Executive Report

Testing Lab

CS472 - 1001 | Group 2 | 9/17/2024
Brandon Bruno
brunob1@unlv.nevada.edu

Repositories

Link to group repository: [Click Here]
Link to my fork: [Click Here]

Task 1 – JPacman Test Coverage

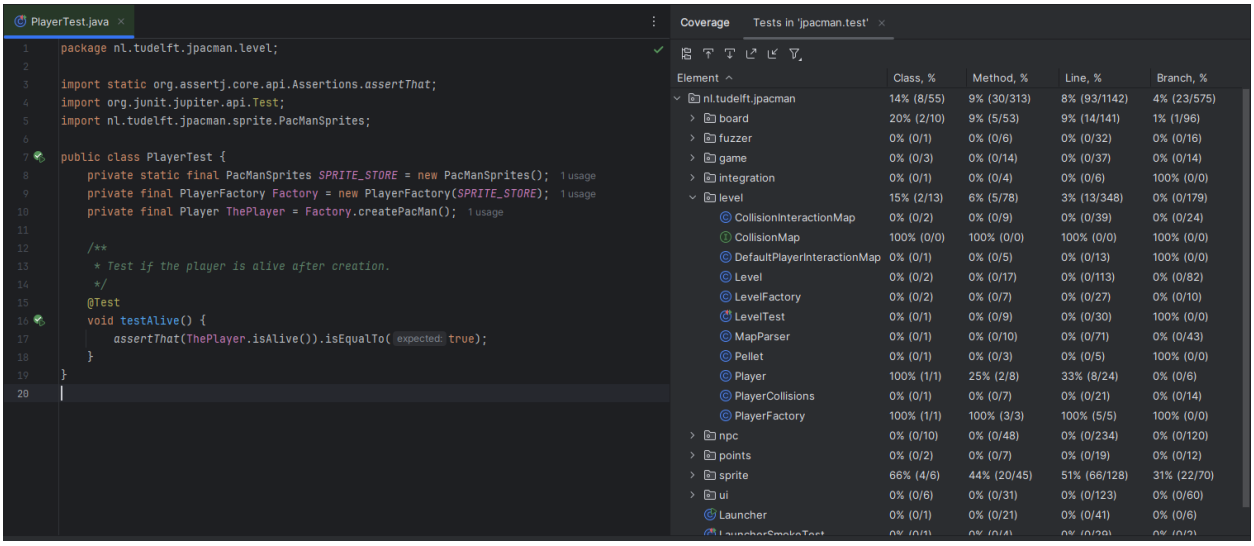
No, this coverage is not good enough. It is at 0% for nearly all packages.

Coverage | Tests in 'jpacman.test'

Element	Class, %	Method, %	Line, %	Branch, %
nl.tudelft.jpacman	3% (2/55)	1% (5/313)	1% (14/1128)	0% (1/557)
board	20% (2/10)	9% (5/53)	9% (14/141)	1% (1/96)
fuzzer	0% (0/1)	0% (0/6)	0% (0/32)	0% (0/16)
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% (0/179)
npc	0% (0/10)	0% (0/48)	0% (0/234)	0% (0/120)
points	0% (0/2)	0% (0/7)	0% (0/19)	0% (0/12)
sprite	0% (0/6)	0% (0/45)	0% (0/119)	0% (0/52)
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)
PacmanConfigurationException	0% (0/1)	0% (0/2)	0% (0/4)	100% (0/0)

Task 2

As we can see below, after creating the PlayerTest class and testing if the player is alive, the coverage increased for level.



Task 2.1

The three methods I created tests for are:

- restart() : src/main/sprite/AnimatedSprite.java
- randomMove() : src/main/npc/ghost/Blinky.java
- remove() : src/main/board/Square.java

The restart() method in the AnimatedSprite both starts and restarts the current animation for any animated sprite. The test verifies that when the method is called, it correctly resets the animation to the first frame, updates the last update time to the current time, and sets the animation to start playing. This ensures that the animation restarts as expected.

