Dynamic Analysis Lab

Kyle Meyer

CS472-1001

September 14, 2024

https://github.com/caseycodes32/CS472_LABS

Task 1 – JPacman Test Coverage

Coverage | Tests in 'jpacman.test'

Element	Class, %	Method, %	Line, %	Branch, %
nl.tudelft.jpacman	3% (2/55)	1% (5/312)	1% (14/1127)	0% (1/521)
board	20% (2/10)	9% (5/53)	9% (14/141)	1% (1/96)
fuzzer	0% (0/1)	0% (0/6)	0% (0/32)	0% (0/8)
game	0% (0/3)	0% (0/14)	0% (0/37)	0% (0/14)
integration	0% (0/1)	0% (0/4)	0% (0/6)	100% (0/0)
level	0% (0/13)	0% (0/78)	0% (0/343)	0% (0/167)
npc	0% (0/10)	0% (0/47)	0% (0/233)	0% (0/116)
points	0% (0/2)	0% (0/7)	0% (0/19)	0% (0/4)
sprite	0% (0/6)	0% (0/45)	0% (0/119)	0% (0/48)
ui	0% (0/6)	0% (0/31)	0% (0/123)	0% (0/60)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)	0% (0/6)
LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)	0% (0/2)
PacmanConfigurationExcept	0% (0/1)	0% (0/2)	0% (0/4)	100% (0/0)

Question: Is the test coverage good enough?

No, the test coverage is not good enough. Most of the code has no test coverage at all. The average amount of coverage is less than 10%. For the test coverage to be good, the amount should exceed 80%.

Task 2 – Increasing Coverage on JPacman

Coverage | Tests in 'jpacman.test'

Element	Class, %	Method, %	Line, %	Branch, %
nl.tudelft.jpacman	14% (8/55)	9% (30/312)	8% (93/1141)	4% (22/539)
board	20% (2/10)	9% (5/53)	9% (14/141)	1% (1/96)
fuzzer	0% (0/1)	0% (0/6)	0% (0/32)	0% (0/8)
game	0% (0/3)	0% (0/14)	0% (0/37)	0% (0/14)
integration	0% (0/1)	0% (0/4)	0% (0/6)	100% (0/0)
level	15% (2/13)	6% (5/78)	3% (13/348)	0% (0/167)
npc	0% (0/10)	0% (0/47)	0% (0/233)	0% (0/116)
points	0% (0/2)	0% (0/7)	0% (0/19)	0% (0/4)
sprite	66% (4/6)	44% (20/45)	51% (66/128)	31% (21/66)
ui	0% (0/6)	0% (0/31)	0% (0/123)	0% (0/60)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)	0% (0/6)
LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)	0% (0/2)
PacmanConfigurationExcepti	0% (0/1)	0% (0/2)	0% (0/4)	100% (0/0)

<- Note that after the suggested test implementations, the test coverage of all lines jumped from 0% to 6%.

Task 2.1 – Adding Unit Tests for JPacman

The following are the three methods that I chose to expand test coverage of:

1	src/main/java/nl.tudelft.jpacman/level/Player/Player.addPoints
2	src/main/java/nl.tudelft.jpacman/level/PlayerCollisions/PlayerCollisions.playerVersusGhost
3	src/main/java/nl.tudelft.jpacman/level/PlayerCollisions/PlayerCollisions.playerVersusPellet

The following screenshots show my test code (left) and test coverage analysis (right), in order, for the above methods.

```java
@Test  new *
void testAddPoints(){
    //set an integer to add to the player's running point value
    int new_points = 13;
    ThePlayer.addPoints(new_points);
    //ensure the getScore function returns the correct vale
    assertThat(ThePlayer.getScore()).isEqualTo(new_points);

    //add more points, and confirm that points do not
    // reset but instead increase by new value
    int more_points = 7;
    ThePlayer.addPoints(new_points);
    assertThat(ThePlayer.getScore()).isEqualTo( expected: new_points
        + more_points);
}
```

Coverage Tests in 'jpacman.test'

