Keith Del Rosario
CS472 - Section 1001

CS472 Unit Testing Lab

Github Fork Repository Link: https://github.com/haiimkeith/munch

Task 1 - JPacman Test Coverage

Coverage | Tests in 'jpacman.test'

Element	Class, %	Method, %	Line, %
nl.tudelft.jpacman	3% (2/55)	1% (5/312)	1% (14/1137)
board	20% (2/1...	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)

Is the coverage good enough?

The coverage is not good enough, as we can see many of the tests have 0% coverage meaning that they have yet to be tested.

Task 2 - Increasing Coverage on JPacman PlayerTest

Coverage | Tests in 'jpacman.test'

Element	Clas...	Meth...	Line, %
nl.tudelft.jpacman	3% (2...	1% (5/3...	1% (14/...
board	20% (...	9% (5/...	9% (14/...
fuzzer	0% (0...	0% (0/6)	0% (0/3...
game	0% (0...	0% (0/1...	0% (0/3...
integration	0% (0...	0% (0/4)	0% (0/6)
level	0% (0...	0% (0/...	0% (0/3...
npc	0% (0...	0% (0/...	0% (0/2...
points	0% (0...	0% (0/7)	0% (0/19)
sprite	0% (0...	0% (0/...	0% (0/1...
ui	0% (0...	0% (0/...	0% (0/1...
Launcher	0% (0...	0% (0/...	0% (0/41)
LauncherSmokeTest	0% (0...	0% (0/4)	0% (0/2...
PacmanConfigurationE	0% (0...	0% (0/2)	0% (0/4)

Keith Del Rosario
CS472 - Section 1001

CS472 Unit Testing Lab

Task 2.1 - Three sample tests and their test coverage improvements afterwards

AddPointsTest

```java
package nl.tudelft.jpacman.level;
import ...
/**
 * AddPointsTest
 * @author Keith Del Rosario
 */
public class AddPointsTest {
    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();


    @Test
    public void testAddPoints() {

        int addedScore = 10;
        int initialScore = ThePlayer.getScore();
        ThePlayer.addPoints(addedScore);
        int expectedScore = addedScore + initialScore;
        int actualScore = ThePlayer.getScore();
        assertThat(actualScore).isEqualTo(expectedScore);
    }
}
```

Coverage | Tests in 'jpacman.test'

Element	Class, %	Method, %	Line, %
nl.tudelft.jpacman	14% (8/5...	9% (31/312)	8% (95/1151)
board	20% (2/1...	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)	7% (6/78)	4% (15/350)
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% (66/12...
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)

Keith Del Rosario
CS472 - Section 1001

CS472 Unit Testing Lab

StartStopTest

```java
//Fix/Rename to start/stop test
public class StartStopTest {
    4 usages
    private Level level;
    1 usage
    private final Square square = mock(Square.class);
    1 usage
    private final Board board = mock(Board.class);
    1 usage
    private final CollisionMap collisions = mock(CollisionMap.class);
    1 usage
    private static final PacManSprites SPRITE_STORE = new PacManSprites();
    4 usages
    private GhostFactory ghostFactory = new GhostFactory(SPRITE_STORE);
    1 usage
    private Ghost Blinky = ghostFactory.createBlinky();
    1 usage
    private Ghost Pinky = ghostFactory.createPinky();
    1 usage
    private Ghost Inky = ghostFactory.createInky();
    1 usage
    private Ghost Clyde = ghostFactory.createClyde();

    @Test
    void testStartStop() {
        level = new Level(board, Lists.newArrayList(Blinky, Pinky, Inky, Clyde), Lists.newArrayList(square), collisions);
        level.start();
        level.stop();
        assertThat(level.isInProgress()).isEqualTo( expected: false);
    }
}
```

Coverage | Tests in 'jpacman.test'

