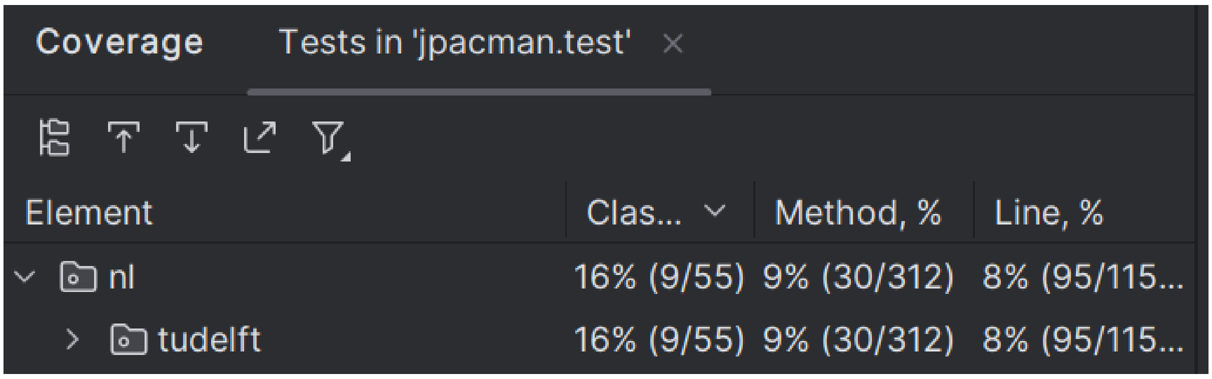
2.1

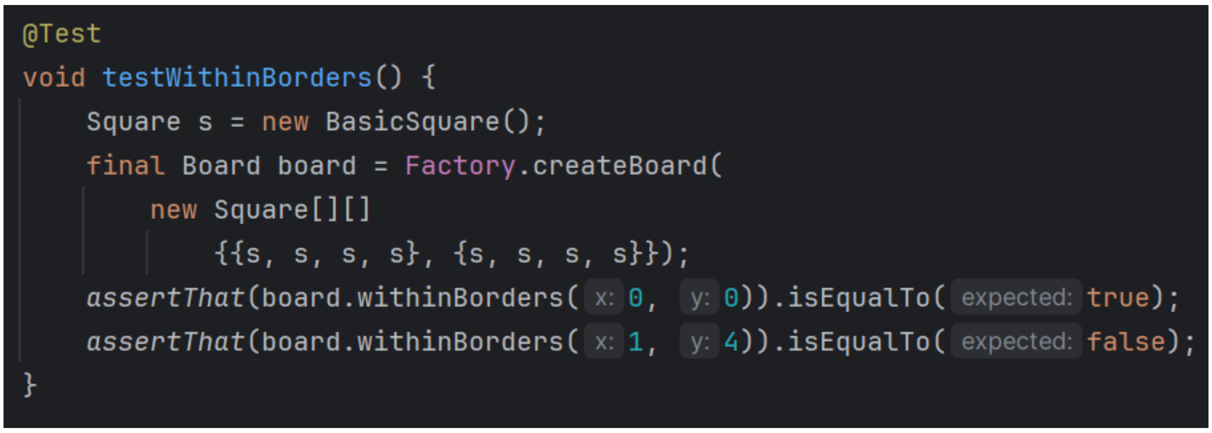
Coverage before implementing tests:



Coverage   Tests in 'jpacman.test'