Element	Class, %	Method, %	Line, %	Branch, %
nl.tudelft.jpacman	14% (8/55)	10% (32/312)	8% (96/1141)	4% (23/539)
board	20% (2/10)	9% (5/53)	9% (14/141)	1% (1/96)
fuzzer	0% (0/1)	0% (0/6)	0% (0/32)	0% (0/8)
game	0% (0/3)	0% (0/14)	0% (0/37)	0% (0/14)
integration	0% (0/1)	0% (0/4)	0% (0/6)	100% (0/0)
level	15% (2/13)	8% (7/78)	4% (16/348)	0% (0/167)
npc	0% (0/10)	0% (0/47)	0% (0/233)	0% (0/116)
points	0% (0/2)	0% (0/7)	0% (0/19)	0% (0/4)
sprite	66% (4/6)	44% (20/45)	51% (66/128)	33% (22/66)
ui	0% (0/6)	0% (0/31)	0% (0/123)	0% (0/60)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)	0% (0/6)
LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)	0% (0/2)
PacmanConfigurationExcepti	0% (0/1)	0% (0/2)	0% (0/4)	100% (0/0)

```java
@Test  new *
void testPlayerVSGhost() {
    Player ThePlayer = PFactory.createPacMan();
    Ghost Blinky = GFactory.createBlinky();
    ThePlayer.setAlive(true);
    PointCalculator PointCalc = new DefaultPointCalculator();
    PlayerCollisions PlayerC = new PlayerCollisions(PointCalc);
    //simulate a player collision with a ghost, which
    //  will set their alive status to false
    PlayerC.playerVersusGhost(ThePlayer, Blinky);
    assertThat(ThePlayer.isAlive()).isEqualTo( expected: false);
}
```

Coverage Tests in 'jpacman.test'

Element	Class, %	Method, %	Line, %	Branch, %
nl.tudelft.jpacman	25% (14/55)	14% (45/312)	11% (137/1156)	5% (30/563)
board	20% (2/10)	9% (5/53)	9% (14/141)	1% (1/96)
fuzzer	0% (0/1)	0% (0/6)	0% (0/32)	0% (0/8)
game	0% (0/3)	0% (0/14)	0% (0/37)	0% (0/14)
integration	0% (0/1)	0% (0/4)	0% (0/6)	100% (0/0)
level	23% (3/13)	14% (11/78)	9% (32/355)	2% (4/167)
npc	40% (4/10)	12% (6/47)	7% (17/240)	0% (1/140)
points	50% (1/2)	14% (1/7)	5% (1/20)	0% (0/4)
sprite	66% (4/6)	48% (22/45)	57% (73/128)	36% (24/66)
ui	0% (0/6)	0% (0/31)	0% (0/123)	0% (0/60)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)	0% (0/6)
LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)	0% (0/2)
PacmanConfigurationExcepti	0% (0/1)	0% (0/2)	0% (0/4)	100% (0/0)

```java
@Test  new *
void testPlayerVSPellet() {
    Player ThePlayer = PFactory.createPacMan();
    int score = ThePlayer.getScore();
    PointCalculator PointCalc = new DefaultPointCalculator();
    PlayerCollisions PlayerC = new PlayerCollisions(PointCalc);
    //simulate a player collision with a pellet worth 5 points
    Pellet P = new Pellet( points: 5, SPRITE_STORE.getPelletSprite());
    PlayerC.playerVersusPellet(ThePlayer, P);
    //ensure that the player's points have increased by 5 from
    //  before the collision
    assertThat(ThePlayer.getScore()).isEqualTo( expected: score + 5);
}
```

Coverage Tests in 'jpacman.test'

Element	Class, %	Method, %	Line, %	Branch, %
nl.tudelft.jpacman	27% (15/55)	16% (52/312)	13% (152/11...	6% (34/563)
board	20% (2/10)	13% (7/53)	12% (18/141)	5% (5/96)
fuzzer	0% (0/1)	0% (0/6)	0% (0/32)	0% (0/8)
game	0% (0/3)	0% (0/14)	0% (0/37)	0% (0/14)
integration	0% (0/1)	0% (0/4)	0% (0/6)	100% (0/0)
level	30% (4/13)	17% (14/78)	11% (40/356)	2% (4/167)
npc	40% (4/10)	12% (6/47)	7% (17/240)	0% (1/140)
points	50% (1/2)	28% (2/7)	15% (3/20)	0% (0/4)
sprite	66% (4/6)	51% (23/45)	57% (74/128)	36% (24/66)
ui	0% (0/6)	0% (0/31)	0% (0/123)	0% (0/60)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)	0% (0/6)
LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)	0% (0/2)
PacmanConfigurationExcepti	0% (0/1)	0% (0/2)	0% (0/4)	100% (0/0)

Task 3 – JaCoCo Report on JPacman

JaCoCo Test Coverage Report:

jpacman

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
nl.tudelft.jpacman.level		68%		58%	73	155	102	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%		59%	43	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	1,206 of 4,694	74%	291 of 637	54%	291	590	227	1,039	51	268	6	47

Question: Are the coverage results from JaCoCo similar to the ones you got from IntelliJ?

