Link to repo: https://github.com/Benek6h/CS472-Team.git

Tast 2.1:

AnimatedSprite.java#getWidth()
AnimatedSprite.java#getHeight()

```
new *
public class getWidthAndHeightSprite {
    /**
     * Do we get the right sprite?
     */
    1 usage
    private static final PacManSprites SPRITE_STORE = new PacManSprites();
    1 usage
    private PlayerFactory Factory = new PlayerFactory(SPRITE_STORE);
    3 usages
    private Player ThePlayer = Factory.createPacMan();
    new *
    @Test
    void sprite() {
        if(!ThePlayer.isAlive()){
            assertThat(ThePlayer.getSprite().getWidth()).isEqualTo( expected: 16);
            assertThat(ThePlayer.getSprite().getHeight()).isEqualTo( expected: 16);
        }
    }
}
```

This code will make a new player sprite and will get the Width and Height, checking to see if they return the proper value.

Before:

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)

After:

Element	Class, %	Method, %	Line, %
nl	16% (9/55)	10% (32/312)	8% (98/1153)
tudelft	16% (9/55)	10% (32/312)	8% (98/1153)
jpacman	16% (9/55)	10% (32/312)	8% (98/1153)

Player.java#getKiller()

```
public class GetKiller {
    /**
     * Do we get empty when alive?
     */
    1 usage
    private static final PacManSprites SPRITE_STORE = new PacManSprites();
    1 usage
    private PlayerFactory Factory = new PlayerFactory(SPRITE_STORE);
    2 usages
    private Player ThePlayer = Factory.createPacMan();
    new *
    @Test
    void killer() {
        if(ThePlayer.isAlive()) {
            assertThat(ThePlayer.getKiller()).isEqualTo( expected: null);
        }
```

This code will create a new player and when the player is alive it should return null, otherwise, it should return the ghost name that killed them

Before:

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)

After:

Element	Class, %	Method, %	Line, %
nl	16% (9/55)	9% (31/312)	8% (96/1153)
tudelft	16% (9/55)	9% (31/312)	8% (96/1153)
jpacman	16% (9/55)	9% (31/312)	8% (96/1153)

Player.java#getScore()

```java
public class GetScore {
    /**
     * Do we get the correct score
     */
    1 usage
    private static final PacManSprites SPRITE_STORE = new PacManSprites();
    1 usage
    private PlayerFactory Factory = new PlayerFactory(SPRITE_STORE);
    4 usages
    private Player ThePlayer = Factory.createPacMan();
    new *
    @Test
    void score() {
        int score = 0;
        ThePlayer.addPoints(5);
        ThePlayer.addPoints(5);
        ThePlayer.addPoints(5);
        score = ThePlayer.getScore();
        assertThat(score).isEqualTo( expected: 15);

    }
}
```

This code creates a player and sets the score value. After updating it it should return the correct score value.

Before:

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)

After:

Element	Class, %	Method, %	Line, %
nl	16% (9/55)	10% (32/312)	8% (98/1153)
tudelft	16% (9/55)	10% (32/312)	8% (98/1153)
jpacman	16% (9/55)	10% (32/312)	8% (98/1153)

Task 3:

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

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
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
getKiller()		100%		n/a	0	1	0	1	0	1
getScore()		100%		n/a	0	1	0	1	0	1
Total	10 of 70	85%	3 of 6	50%	3	11	3	24	0	8

jpacman

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
nl.tudelft.jpacman.level		67%		56%	74	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%	294 of 637	53%	293	590	228	1,039	50	268	6	47

AnimatedSprite

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
update()		74%		50%	3	5	3	11	0	1
split(int, int, int, int)		0%		n/a	1	1	2	2	1	1
AnimatedSprite(Sprite[], int, boolean, boolean)		87%		50%	2	3	0	9	0	1
currentSprite()		83%		50%	3	4	0	5	0	1
getWidth()		69%		50%	2	3	0	2	0	1
getHeight()		69%		50%	2	3	0	2	0	1

The results were somewhat different. Even though I did not check for classes, the overall number went down compared to what was already tested in base. Other than that, the line count number is the same.

- Did you find helpful the source code visualization from JaCoCo on uncovered branches?

