Michelle McGowan

CS 472

Dynamic Analysis

Repository Link: https://github.com/mitshelle/jpacman

Below is the test coverage before any tests were added. The coverage is very bad. There is barely any testing.

Coverage | Tests in 'jpacman.test'

Element	Class, %	Method, %	Line, %
nl.tudelft.jpacman	3% (2/55)	1% (5/312)	1% (14/1137)
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	0% (0/13)	0% (0/78)	0% (0/345)
npc	0% (0/10)	0% (0/47)	0% (0/237)
points	0% (0/2)	0% (0/7)	0% (0/19)
sprite	0% (0/6)	0% (0/45)	0% (0/119)
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)
PacmanConfigurationException	0% (0/1)	0% (0/2)	0% (0/4)

Task 2

By adding a test to see if a player is alive, the test coverage increased by 15% for the class.

Coverage | Tests in 'jpacman.test'

Element	Class, %	Method, %	Line, %
nl.tudelft.jpacman	14% (8/55)	9% (30/312)	8% (93/1151)
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)
npc	0% (0/10)	0% (0/47)	0% (0/237)
points	0% (0/2)	0% (0/7)	0% (0/19)
sprite	66% (4/6)	44% (20/45)	51% (66/128)
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)
PacmanConfigurationException	0% (0/1)	0% (0/2)	0% (0/4)

Task 2.1

I added test functions for the methods listed below:

level/Player/getKiller()

level/Level/getBoard()

level/Level/isInProgress()

1. *level/Player/getKiller()*

After I added a test for the getKiller() method, the test coverage for the class increased to 23%.

level	23% (3/13)	12% (10/78)	7% (28/358)
CollisionInteractionMap	0% (0/2)	0% (0/9)	0% (0/41)
CollisionMap	100% (0/0)	100% (0/0)	100% (0/0)
DefaultPlayerInteractionMap	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% (0/6)
Player	100% (1/1)	62% (5/8)	66% (16/24)
PlayerCollisions	100% (1/1)	28% (2/7)	25% (7/28)
PlayerFactory	100% (1/1)	100% (3/3)	100% (5/5)

```
/**
 * Test to see if getKiller has a killer
 */
Michelle McGowan
@Test
void testGetKiller() {
    Ghost ghost = ghostFactory.createClyde();
    collision.playerVersusGhost(player, ghost);
    assertThat(player.getKiller()).isEqualTo(ghost);
}
```

Created a ghost and called a function to get a collision with the player. Asserted that the killer was the ghost that I created.

2. *level/Level/isInProgress()*

After adding a test function for IsInProgress the coverage level for the class went up to 38%.

nl.tudelft.jpacman	30% (17/55)	18% (56/30...	16% (188/1...
board	40% (4/10)	13% (7/53)	11% (16/142)
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	38% (5/13)	29% (21/71)	22% (76/3...
npc	40% (4/10)	12% (6/47)	9% (23/243)
points	0% (0/2)	0% (0/7)	0% (0/19)
sprite	66% (4/6)	48% (22/45)	57% (73/12...
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)
PacmanConfigurationException	0% (0/1)	0% (0/2)	0% (0/4)

```java
public class LevelTest {

    2 usages
    private static final PacManSprites sprites = new PacManSprites();
    5 usages
    private static final GhostFactory ghostFactory = new GhostFactory(sprites);
    //private static final Ghost ghost = mock(Ghost.class);

    1 usage
    private static final PointCalculator point = mock(PointCalculator.class);
    1 usage
    public LevelFactory levelFactory = new LevelFactory(sprites, ghostFactory, point);
    6 usages
    static List<Ghost> ghostList = new ArrayList<>();
    2 usages
    static List<Square> startPositionsList = new ArrayList<>();
    3 usages
    private static final Board board = mock(Board.class);
    1 usage
    private static final Square startPos1 = mock(Square.class);
    1 usage
    private static final CollisionMap cmap = mock(CollisionMap.class);

    Michelle McGowan
    @Test
    void testisInProgress() {
        // get the level
        // pass in the board, ghost list, starting positions, collision map
        Level level = new Level(board, ghostList, Lists.newArrayList(startPos1), cmap);
        // start the level
        level.start();
        assertThat(level.isInProgress()).isEqualTo( expected: true);
    }
```

