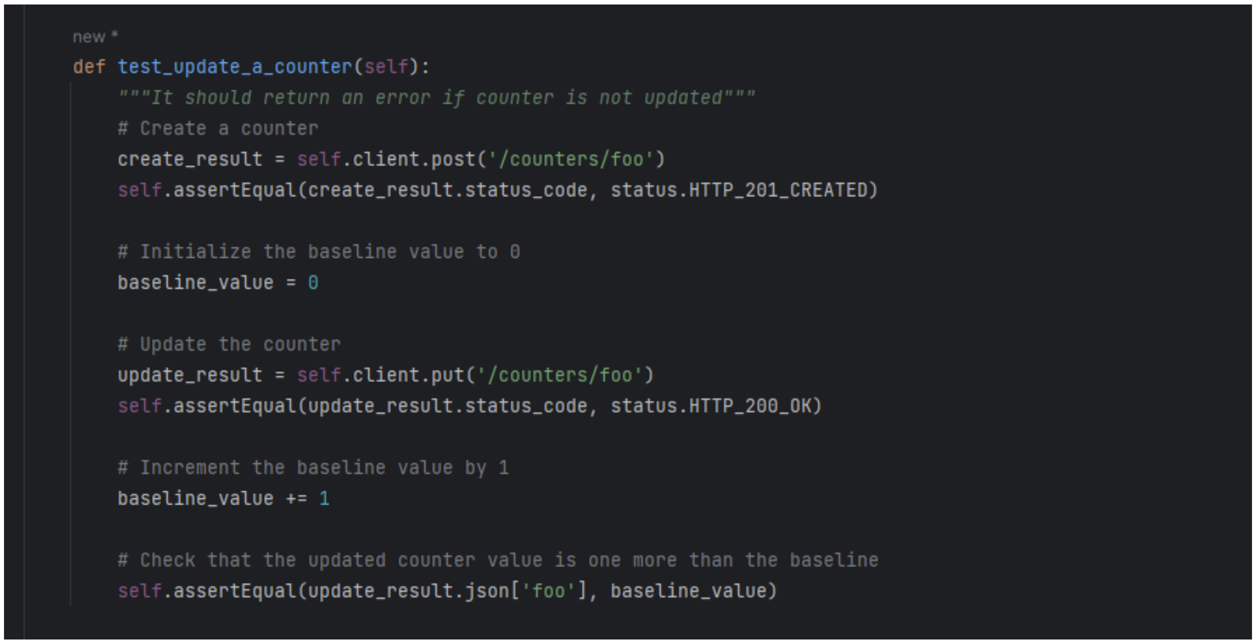
```python
new *
def test_delete_account(self):
    """ Test deleting an existing Account """
    data = ACCOUNT_DATA[self.rand]  # get a random account
    account = Account(**data)
    account.create()
    account_id = account.id
    account.delete()
    deleted_account = Account.find(account_id)
    self.assertIsNone(deleted_account)


new *
def test_from_dict(self):
    """ Test creating an Account instance from a dictionary """
    data = ACCOUNT_DATA[self.rand]  # get a random account
    account = Account()
    account.from_dict(data)

    # Ensure that the attributes match the data from the dictionary
    self.assertEqual(account.name, data["name"])
    self.assertEqual(account.email, data["email"])
    self.assertEqual(account.phone_number, data["phone_number"])
    self.assertEqual(account.disabled, data["disabled"])
```



Terminal Local

```
- Test updating an Account with an empty ID

Name                  Stmts   Miss  Cover   Missing
---------------------------------------------------
models\__init__.py        6      0   100%
models\account.py        40      0   100%
---------------------------------------------------
TOTAL                    46      0   100%
----------------------------------------------------------------------
Ran 8 tests in 0.439s

OK

(env) PS C:\Users\acero\test_coverage>
```

Task 5:



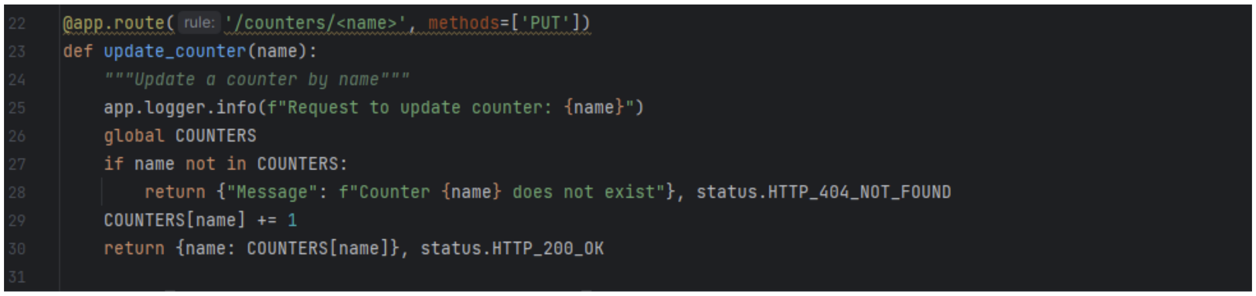
```
def test_update_a_counter(self):
    """It should return an error if counter is not updated"""
    # Create a counter
    create_result = self.client.post('/counters/foo')
    self.assertEqual(create_result.status_code, status.HTTP_201_CREATED)

    # Initialize the baseline value to 0
    baseline_value = 0

    # Update the counter
    update_result = self.client.put('/counters/foo')
    self.assertEqual(update_result.status_code, status.HTTP_200_OK)

    # Increment the baseline value by 1
    baseline_value += 1

    # Check that the updated counter value is one more than the baseline
    self.assertEqual(update_result.json['foo'], baseline_value)
```