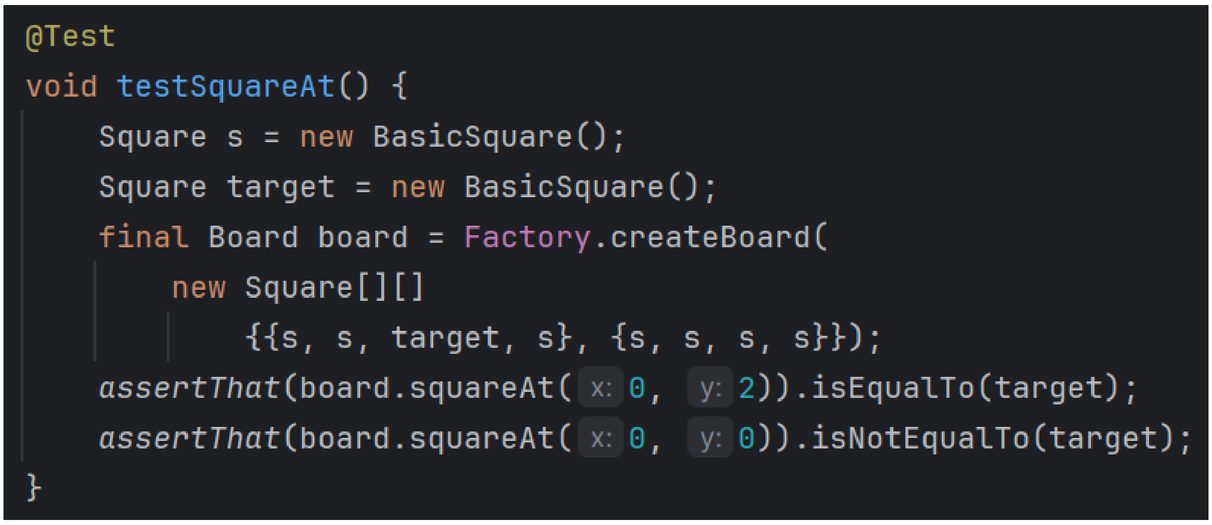
Element	Clas...	Method, %	Line, %
nl	16% (9/55)	9% (30/312)	8% (95/115...
tudelft	16% (9/55)	9% (30/312)	8% (95/115...

Fully qualified method name:
src/main/java/nl/tudelft/jpacman/board/BoardTest.testWithinBorders



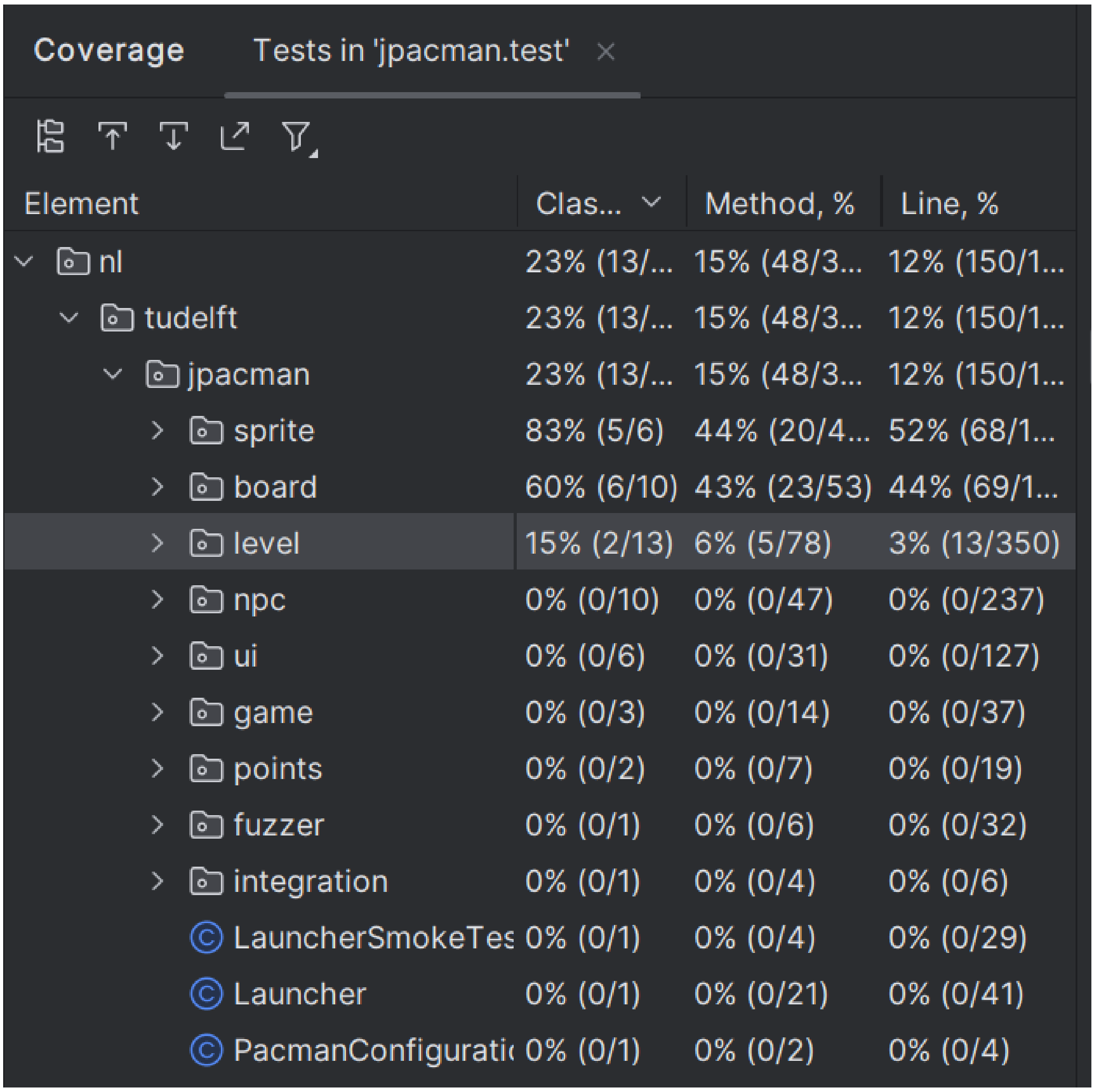
```
@Test
void testWithinBorders() {
    Square s = new BasicSquare();
    final Board board = Factory.createBoard(
        new Square[][]
            {{s, s, s, s}, {s, s, s, s}});
    assertThat(board.withinBorders( x: 0,  y: 0)).isEqualTo( expected: true);
    assertThat(board.withinBorders( x: 1,  y: 4)).isEqualTo( expected: false);
}
```