Created a new level by passing in a board, the ghosts, the starting position and collision map. Started the level and asserted that the level was in progress.

level	38% (5/13)	29% (21/71)	22% (76/3...
CollisionInteractionMap	0% (0/2)	0% (0/9)	0% (0/41)
CollisionMap	100% (0/0)	100% (0/0)	100% (0/0)
DefaultPlayerInteractionMap	0% (0/1)	0% (0/5)	0% (0/13)
Level	50% (1/2)	52% (9/17)	35% (40/1...
LevelFactory	0% (0/2)	0% (0/7)	0% (0/27)
LevelTest	100% (1/1)	100% (2/2)	100% (8/8)
MapParser	0% (0/1)	0% (0/10)	0% (0/71)
Pellet	0% (0/1)	0% (0/3)	0% (0/6)
Player	100% (1/1)	62% (5/8)	66% (16/24)
PlayerCollisions	100% (1/1)	28% (2/7)	25% (7/28)
PlayerFactory	100% (1/1)	100% (3/3)	100% (5/5)

3. *level/Level/getBoard()*

The picture below shows the code coverage after the getBoard() test was added.

```
Michelle McGowan
@Test
void getBoard() {
    // create ghosts for the game
    ghostList.add(ghostFactory.createClyde());
    ghostList.add(ghostFactory.createBlinky());
    ghostList.add(ghostFactory.createInky());
    ghostList.add(ghostFactory.createPinky());
    // starting position
    startPositionsList.add(board.squareAt( x: 0,  y: 0));
    // get the level
    Level level = levelFactory.createLevel(board, ghostList, startPositionsList);
    assertThat(level.getBoard()).isNotNull();
}
```

The function creates a list of ghosts and sets the starting position. Creates a new level and then asserts that there is a board and it is not null.

nl.tudelft.jpacman	40% (22/55)	21% (66/306)	19% (223/1...
board	40% (4/10)	13% (7/53)	11% (16/142)
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	53% (7/13)	31% (23/72)	29% (103/3...
npc	70% (7/10)	31% (15/47)	14% (35/243)
points	0% (0/2)	0% (0/7)	0% (0/19)
sprite	66% (4/6)	46% (21/45)	53% (69/128)
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)
PacmanConfigurationException	0% (0/1)	0% (0/2)	0% (0/4)

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?

- They are similar but each puts emphasis on something else. JaCoCo shows users what they missed, whereas IntelliJ shows more of how much is done. This is why reading the JaCoCo coverage seems worse than the IntelliJ coverage

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

- Visualizing what needs to be done is helpful. It shows how far you have left to go for each element instead of just how many are left like in Intellij.

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

- I liked the JaCoCo report better. Because IntelliJ's coverage window is in the IDE, there are a lot of things going on which can be overwhelming to decipher. JaCoCo has its own window and allows you to visualize what still needs to be done. It is cleaner and more organized.

jpacman

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
nl.tudelft.jpacman.level		67%		57%	73	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%	293 of 637	54%	292	590	228	1,039	50	268	6	47

Created with JaCoCo

Task 4

```
(venv) michellemcgowan@mitshelle test_coverage % nosetests

Test Account Model
- Test Account deletion in database
- Test Account update in database
- Test creating multiple Accounts
- Test Account creation using known data
- Test the attribute from dict
- Test the representation of an account
- Test account to dict

Name                 Stmts   Miss  Cover   Missing
----------------------------------------------------
models/__init__.py       7      0   100%
models/account.py       40      0   100%
----------------------------------------------------
TOTAL                   47      0   100%
----------------------------------------------------------------------
Ran 7 tests in 0.202s

OK
```