Yes, this is much more organized and I can see exactly what was tested and what was not. It also reminds me of branches and classes that are not tested which is something that I did not think about. Such was when I was writing getSprite() and I forgot the branch for when the player was not alive, thus it would return deadSprite. I had forgotten to check for that, but I didn't realize until I used JaCoCo's report because the graphs next to each function is much easier to see.

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

I prefer JaCoCo's report. I personally like a mor visual report and it is hard to see what lines I am specifically missing, such as how I was missing the branch for getSprite().

Task 4:

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

This test is to test the representation of the account. First it creates an account and sets the name to be Foo. Thus the string representation of the account should return
'<Account %r>' % self.name
More accurately:
<Account 'Foo'>
Because Foo is the name.

```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"])
```

This test is to test if the account can be made into a dictionary using to_dict(). First, a random account is grabbed and filled in, then turned into a dictionary.
The account should now be populated with the base results of the dictionary, thus it checks if all the base values such as name and email are correct.

```python
def test_from_dict(self):
    """Test set attributes from dict"""
    data = ACCOUNT_DATA[self.rand]   # get a random account
    account = Account(**data)
    result = account.to_dict()
    account.from_dict(result)
    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"])
```

This test is very similar to the previous one. It will test if you can set the attributes of the dictionary. Thus we still have to check if all the attricubtes in result is equal to account.

```python
def test_update(self):
    """Test update attributes from dict"""
    id_number = self.rand
    data = ACCOUNT_DATA[id_number]   # get a random account
    account = Account(**data)
    account.name = "NewName"
    account.id = id_number
    account.update()
    self.assertEqual(account.name, second: "NewName")
```

This tests is to see if the update function is working. First it makes an account and changes the account name and sets the id number. Then it will update the account in the database. Thus we can check if the name changed in the database.

```
def test_update2(self):
    """Test update branch attributes from dict"""
    data = ACCOUNT_DATA[self.rand]   # get a random account
    account = Account(**data)
    account.name = "NewName"
    with self.assertRaises(DataValidationError):
        account.update()
    self.assertEqual(account.name, second: "NewName")
```

This test is similar to the previous one, only checking to see if the same applies to when an id is not given to the account when updating. It will catch the exception raised from not including an id.

```
def test_delete(self):
    """Test delete attributes from dict"""
    id_number = self.rand
    data = ACCOUNT_DATA[id_number]   # get a random account
    account = Account(**data)
    account.create()
    account.delete()
    self.assertIsNone(Account.find(id_number))
```

This test makes sure that an account is deleted from the database when running .delete(). First, it will grab an account from a random id number and then create it in the database. Then it will delete the account and try to search for the account using the id number, which should return none.

```python
def test_fid(self):
    """Test find attributes from dict"""
    id_number = self.rand
    data = ACCOUNT_DATA[id_number]   # get a random account
    account = Account(**data)
    account.create()
    second_account = account.find(id_number)
    self.assertEqual(account.name, second_account.name)
    self.assertEqual(account.id, second_account.id)
```

This test is to see if the correct account is found using .find(). First, a random id has an account created, then the account is created in the database. After we will make a second account and use .find() to assign it to the previous account. Thus when checking, their information should exactly match because they are the same account.

Task 5:

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

```
student@VirtualBox:~/tdd$ nosetests

Counter tests
- It should create a counter (FAILED)

======================================================================
FAIL: It should create a counter
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/student/tdd/tests/test_counter.py", line 30, in test_create_a_counter
    self.assertEqual(result.status_code, status.HTTP_201_CREATED)
AssertionError: 404 != 201

Name             Stmts   Miss  Cover   Missing
---------------------------------------------
src/counter.py       2      0   100%
src/status.py        6      0   100%
---------------------------------------------
TOTAL                8      0   100%
----------------------------------------------------------------------
```

Here I created the test to create a counter, but I didnt create a POST route, so the test failed because it returned not found.

```python
@app.route( rule: '/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
```

```python
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)
```

```
student@VirtualBox:~/tdd$ nosetests

Counter tests
- It should create a counter
- It should return an error for duplicates

Name              Stmts   Miss  Cover   Missing
-----------------------------------------------
src/counter.py       11      0   100%
src/status.py         6      0   100%
-----------------------------------------------
TOTAL                17      0   100%
----------------------------------------------------------------------
Ran 2 tests in 0.067s

OK
```

