Software Testing Lab Report

Alexander Schupp

My Forked Repo: https://github.com/MoldyPotat/UNLV-S24-CS472-Group7

Task 1

Test jpacman before any changes

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)

Question: Is the coverage good enough?
Answer: The coverage on this code is terrible very little of the code has any tests on it and needs to be drastically improved.

Task 2

Implemented test for Player.isAlive() in level

```java
public class PlayerTest {
    /**
     * I prefer to save the instances for this test in particular
     * because it is really a pain to instantiate Player, and I
     * will want to test other methods of Player in here.
     */
    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
    void testAlive() { assertThat(ThePlayer.isAlive()).isEqualTo( expected: true); }
    @Test
    void testAddPoints() {
        ThePlayer.addPoints(10);
        assertThat(ThePlayer.getScore()).isEqualTo( expected: 10);
    }
}
```

Test coverage after adding player test

Element	Clas...	Metho...	Line, %
nl.tudelft.jpacman	14% (8...	9% (30/...	8% (93/1...
board	20% (...	9% (5/53)	9% (14/1...
fuzzer	0% (0/1)	0% (0/6)	0% (0/32)
game	0% (0/...	0% (0/14)	0% (0/37)
integration	0% (0/1)	0% (0/4)	0% (0/6)
level	15% (2...	6% (5/78)	3% (13/3...
npc	0% (0/...	0% (0/47)	0% (0/23...
points	0% (0/...	0% (0/7)	0% (0/19)
sprite	66% (...	44% (20...	51% (66/...
ui	0% (0/...	0% (0/31)	0% (0/127)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)
LauncherSmokeTe	0% (0/1)	0% (0/4)	0% (0/29)
PacmanConfigura	0% (0/1)	0% (0/2)	0% (0/4)

Task 2.1

Wrote 3 tests

1. Test for createPinky()

```
public class GhostTest {
    2 usages
    private GhostFactory ghostFactory;
    @BeforeEach
    void setUp() { ghostFactory = new GhostFactory(new PacManSprites()); }

    @Test
    void testCreatePinky() {
        Ghost pinky = ghostFactory.createPinky();
        assertThat(pinky).isNotNull();
    }
}
```

Coverage after test createPinky()

Element	Class, %	Method, %	Line, %
nl.tudelft.jpacman	21% (12/55)	12% (39/312)	10% (116/11...
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)	8% (7/78)	4% (16/350)
npc	40% (4/10)	12% (6/47)	6% (17/243)
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)

2. Test for consumedPellet()

```
@Test
void testConsumedPellet(){
    pointCalculator.consumedAPellet(ThePlayer, pellet);
    assertThat(ThePlayer.getScore()).isEqualTo( expected: 10);
}
```

Coverage after test consumedPellet()

Element	Class, %	Method, %	Line, %
nl.tudelft.jpacman	25% (14/55)	13% (43/312)	10% (124/11...
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	23% (3/13)	11% (9/78)	5% (21/351)
npc	40% (4/10)	12% (6/47)	6% (17/243)
points	50% (1/2)	14% (1/7)	10% (2/20)
sprite	66% (4/6)	48% (22/45)	54% (70/128)
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)

3. Test for collidedWithAGhost()

```
@Test
void testcollidedWithAGhost(){
    Ghost pinky = ghostFactory.createPinky();
    pointCalculator.collidedWithAGhost(ThePlayer, pinky);
    assertThat(ThePlayer.getScore()).isEqualTo( expected: 0);
}
```

Coverage after test collidedWithAGhost

Element	Class, %	Method, %	Line, %
nl.tudelft.jpacman	25% (14/55)	14% (44/312)	10% (125/11...
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	23% (3/13)	11% (9/78)	5% (21/351)
npc	40% (4/10)	12% (6/47)	6% (17/243)
points	50% (1/2)	28% (2/7)	15% (3/20)
sprite	66% (4/6)	48% (22/45)	54% (70/128)
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