No, JaCoCo shows a much higher percentage of coverage than IntelliJ. This is likely because JaCoCo has a different process of determining whether code has been tested, or requires testing at all.

Question: Did you find helpful the source code visualization from JaCoCo?

Yes, JaCoCo's visualization was extremely helpful for discovering uncovered code branches. It is very easy to click the directories, files, and classes to see which code needs more coverage, as it is indicated by a red and green bar diagram.

Question: Which visualization did you prefer and why?

I much preferred the JaCoCo report, because the visual and interactive nature of it was extremely easy to use for analysis of code test coverage. It was very simple to see a mostly-red bar, click the file, and realize which methods need expanded test coverage.

Task 4 – Working with Python Test Coverage

To the right is the initial test coverage report, before any tests have been added:

```
amethyst@amethyst:~/CS472/2/test_coverage$ pytest
============================= test session starts ==============================
platform linux -- Python 3.8.10, pytest-8.3.3, pluggy-1.5.0 -- /usr/bin/python3
cachedir: .pytest_cache
rootdir: /home/amethyst/CS472/2/test_coverage
configfile: pytest.ini
plugins: cov-5.0.0
collected 2 items

tests/test_account.py::test_create_all_accounts PASSED                   [ 50%]
tests/test_account.py::test_create_an_account PASSED                     [100%]

---------- coverage: platform linux, python 3.8.10-final-0 -----------
Name                  Stmts   Miss  Cover   Missing
-----------------------------------------------------
models/__init__.py        7      0   100%
models/account.py        40     13    68%   26, 30, 34-35, 45-48, 52-54, 74-75
-----------------------------------------------------
TOTAL                    47     13    72%
```

After completing the remaining tests, I was able to expand the test coverage to 100%. There is a warning displayed, but it has to do with deprecation of a function used in the original code and does not pose a real issue.

```
amethyst@amethyst:~/CS472/2/test_coverage$ pytest
============================= test session starts ==============================
platform linux -- Python 3.8.10, pytest-8.3.3, pluggy-1.5.0 -- /usr/bin/python3
cachedir: .pytest_cache
rootdir: /home/amethyst/CS472/2/test_coverage
configfile: pytest.ini
plugins: cov-5.0.0
collected 8 items

tests/test_account.py::test_create_all_accounts PASSED                   [ 12%]
tests/test_account.py::test_create_an_account PASSED                     [ 25%]
tests/test_account.py::test_repr PASSED                                  [ 37%]
tests/test_account.py::test_to_dict PASSED                               [ 50%]
tests/test_account.py::test_from_dict PASSED                             [ 62%]
tests/test_account.py::test_update PASSED                                [ 75%]
tests/test_account.py::test_delete PASSED                                [ 87%]
tests/test_account.py::test_find PASSED                                  [100%]

=========================== warnings summary ===============================
tests/test_account.py::test_update
tests/test_account.py::test_delete
tests/test_account.py::test_find
  /home/amethyst/CS472/2/test_coverage/models/account.py:75: LegacyAPIWarning: The Query.get() meth
od is considered legacy as of the 1.x series of SQLAlchemy and becomes a legacy construct in 2.0. T
he method is now available as Session.get() (deprecated since: 2.0) (Background on SQLAlchemy 2.0 a
t: https://sqlalche.me/e/b8d9)
    return cls.query.get(account_id)

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html

---------- coverage: platform linux, python 3.8.10-final-0 -----------
Name                  Stmts   Miss  Cover   Missing
-----------------------------------------------------
models/__init__.py        7      0   100%
models/account.py        40      0   100%
-----------------------------------------------------
TOTAL                    47      0   100%

================================ 8 passed, 3 warnings in 0.70s =================
```