Here I created the POST route and the test succeeded to create a counter. It also returns a 409 error code for having duplicate counter names.

```python
def test_update_a_counter(self):
    """Should update counter"""
    client = app.test_client()
    result = client.post('/counters/live')
    self.assertEqual(result.status_code, status.HTTP_201_CREATED)

    baseline = result.get_json()['live']
    result = self.client.put('/counters/live')

    self.assertEqual(result.status_code, status.HTTP_200_OK)
    self.assertEqual(result.get_json()['live'], baseline + 1)
```

```
- It should return an error for duplicates
- Should update counter (FAILED)

======================================================================
FAIL: Should update counter
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/student/tdd/tests/test_counter.py", line 50, in test_update_a_counter
    self.assertEqual(baseline, status.HTTP_200_OK)
AssertionError: 0 != 200
-------------------- >> begin captured logging << --------------------
src.counter: INFO: Request to create counter: live
--------------------- >> end captured logging << ---------------------

Name              Stmts   Miss  Cover   Missing
-----------------------------------------------
src/counter.py       11      0   100%
src/status.py         6      0   100%
-----------------------------------------------
TOTAL                17      0   100%
----------------------------------------------------------------------
Ran 3 tests in 0.068s
```

Here I made a test to update the counter but didnt make a put route, so my error code didnt return correctly.

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

    if name in COUNTERS:
        COUNTERS[name] = COUNTERS[name] + 1
        return {name: COUNTERS[name]}, status.HTTP_200_OK
    return {name: COUNTERS[name]}, status.HTTP_404_NOT_FOUND
```

```
student@VirtualBox:~/tdd$ nosetests

Counter tests
- It should create a counter
- It should return an error for duplicates
- Should read a counter (ERROR)
- Should update counter (FAILED)

======================================================================
ERROR: Should read a counter
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/student/tdd/tests/test_counter.py", line 59, in test_read_a_counter
    baseline = result.get_json()['/counters/laugh']
KeyError: '/counters/laugh'
-------------------- >> begin captured logging << --------------------
src.counter: INFO: Request to create counter: laugh
--------------------- >> end captured logging << ---------------------

======================================================================
FAIL: Should update counter
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/student/tdd/tests/test_counter.py", line 51, in test_update_a_counter
    self.assertEqual(result, baseline + 1)
AssertionError: <WrapperTestResponse streamed [200 OK]> != 1
-------------------- >> begin captured logging << --------------------
```

Here is my PUT route for updating a counter which updates the counter if it does exits and returns not found if it does not. It however still failed the test because I had used the wrong call. It should have not included '/counters/'

My read a counter also had the same issue, as shown above. The code is below.

```python
@app.route( rule: '/counters/<name>', methods=['GET'])
def get_counter(name):
    """read a counter"""
    app.logger.info(f"Request to read counter: {name}")

    if name in COUNTERS:
        return {name: COUNTERS[name]}, status.HTTP_200_OK
```

```python
def test_read_a_counter(self):
    """Should read a counter"""
    client = app.test_client()
    result = client.post('/counters/laugh')
    self.assertEqual(result.status_code, status.HTTP_201_CREATED)

    baseline = result.get_json()['laugh']
    result = self.client.get('/counters/laugh')

    self.assertEqual(result.status_code, status.HTTP_200_OK)
```

After changing the issue, all my tests passed.

```
Counter tests
- It should create a counter
- It should return an error for duplicates
- Should read a counter (ERROR)
- Should update counter

======================================================================
ERROR: Should read a counter
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/student/tdd/tests/test_counter.py", line 59, in test_read_a_counter
    baseline = result.get_json()['/counters/laugh']
KeyError: '/counters/laugh'
-------------------- >> begin captured logging << --------------------
src.counter: INFO: Request to create counter: laugh
--------------------- >> end captured logging << ---------------------

Name              Stmts   Miss  Cover   Missing
-----------------------------------------------
src/counter.py       23      4    83%   33, 38-41
src/status.py         6      0   100%
-----------------------------------------------
TOTAL                29      4    86%
----------------------------------------------------------------------
Ran 4 tests in 0.067s
```

```
Counter tests
- It should create a counter
- It should return an error for duplicates
- Should read a counter
- Should update counter

Name              Stmts   Miss  Cover   Missing
-----------------------------------------------
src/counter.py       23      1    96%   33
src/status.py         6      0   100%
-----------------------------------------------
TOTAL                29      1    97%
----------------------------------------------------------------------
Ran 4 tests in 0.069s

OK
```