For lines 34-35, I added a test to make an account into a dictionary.

```
    def test_from_dict(self):
        """ Test the attribute from dict """
        data = ACCOUNT_DATA[self.rand] # get a random account
        account = Account(**data)
        accountDict = account.to_dict()
        newAcc = Account()
        newAcc.from_dict(accountDict)
        self.assertEqual(account.name, newAcc.name)
        self.assertEqual(account.email, newAcc.email)
        self.assertEqual(account.phone_number, newAcc.phone_number)
        self.assertEqual(account.disabled, newAcc.disabled)
        self.assertEqual(account.date_joined, newAcc.date_joined)
```

I added a test to try and update an account for lines 45-48. Lines 74-75 were tested by adding a case for trying to update an account that did not have any data.

```python
def test_account_update(self):
    """ Test Account update in database"""
    data = ACCOUNT_DATA[self.rand]
    account = Account(**data)
    account.create()
    #print(account.id)
    # update name and phone num
    account.name = "Michelle"
    account.email = "mcgowm1@unlv.nevada.edu"
    account.update()
    updateAcc = Account.find(account.id)
    self.assertEqual(updateAcc.name, "Michelle")
    self.assertEqual(updateAcc.email, "mcgowm1@unlv.nevada.edu")
    empty_id = Account()
    try:
        empty_id.update()
    except DataValidationError as e:
        errorMsg = e
    self.assertEqual(str(errorMsg), 'Update called with empty ID field')
```

Lines 52-54 were covered by adding a test to delete an account.

```python
def test_account_deletion(self):
    """ Test Account deletion in database"""
    data = ACCOUNT_DATA[self.rand] # get a random account
    account = Account(**data)
    account.create()
    account.delete()
    self.assertEqual(len(Account.all()), 0)
```

Task 5

```
(.venv) michellemcgowan@mitshelle tdd % nosetests

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

Name              Stmts   Miss  Cover   Missing
-----------------------------------------------
src/counter.py       25      0   100%
src/status.py         6      0   100%
-----------------------------------------------
TOTAL                31      0   100%
----------------------------------------------------------------------
Ran 4 tests in 0.067s

OK
```

To implement a test for updating the counter, I created a function test_update_a_counter and ran nosetests. However since there was no update counter function in counter.py, the nosetests were red. Below is my code snippet as well as the nosetests output.

```python
def test_update_a_counter(self):
    """It should return an error for updates"""
    # create counter
    client = app.test_client()
    result = client.post('/counters/updateCounter')
    # check return success code
    self.assertEqual(result.status_code, status.HTTP_201_CREATED)
    # check baseline
    baseReq = result.get_json()['updateCounter']
    self.assertEqual(baseReq, 0)
    # update counter
    update = client.put('/counters/updateCounter')
    # check greater than 0
    baseReq = update.get_json()['updateCounter']
    self.assertEqual(baseReq, 1)
    # check return success code
    self.assertEqual(update.status_code, status.HTTP_200_OK)
    # check if not exist yet
    update = client.put('/counters/updateCounter2')
    # check return success code
    self.assertEqual(update.status_code, status.HTTP_204_NO_CONTENT)
```

```
(.venv) michellemcgowan@mitshelle tdd % nosetests

Counter
- It should create a counter
- It should return an error for duplicates
- It should return an error for updates (FAILED)

======================================================================
FAIL: It should return an error for updates
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/michellemcgowan/code/tdd/tests/test_counter.py", line 57, in test_update_a_cou
    self.assertEqual(baseReq, 1)
AssertionError: 0 != 1
--------------------- >> begin captured logging << ---------------------
src.counter: INFO: Request to create counter: updateCounter
--------------------- >> 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

FAILED (failures=1)
```