JaCoCo overview Report

jpacman

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
nl.tudelft.jpacman.level		67%		57%	74	155	104	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		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	1,213 of 4,694	74%	293 of 637	54%	293	590	229	1,039	51	268	6	47

JaCoCo Player Report

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

Questions:

- Are the coverage results from JaCoCo similar to the ones you got from IntelliJ in the last task? Why so or why not?
    - No, the coverage results for the JaCoCo report show much higher coverage than from the inbuilt coverage report. This is likely due to the way each coverage report generator declares what needs testing and what is covered by each test.
- Did you find helpful the source code visualization from JaCoCo on uncovered branches?
    - The visualization of the source code given by JaCoCo is a very useful feature for figuring out what exactly needs to be tested at a glance compared to the inbuilt report that only says the files.
- Which visualization did you prefer and why? IntelliJ's coverage window or JaCoCo's report?
    - I prefer the JaCoCo report for the visualization aspect but overall the Intellij's coverage window was much faster and easier to use as it could all be done within the same window.

Task 4

Code to get coverage of accounts.py to 100%

Lines 34-35 test

```
def test_from_dict(self):
    """Test account from dict"""
    data =  {'name': 'name',
    'email': 'email',
    'phone_number': 'phone_number',
    'disabled': 'disabled',
    'date_joined': 'date_joined',
    }
    account = Account()
    account.from_dict(data)
    self.assertEqual(account.name,  second: "name")
    self.assertEqual(account.email,  second: "email")
    self.assertEqual(account.phone_number,  second: "phone_number")
    self.assertEqual(account.disabled,  second: "disabled")
    self.assertEqual(account.date_joined,  second: "date_joined")
```

Lines 45-48 test

```
def test_update(self):
    initial_account = Account(name='John Doe', email='john.doe@example.com')
    initial_account.create()
    account = Account.find(initial_account.id)
    account.name = 'Updated Name'
    account.update()
    account = Account.find(initial_account.id)
    self.assertEqual(account.name,  second: 'Updated Name')
```

Lines 52-54 test

```
def test_update_non_existing_account(self):
    new_account = Account(name='name', email='email')

    # Expect a DataValidationError when trying to update a non-existing account
    with self.assertRaises(DataValidationError):
        new_account.update()
```

Lines 74-75 test

```
def test_delete_account(self):
    data = ACCOUNT_DATA[self.rand] # get a random account
    account = Account(**data)
    account.create()
    account.delete()
    self.assertEqual(len(Account.all()), second: 0)
```

Task 5

Create test_update_a_counter(self)

```
def test_update_a_counter(self):
    """It should update a counter"""
    result = self.client.post('/counters/foo')
    updatedCounter = self.client.put('/counters/foo')
    self.assertEqual(updatedCounter.status_code, status.HTTP_200_OK)
    self.assertEqual(b'{"foo":1}\n', updatedCounter.data)
    self.assertGreater(updatedCounter.data, result.data)
```

This puts us in the RED phase.

Create update_counter(name)

```
@app.route( rule: '/counters/<name>', methods=['PUT'])
def update_counter(name):
    app.logger.info(f"Request to update counter: {name}")
    if name in COUNTERS:
        # increment 1
        COUNTERS[name] = COUNTERS[name] + 1
    return {name: COUNTERS[name]}, status.HTTP_200_OK
```

Now are GREEN as we have test and code to be tested

Create test to read a counter

```python
def test_read_a_counter(self):
    """It should read a counter"""
    readCounter = self.client.get('/counters/foo')
    self.assertEqual(readCounter.status_code, status.HTTP_200_OK)
    self.assertEqual( first: b'{"foo":0}\n', readCounter.data)
```

Create implementation of read a counter

```python
@app.route( rule: '/counters/<name>', methods=['GET'])
def read_counter(name):
    app.logger.info(f"Request to get counter: {name}")
    if name in COUNTERS:
        return {name: COUNTERS[name]}, status.HTTP_200_OK
```

Exceptions were all cleared as I did not complete this all at once.