So after implementing this test case that would test for updating a counter, I would get a red error case on the line

Update_result = self.client.put('/counters/foo')

This was an error 405 which meant the method "put" was not allowed

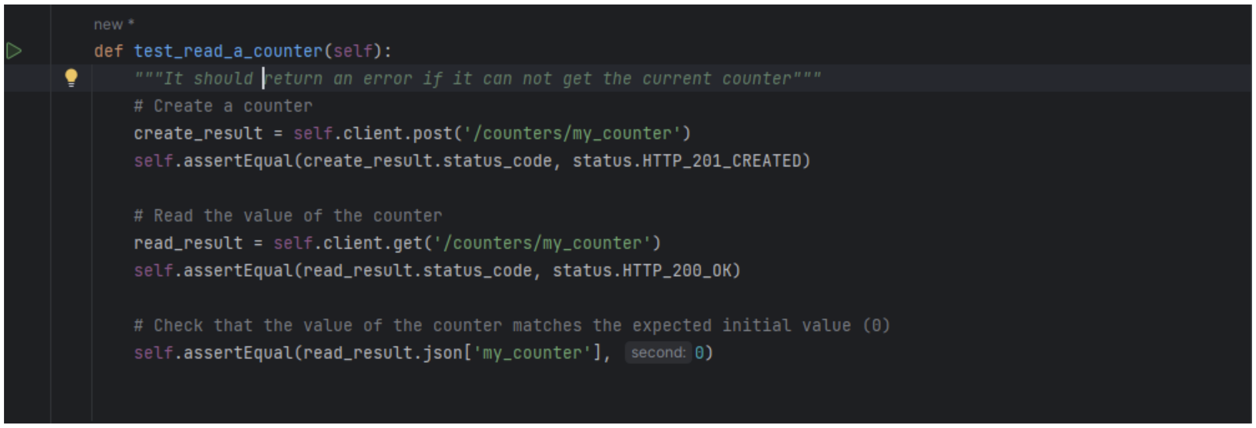
So in counter.py I implemented this:



```python
@app.route( rule: '/counters/<name>', methods=['PUT'])
def update_counter(name):
    """Update a counter by name"""
    app.logger.info(f"Request to update counter: {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
```

After implementing that code snipped, it allowed the test case to bypass the error and give a green ok test

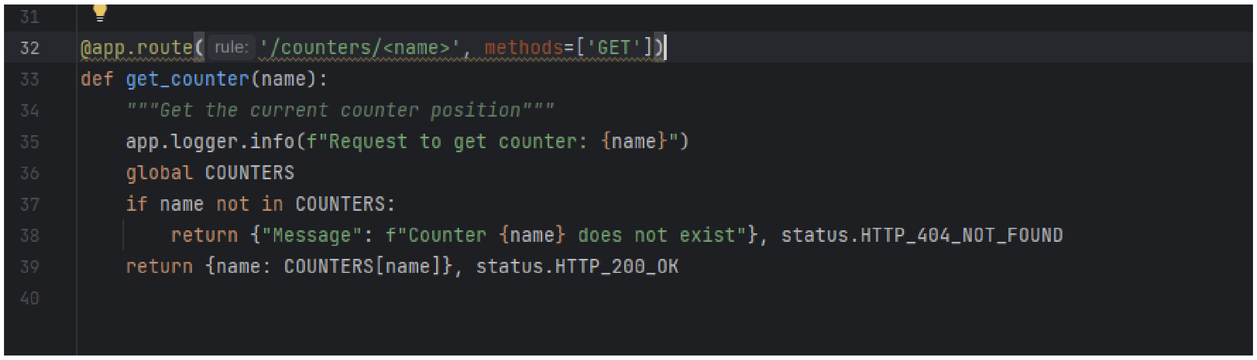
Then I implemented a test case for reading a counter:



```python
new *
def test_read_a_counter(self):
    """It should return an error if it can not get the current counter"""
    # Create a counter
    create_result = self.client.post('/counters/my_counter')
    self.assertEqual(create_result.status_code, status.HTTP_201_CREATED)

    # Read the value of the counter
    read_result = self.client.get('/counters/my_counter')
    self.assertEqual(read_result.status_code, status.HTTP_200_OK)

    # Check that the value of the counter matches the expected initial value (0)
    self.assertEqual(read_result.json['my_counter'],  second: 0)
```

I would get an error on the line
Read_result = self.client.get('/counters/my_counter')
And it would be the same error code as last time which meant it would not allow the method "get"

So I implemented this code in counter.py:



```python
@app.route( rule: '/counters/<name>', methods=['GET'])
def get_counter(name):
    """Get the current counter position"""
    app.logger.info(f"Request to get counter: {name}")
    global COUNTERS
    if name not in COUNTERS:
        return {"Message": f"Counter {name} does not exist"}, status.HTTP_404_NOT_FOUND
    return {name: COUNTERS[name]}, status.HTTP_200_OK
```

This resulted in a green ok code after running nosetests