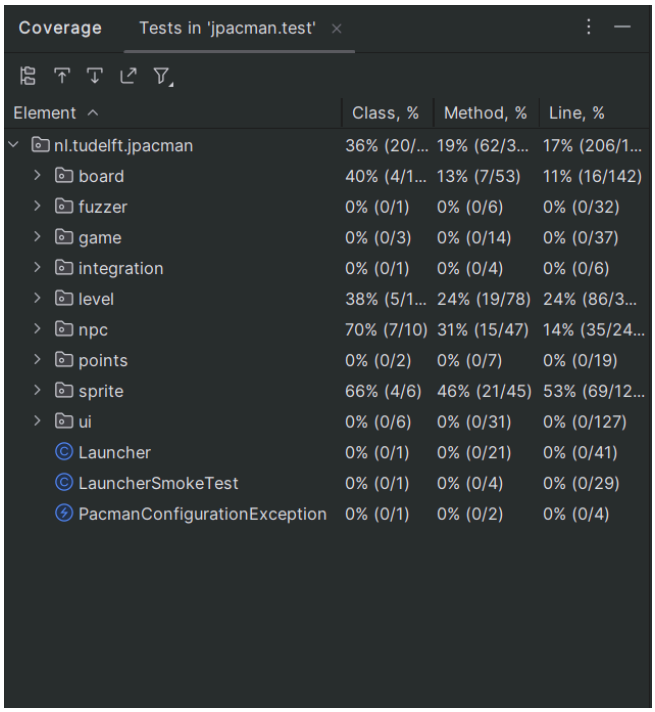
Element	Class, %	Method, %	Line, %
nl.tudelft.jpacman	21% (12/...	14% (44/3...	13% (153/1...
board	40% (4/1...	13% (7/53)	11% (16/142)
fuzzer	0% (0/1)	0% (0/6)	0% (0/32)
game	0% (0/3)	0% (0/14)	0% (0/37)
integration	0% (0/1)	0% (0/4)	0% (0/6)
level	30% (4/1...	21% (17/78)	20% (71/35...
npc	0% (0/10)	0% (0/47)	0% (0/237)
points	0% (0/2)	0% (0/7)	0% (0/19)
sprite	66% (4/6)	44% (20/45)	51% (66/12...
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)

registerPlayerTest

```java
 * @author Keith Del Rosario
 */
public class registerPlayerTest {
    2 usages
    private Level level;
    3 usages
    private Square square = mock(Square.class);
    1 usage
    private final Board board = mock(Board.class);
    1 usage
    private final CollisionMap collisions = mock(CollisionMap.class);
    1 usage
    private static final PacManSprites SPRITE_STORE = new PacManSprites();
    4 usages
    private GhostFactory ghostFactory = new GhostFactory(SPRITE_STORE);
    1 usage
    private Ghost Blinky = ghostFactory.createBlinky();
    1 usage
    private Ghost Pinky = ghostFactory.createPinky();
    1 usage
    private Ghost Inky = ghostFactory.createInky();
    1 usage
    private Ghost Clyde = ghostFactory.createClyde();
    @Test
    void testRegister(){
        level = new Level(board, Lists.newArrayList(Blinky,Pinky,Inky,Clyde), Lists.newArrayList(square), collisions);
        Player p = mock(Player.class);
        level.registerPlayer(p);
        p.occupy(square);
        assertThat(square).isNotNull();
        assertThat(p).isNotNull();
    }
}
```

Keith Del Rosario
CS472 - Section 1001

CS472 Unit Testing Lab



Coverage | Tests in 'jpacman.test'

Element	Class, %	Method, %	Line, %
nl.tudelft.jpacman	36% (20/...	19% (62/3...	17% (206/1...
board	40% (4/1...	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% (5/1...	24% (19/78)	24% (86/3...
npc	70% (7/10)	31% (15/47)	14% (35/24...
points	0% (0/2)	0% (0/7)	0% (0/19)
sprite	66% (4/6)	46% (21/45)	53% (69/12...
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)

Task 3 - JaCoCo Report on Pacman

jpacman

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
nl.tudelft.jpacman.level		67%		58%	73	155	103	344	21	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		88%		62%	29	70	10	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	1,204 of 4,694	74%	290 of 637	54%	291	590	227	1,039	51	268	6	47

- ***Are the coverage results from JaCoCo similar to the ones you got from IntelliJ in the last task? Why so or why not?***
    - ***The coverage results from JaCoCo aren't exactly the same as my coverage results in IntelliJ in the last task, this is since the JaCoCo tasks are more likely to support the testing of many more methods in comparison to my tests where I only have three method tests. Adding more method tests in each section would support getting more coverage and get closer to having similar results to JaCoCo***
- ***Did you find the source code visualization helpful from JaCoCo on uncovered branches?***
    - ***I believe the source code visualization on JaCoCo to be somewhat helpful since it helps see how much coverage a certain branch contains and can make readability easier in that sense by rather than looking at percentages***

and seeing which branch is missing which methods, to get a general estimate of how much testing coverage is done in each branch instead.