Fully qualified method name:
src/main/java/nl/tudelft/jpacman/board/BoardTest.testSquareAt



```
@Test
void testSquareAt() {
    Square s = new BasicSquare();
    Square target = new BasicSquare();
    final Board board = Factory.createBoard(
        new Square[][]
            {{s, s, target, s}, {s, s, s, s}});
    assertThat(board.squareAt( x: 0,  y: 2)).isEqualTo(target);
    assertThat(board.squareAt( x: 0,  y: 0)).isNotEqualTo(target);
}
```

Coverage after implementing tests:

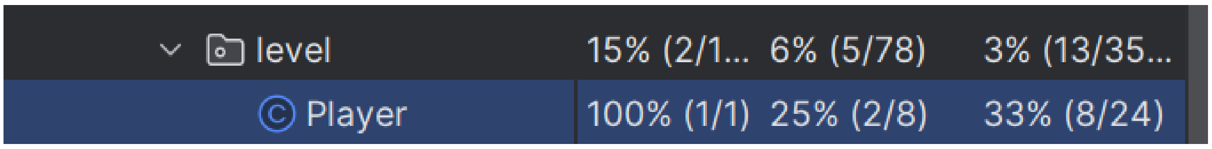


Coverage   Tests in 'jpacman.test'

Element	Clas...	Method, %	Line, %
nl	23% (13/...	15% (48/3...	12% (150/1...
tudelft	23% (13/...	15% (48/3...	12% (150/1...
jpacman	23% (13/...	15% (48/3...	12% (150/1...
sprite	83% (5/6)	44% (20/4...	52% (68/1...
board	60% (6/10)	43% (23/53)	44% (69/1...
level	15% (2/13)	6% (5/78)	3% (13/350)
npc	0% (0/10)	0% (0/47)	0% (0/237)
ui	0% (0/6)	0% (0/31)	0% (0/127)
game	0% (0/3)	0% (0/14)	0% (0/37)
points	0% (0/2)	0% (0/7)	0% (0/19)
fuzzer	0% (0/1)	0% (0/6)	0% (0/32)
integration	0% (0/1)	0% (0/4)	0% (0/6)
LauncherSmokeTes	0% (0/1)	0% (0/4)	0% (0/29)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)
PacmanConfigurati	0% (0/1)	0% (0/2)	0% (0/4)