This function creates an account and checks for the CREATED status. It then checks that the value of the counter is 0 at first. Next, it updates the counter and checks if the value is greater than 0 and if an OK status was returned. It also checks if the NO_CONTENT status is returned if it tries to update a counter that does not exist. In counter.py, I added a function called update_counter to find a counter in the list and if it was not there, it created it and returned a no content flag. Otherwise it updated the counter by 1. If everything went well, the function would then return an OK status.

```python
@app.route('/counters/<name>', methods=['PUT'])
def update_counter(name):
    """Updtae a counter"""
    app.logger.info(f"Request to update counter: {name}")
    global COUNTERS
    if name not in COUNTERS:
        COUNTERS[name] = 1
        return {"Message":f"Counter {name} has not been created"}, status.HTTP_204_NO_CONTENT
    else:
        COUNTERS[name] += 1

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

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

Name              Stmts   Miss  Cover   Missing
-----------------------------------------------
src/counter.py       19      0   100%
src/status.py         6      0   100%
-----------------------------------------------
TOTAL                25      0   100%
----------------------------------------------------------------------
Ran 3 tests in 0.066s

OK
```

The nosetest returned green after adding the update_counter function for updating an account. This function uses PUT to update the account.

Next, I added a test to try and read a counter value. This function makes an account and updates it. Then, it retrieves data and compares if an OK flag was returned. The nosetest was red since there was not a read function in counter.py.

```python
    def test_read_a_counter(self):
        """It should return an error for reading"""
        # create counter
        client = app.test_client()
        client.post('/counters/readCounter')
        # update counter
        client.put('/counters/readCounter')
        getResult = client.get('/counters/readCounter')
        # check return success code
        self.assertEqual(getResult.status_code, status.HTTP_200_OK)
        # does not exist check
        getResult = client.get('/counters/readCounter2')
        # check return success code
        self.assertEqual(getResult.status_code, status.HTTP_404_NOT_FOUND)
```

```
(.venv) michellemcgowan@mitshelle tdd % nosetests

Counter
- It should create a counter
- It should return an error for duplicates
- It should return an error for reading (FAILED)
- It should return an error for updates

======================================================================
FAIL: It should return an error for reading
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/michellemcgowan/code/tdd/tests/test_counter.py", line 74, in test_read_a_counter
    self.assertEqual(getResult.status_code, status.HTTP_200_OK)
AssertionError: 405 != 200
-------------------- >> begin captured logging << --------------------
src.counter: INFO: Request to create counter: readCounter
src.counter: INFO: Request to update counter: readCounter
--------------------- >> end captured logging << ---------------------

Name             Stmts   Miss  Cover   Missing
---------------------------------------------
src/counter.py      19      0   100%
src/status.py        6      0   100%
---------------------------------------------
TOTAL               25      0   100%
----------------------------------------------------------------------
Ran 4 tests in 0.070s

FAILED (failures=1)
```

After adding read_counter function to counter.py, the nosetests turned green. In this case, the function uses GET to retrieve information and returns NOT_FOUND status if the account does not exist. It returns an OK if it does exist.

```python
@app.route('/counters/<name>', methods=['GET'])
def read_counter(name):
    """Read a counter"""
    app.logger.info(f"Request to read counter: {name}")
    global COUNTERS
    # check if not already exist
    if name not in COUNTERS:
        return {"Message":f"Counter {name} has not been created"}, status.HTTP_404_NOT_FOUND
    else:
        return {name: COUNTERS[name]}, status.HTTP_200_OK
```

```
(.venv) michellemcgowan@mitshelle tdd % nosetests

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

Name             Stmts   Miss  Cover   Missing
-----------------------------------------------
src/counter.py      25      0   100%
src/status.py        6      0   100%
-----------------------------------------------
TOTAL               31      0   100%
----------------------------------------------------------------------
Ran 4 tests in 0.070s

OK
```