- ***Which visualization did you prefer and why? IntelliJ's coverage window or JaCoCo's report?***
  - ***I prefer the JaCoCo's report visualization more partially because it represents the missed and uncovered branches better and provides more information overall in comparison to the IntelliJ coverage window. Although, there is quality of life in the IntelliJ coverage window just by simply having it inside the IntelliJ IDE compared to having it inside an HTML link.***

Task 4 - Working with Python Test Coverage

```
PS C:\Users\haiim\VSCode\CS472\test_coverage> nosetests
Test creating multiple Accounts ... ok
Test Account creation using known data ... ok
test_delete (test_account.TestAccountModel) ... ok
test_find (test_account.TestAccountModel) ... ok
Test account to dict ... ok
Test the representation of an account ... ok
Test account to dict ... ok
test_update (test_account.TestAccountModel) ... ok

Name                    Stmts   Miss  Cover   Missing
-------------------------------------------------
models\__init__.py          7      0   100%
models\account.py          40      0   100%
-------------------------------------------------
TOTAL                      47      0   100%
----------------------------------------------------------------------
Ran 8 tests in 1.658s

OK
```

Code Snippets of All Tests

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

    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"])

    def test_from_dict(self):
        """ Test account to dict """
        data = ACCOUNT_DATA[self.rand] # get a random account
        account = Account(**data)
        result = account.to_dict()
        result2 = Account(**data)
        result2.from_dict(result)
        self.assertEqual(account.name, result2.name)
        self.assertEqual(account.email, result2.email)
        self.assertEqual(account.phone_number, result2.phone_number)
        self.assertEqual(account.disabled, result2.disabled)
        self.assertEqual(account.date_joined, result2.date_joined)
```

Keith Del Rosario
CS472 - Section 1001

CS472 Unit Testing Lab

```python
    def test_update(self):
        data = ACCOUNT_DATA[self.rand] # get a random account
        account = Account(**data)
        account.create()

        account2 = Account(**data)
        account.name = "Testing Changed Name"
        account.update()

        with self.assertRaises(DataValidationError):
            account2.update()

        self.assertNotEqual(account.name, account2.name)

    def test_delete(self):
        data = ACCOUNT_DATA[self.rand] # get a random account
        account = Account(**data)
        deleting_id = account.id
        account.create()
        account.delete()
        self.assertEqual(deleting_id, None)

    def test_find(self):
        data = ACCOUNT_DATA[self.rand]
        account = Account(**data)
        account.create()

        test_account = Account.find(account.id)

        self.assertEqual(test_account, account)
```

Task 5 - TDD

```
PS C:\Users\haiim\VSCode\CS472\tdd> nosetests

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

Name              Stmts   Miss  Cover   Missing
-----------------------------------------------
src\counter.py       20      0   100%
src\status.py         6      0   100%
-----------------------------------------------
TOTAL                26      0   100%
----------------------------------------------------------------------
Ran 4 tests in 0.453s

OK
```

Keith Del Rosario
CS472 - Section 1001

CS472 Unit Testing Lab

Code Snippet of both files

test_counter.py

```python
from unittest import TestCase

# we need to import the unit under test - counter
from src.counter import app, COUNTERS

# we need to import the file that contains the status codes
from src import status


class CounterTest(TestCase):
  """Counter tests"""

  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)


  def setUp(self):
    self.client = app.test_client()

  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):
    result = self.client.post('/counters/abc')
    self.assertEqual(result.status_code, status.HTTP_201_CREATED)
    self.assertEqual(result.json['abc'], 0)

    result = self.client.put('/counters/abc')
    self.assertEqual(result.status_code, status.HTTP_200_OK)
    self.assertEqual(result.json['abc'], 1)

  def test_get_a_counter(self):
    result = self.client.post('/counters/kdr')
    self.assertEqual(result.status_code, status.HTTP_201_CREATED)

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

    result = self.client.get('/counters/doesntexist')
    self.assertEqual(result.status_code, status.HTTP_409_CONFLICT)
```

counter.py

```python
@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):
  COUNTERS[name] += 1
  return {name: COUNTERS[name]}, status.HTTP_200_OK

@app.route('/counters/<name>', methods=['GET'])
def get_counter(name):
  if name in COUNTERS:
    return {name: COUNTERS[name]}, status.HTTP_200_OK
  return {"Message": f"Counter {name} does not exist"}, status.HTTP_409_CONFLICT
```