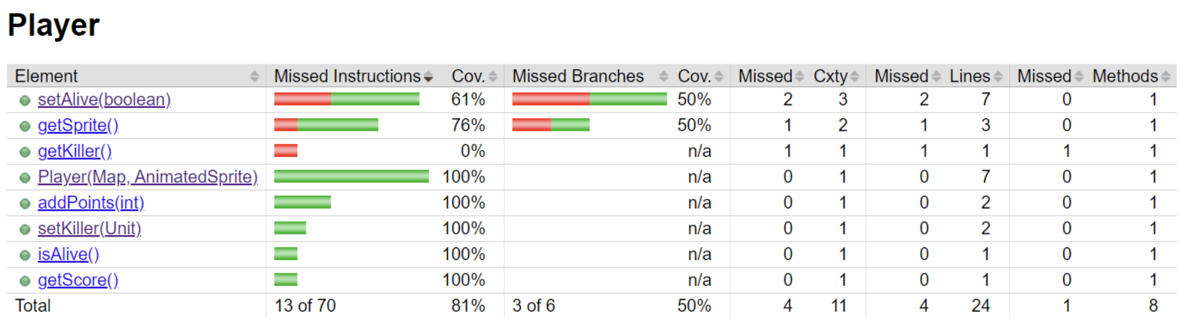
3

IntelliJ report:



level	15% (2/1...	6% (5/78)	3% (13/35...
Player	100% (1/1)	25% (2/8)	33% (8/24)

JaCoCo report:



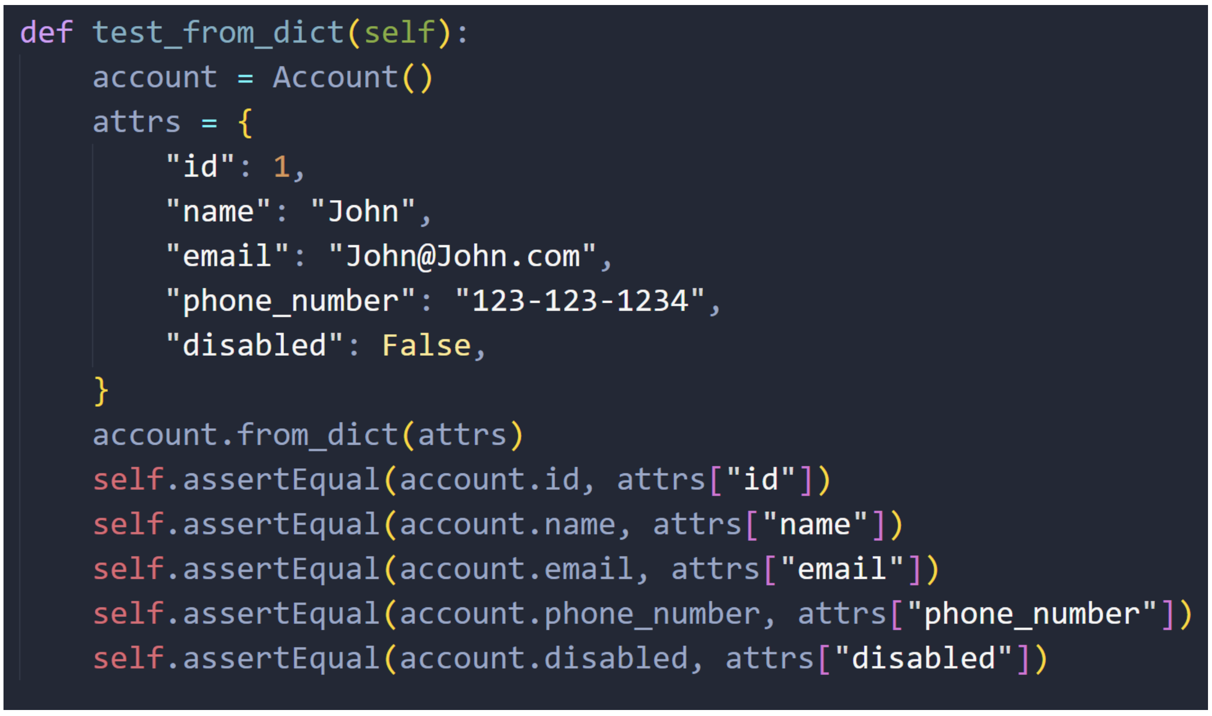
Player

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
setAlive(boolean)	🟥🟩	61%	🟥🟩	50%	2	3	2	7	0	1
getSprite()	🟥🟩	76%	🟥🟩	50%	1	2	1	3	0	1
getKiller()	🟥	0%		n/a	1	1	1	1	1	1
Player(Map, AnimatedSprite)	🟩	100%		n/a	0	1	0	7	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	13 of 70	81%	3 of 6	50%	4	11	4	24	1	8