The code that I wrote to achieve the full test coverage is provided on the next page.

```python
def test_to_dict():
    """Test the account to dict conversion function"""
    rand = randrange(0, len(ACCOUNT_DATA))
    data = ACCOUNT_DATA[rand]  # get a random account
    account = Account(**data)
    account_dict = account.to_dict()
    assert account.id == account_dict["id"]
    assert str(account.name) == account_dict["name"]
    assert str(account.email) == account_dict["email"]
    assert str(account.phone_number) == account_dict["phone_number"]
    assert bool(account.disabled) == account_dict["disabled"]
    assert account.date_joined == account_dict["date_joined"]

def test_from_dict():
    """Test the dictionary to account data function"""
    rand = randrange(0, len(ACCOUNT_DATA))
    data = ACCOUNT_DATA[rand]  # get a random account
    account = Account()
    account.from_dict(data)
    assert account.id == data.get("id")
    assert str(account.name) == data.get("name")
    assert str(account.email) == data.get("email")
    assert str(account.phone_number) == data.get("phone_number")
    assert bool(account.disabled) == data.get("disabled")
    assert account.date_joined == data.get("date_joined")

def test_update():
    """ Test Account update using known data """
    rand = randrange(0, len(ACCOUNT_DATA))
    data = ACCOUNT_DATA[rand]  # get a random account
    account = Account(**data)
    account.create()

    account.name = "HelloWorld123"
    account.update()
    account_from_db = Account.find(account.id)
    assert str(account_from_db.name) == str(account.name)

    #empty id field to test branch condition where id = 0
    account.id = 0
    try:
        account.update()
    except:
        assert "Data Validation Error"

def test_delete():
    """ Test Account delete using known data """
    rand = randrange(0, len(ACCOUNT_DATA))
    data = ACCOUNT_DATA[rand]  # get a random account
    account = Account(**data)
    account.create()
    account.delete()

    account2 = Account.find(account.id)
    assert account != account2

def test_find():
    """ Test Account find """
    rand = randrange(0, len(ACCOUNT_DATA))
    data = ACCOUNT_DATA[rand]  # get a random account
    account = Account(**data)
    account.create()

    account2 = Account.find(account.id)
    assert str(account.name) == str(account2.name)
    assert str(account.email) == str(account2.email)
    assert str(account.phone_number) == str(account2.phone_number)
    assert bool(account.disabled) == bool(account2.disabled)
    assert account.date_joined == account2.date_joined
```

Task 5 – Test driven development (TDD)

The code to the right is the update_a_counter() method I made in test_counter.py. I started with this, checking for the correct HTTP values being returned. I did have to complete it after completing update_counter, so that I could check the result data and correctly compare it.

```
def test_update_a_counter(self, client):
    #create a couter and check that the result is 201
    result = client.post('/counters/xyz')
    assert result.status_code == status.HTTP_201_CREATED
    #increment a counter, and check that the result is OK
    #and contains 1
    result = client.put('/counters/xyz')
    assert result.status_code == status.HTTP_200_OK
    assert result.data == b'{"xyz":1}\n'
    #attempt to increment an invalid counter, check tfor 404
    result = client.put('/counters/not_xyz')
    assert result.status_code == status.HTTP_404_NOT_FOUND
    #increment the xyz counter again, and check result 2
    result = client.put('/counters/xyz')
    assert result.status_code == status.HTTP_200_OK
    assert result.data == b'{"xyz":2}\n'
```

This is the function update_counter in counter.py. This responds to client requests.

```
@app.route('/counters/<name>', methods=['PUT'])
def update_counter(name):
    """Update a counter"""
    app.logger.info(f"Request to update a counter: {name}")
    global COUNTERS
    #if given name exists in counters, increment counter and respond OK
    if name in COUNTERS:
        COUNTERS[name] += 1
        return {name: COUNTERS[name]}, status.HTTP_200_OK
    else:
        return {"Message":f"Counter {name} not found"}, status.HTTP_404_NOT_FOUND
```

Finally, the following is the pytest output showing 100% coverage.

```
amethyst@amethyst:~/CS472/2/tdd$ pytest
============================= test session starts ==============================
platform linux -- Python 3.8.10, pytest-8.3.3, pluggy-1.5.0 -- /usr/bin/python3
cachedir: .pytest_cache
rootdir: /home/amethyst/CS472/2/tdd
configfile: pytest.ini
plugins: cov-5.0.0
collected 3 items

tests/test_counter.py::TestCounterEndPoints::test_create_a_counter PASSED [ 33%]
tests/test_counter.py::TestCounterEndPoints::test_duplicate_a_counter PASSED [ 66%]
tests/test_counter.py::TestCounterEndPoints::test_update_a_counter PASSED [100%]

---------- coverage: platform linux, python 3.8.10-final-0 -----------
Name              Stmts   Miss  Cover   Missing
-----------------------------------------------
src/__init__.py       0      0   100%
src/counter.py       18      0   100%
src/status.py         6      0   100%
-----------------------------------------------
TOTAL                24      0   100%


============================== 3 passed in 0.20s ===============================
```