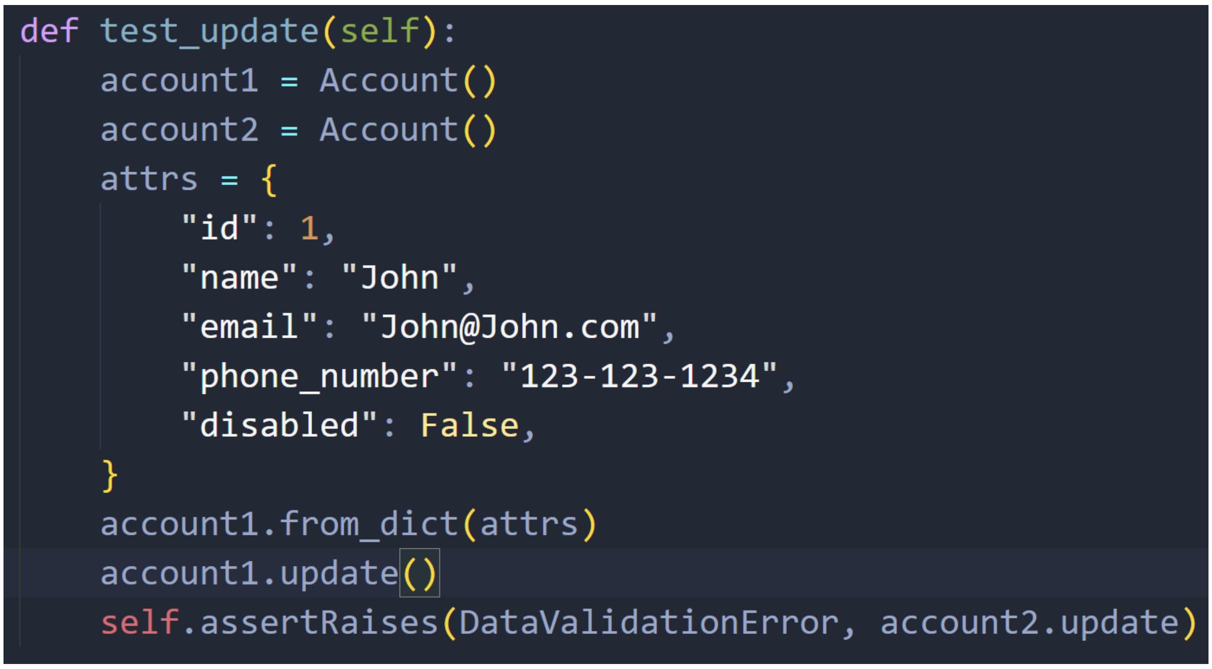
The two reports are sort of similar, but also completely different. The JaCoCo report is more concerned with all of the possible paths through the code, as given by the Cxty value in the table. The IntelliJ report only focuses on if a function or line is covered at all.

The JaCoCo report is significantly more detailed and nicer to work with, though more confusing to initially understand what the values are corresponding to.

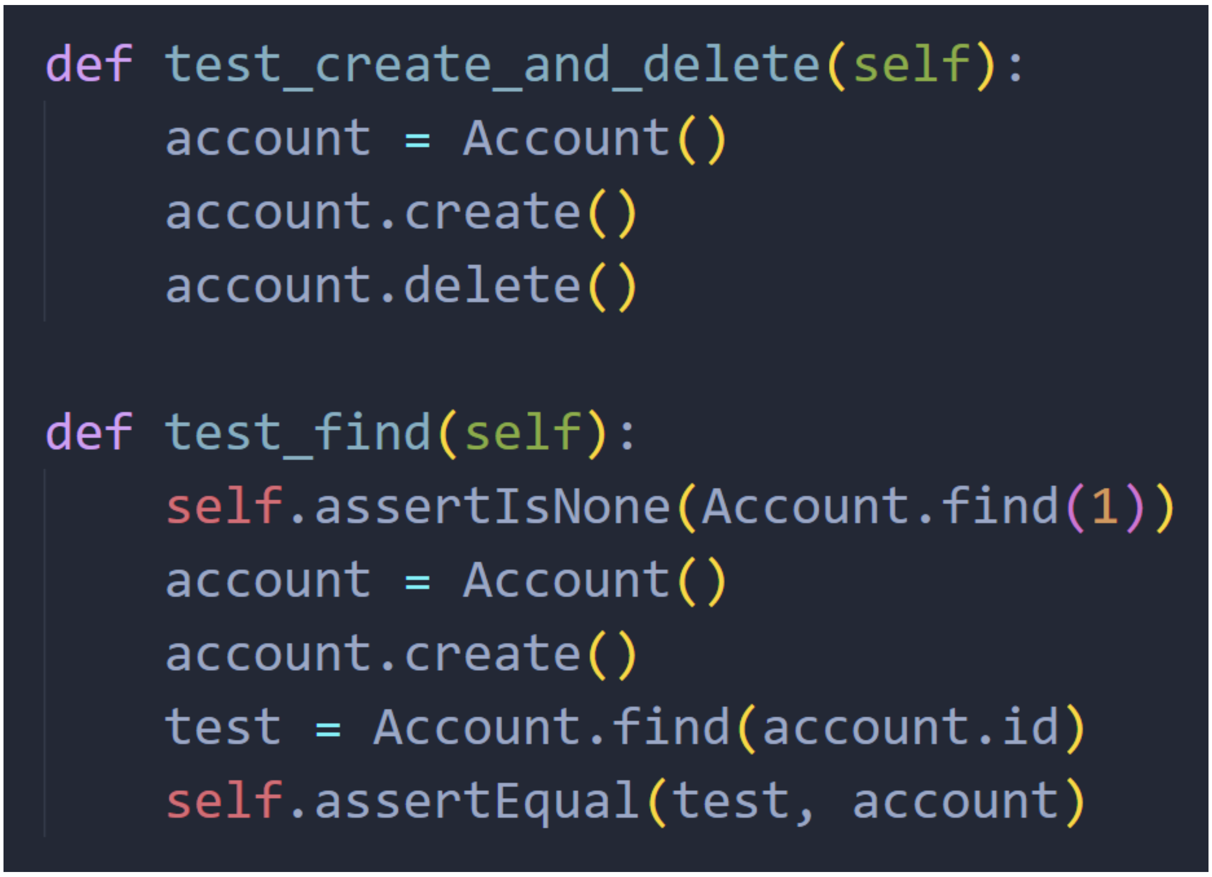
4



```python
def test_from_dict(self):
    account = Account()
    attrs = {
        "id": 1,
        "name": "John",
        "email": "John@John.com",
        "phone_number": "123-123-1234",
        "disabled": False,
    }
    account.from_dict(attrs)
    self.assertEqual(account.id, attrs["id"])
    self.assertEqual(account.name, attrs["name"])
    self.assertEqual(account.email, attrs["email"])
    self.assertEqual(account.phone_number, attrs["phone_number"])
    self.assertEqual(account.disabled, attrs["disabled"])
```

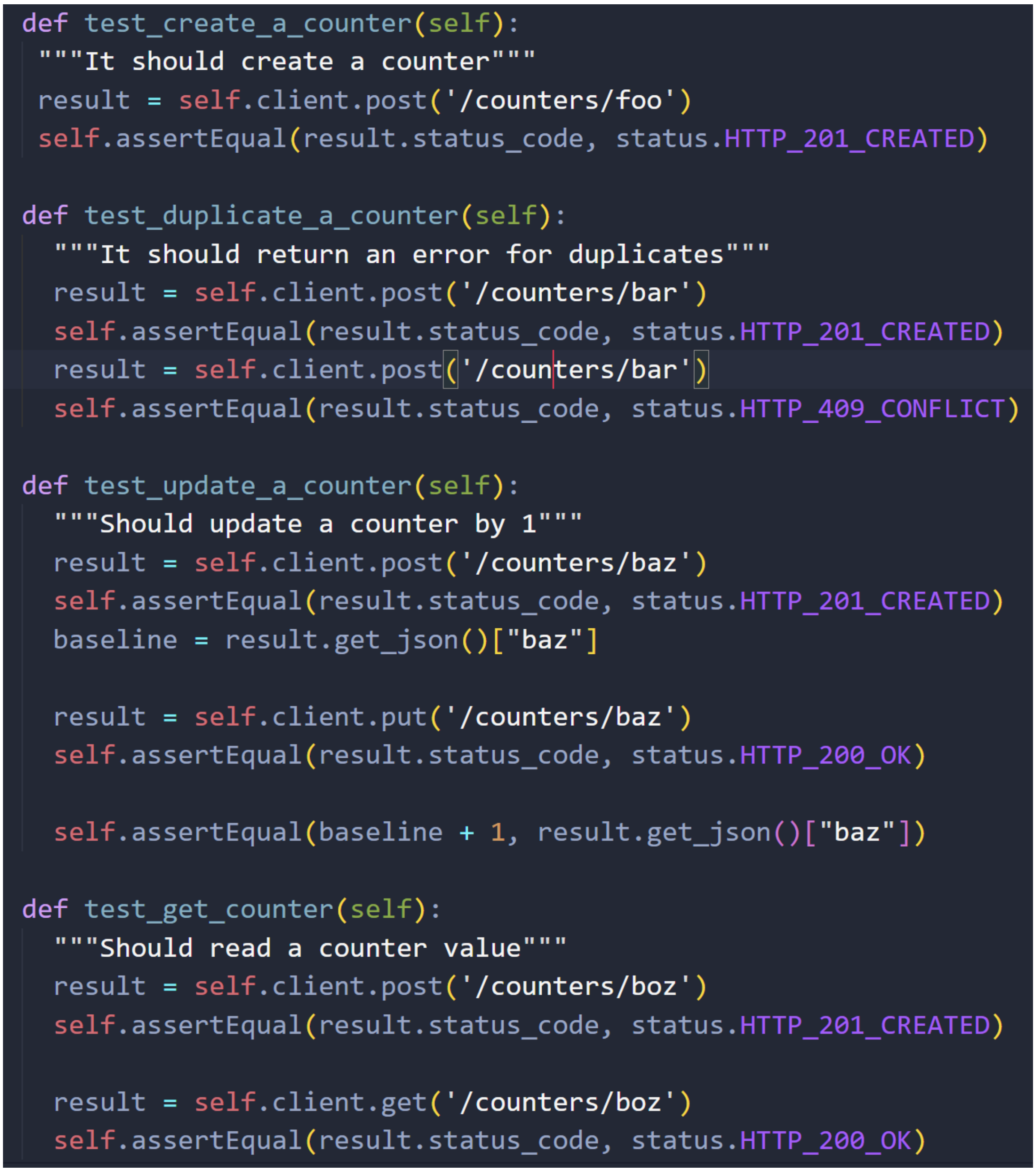


```python
def test_update(self):
    account1 = Account()
    account2 = Account()
    attrs = {
        "id": 1,
        "name": "John",
        "email": "John@John.com",
        "phone_number": "123-123-1234",
        "disabled": False,
    }
    account1.from_dict(attrs)
    account1.update()
    self.assertRaises(DataValidationError, account2.update)
```

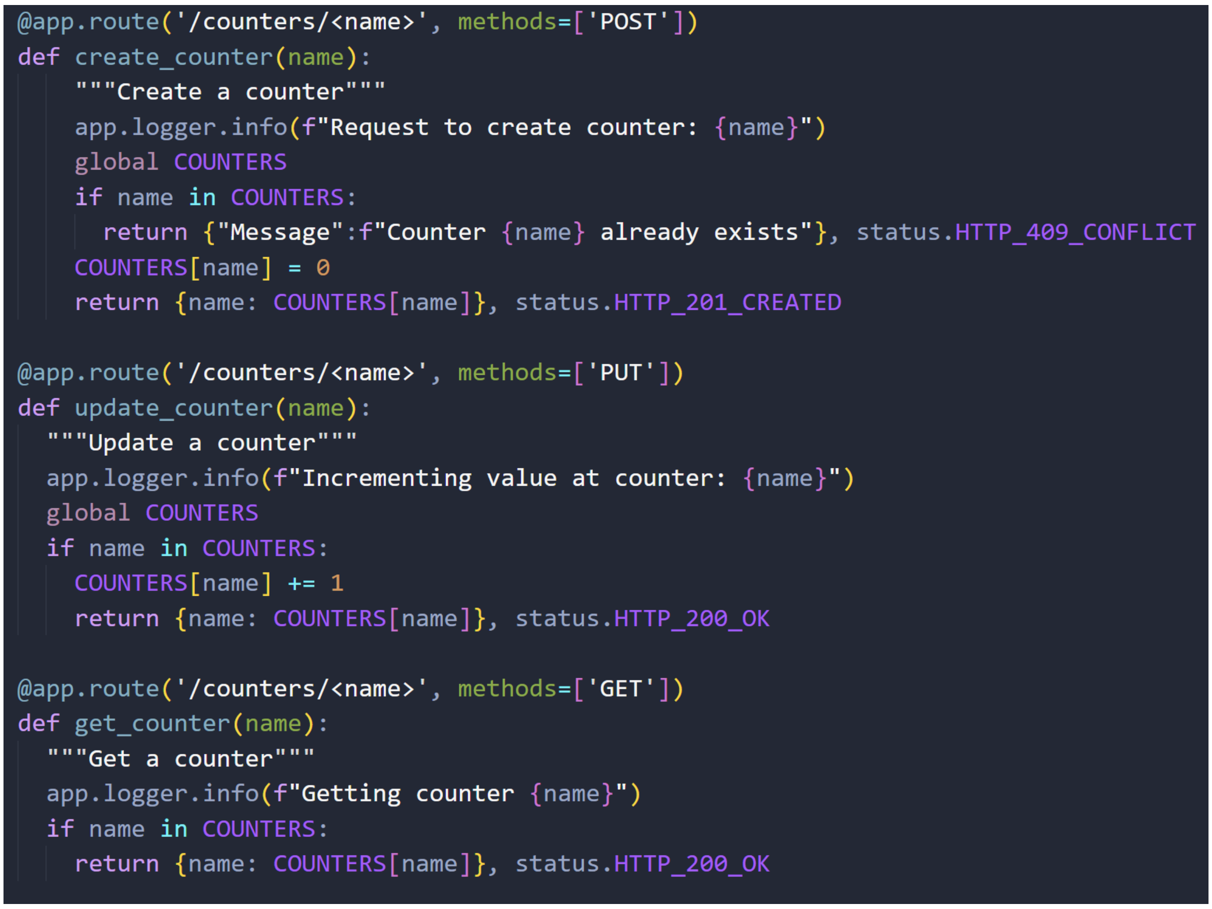


```python
def test_create_and_delete(self):
    account = Account()
    account.create()
    account.delete()

def test_find(self):
    self.assertIsNone(Account.find(1))
    account = Account()
    account.create()
    test = Account.find(account.id)
    self.assertEqual(test, account)
```

5



```
def test_create_a_counter(self):
    """It should create a counter"""
    result = self.client.post('/counters/foo')
    self.assertEqual(result.status_code, status.HTTP_201_CREATED)

def test_duplicate_a_counter(self):
    """It should return an error for duplicates"""
    result = self.client.post('/counters/bar')
    self.assertEqual(result.status_code, status.HTTP_201_CREATED)
    result = self.client.post('/counters/bar')
    self.assertEqual(result.status_code, status.HTTP_409_CONFLICT)

def test_update_a_counter(self):
    """Should update a counter by 1"""
    result = self.client.post('/counters/baz')
    self.assertEqual(result.status_code, status.HTTP_201_CREATED)
    baseline = result.get_json()["baz"]

    result = self.client.put('/counters/baz')
    self.assertEqual(result.status_code, status.HTTP_200_OK)

    self.assertEqual(baseline + 1, result.get_json()["baz"])

def test_get_counter(self):
    """Should read a counter value"""
    result = self.client.post('/counters/boz')
    self.assertEqual(result.status_code, status.HTTP_201_CREATED)

    result = self.client.get('/counters/boz')
    self.assertEqual(result.status_code, status.HTTP_200_OK)
```



```
@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

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

@app.route('/counters/<name>', methods=['GET'])
def get_counter(name):
  """Get a counter"""
  app.logger.info(f"Getting counter {name}")
  if name in COUNTERS:
    return {name: COUNTERS[name]}, status.HTTP_200_OK
```