My code for the test method can be seen on the following page.

```java
package nl.tudelft.jpacman.sprite;

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import static org.assertj.core.api.Assertions.assertThat;

/**
 * Test class for the AnimatedSprite class.
 */
public class AnimatedSpriteTest {

    private AnimatedSprite animatedSprite;

    @BeforeEach
    void setUp() {
        Sprite[] frames = { new EmptySprite(), new EmptySprite() };
        int delay = 100;
        boolean loop = true;
        animatedSprite = new AnimatedSprite(frames, delay, loop);
    }

    /**
     * Test if the animation restarts correctly.
     */
    @Test
    void testRestart() {
        // Set some initial values
        animatedSprite.setAnimating(false);
        animatedSprite.restart();

        // Verify the values are reset correctly
        assertThat(getCurrentFrameIndex(animatedSprite)).isEqualTo(0);
        assertThat(isAnimating(animatedSprite)).isTrue();

        // Verify that lastUpdate is within a reasonable range of the current time
        long now = System.currentTimeMillis();
        long lastUpdate = getLastUpdate(animatedSprite);
        assertThat(lastUpdate).isGreaterThanOrEqualTo(now - 100);
        assertThat(lastUpdate).isLessThanOrEqualTo(now + 100);
    }

    // Helper methods to access private fields for testing purposes
    private int getCurrentFrameIndex(AnimatedSprite sprite) {
        try {
            java.lang.reflect.Field field = AnimatedSprite.class.getDeclaredField("current");
            field.setAccessible(true);
            return field.getInt(sprite);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

    private long getLastUpdate(AnimatedSprite sprite) {
        try {
            java.lang.reflect.Field field = AnimatedSprite.class.getDeclaredField("lastUpdate");
            field.setAccessible(true);
            return field.getLong(sprite);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

    private boolean isAnimating(AnimatedSprite sprite) {
        try {
            java.lang.reflect.Field field = AnimatedSprite.class.getDeclaredField("animating");
            field.setAccessible(true);
            return field.getBoolean(sprite);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }    }
}
```

Comparing the results to the screenshot from Task 2, we can see that the sprite package 'Method, %' has increased by 2%. Upon clicking the dropdown, we can see that the AnimatedSprite went up in percentage.

Coverage | Tests in 'jpacman.test'

Element	Class, %	Method, %	Line, %	Branch, %
nl.tudelft.jpacman	14% (8/55)	9% (31/313)	8% (97/1142)	4% (23/575)
board	20% (2/10)	9% (5/53)	9% (14/141)	1% (1/96)
fuzzer	0% (0/1)	0% (0/6)	0% (0/32)	0% (0/16)
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% (0/179)
npc	0% (0/10)	0% (0/48)	0% (0/234)	0% (0/120)
points	0% (0/2)	0% (0/7)	0% (0/19)	0% (0/12)
sprite	66% (4/6)	46% (21/45)	54% (70/128)	31% (22/70)
AnimatedSprite	100% (1/1)	45% (5/11)	43% (19/44)	10% (3/28)
EmptySprite	0% (0/1)	0% (0/4)	0% (0/4)	100% (0/0)
ImageSprite	100% (1/1)	85% (6/7)	76% (13/17)	50% (6/12)
PacManSprites	100% (1/1)	55% (5/9)	66% (16/24)	33% (2/6)
Sprite	100% (0/0)	100% (0/0)	100% (0/0)	100% (0/0)
SpriteStore	100% (1/1)	100% (5/5)	95% (22/23)	45% (11/24)
SpriteTest	0% (0/1)	0% (0/9)	0% (0/16)	100% (0/0)
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)
PacmanConfigurationException	0% (0/1)	0% (0/2)	0% (0/4)	100% (0/0)

The BlinkyTest class is designed to test the randomMove() method of the Blinky class, which is the red ghost in the Pac-Man game. The test class includes two main test methods. The first method, testRandomMove, simulates a scenario where Blinky can move in multiple directions (NORTH, SOUTH, EAST, WEST) and verifies that the chosen move is one of these accessible directions. The second method, testRandomMoveNoAccessibleDirections, simulates a scenario where no directions are accessible and verifies that the method returns null. These tests ensure that Blinky's random movement logic works correctly, both when there are available directions to move and when Blinky is surrounded by inaccessible squares. Since all the ghosts share this method, the same test could be applied to each ghost. However, for the purposes of this assignment, I only tested with Blinky. The following page contains my code for my BlinkyTest class.

```java
package nl.tudelft.jpacman.npc.ghost;

import nl.tudelft.jpacman.board.Direction;
import nl.tudelft.jpacman.board.Square;
import nl.tudelft.jpacman.sprite.Sprite;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import java.util.*;

import static org.assertj.core.api.Assertions.assertThat;
import static org.mockito.Mockito.mock;
import static org.mockito.Mockito.when;

/**
 * Test class for the Blinky class.
 */
public class BlinkyTest {

    private Blinky blinky;
    private Square square;

    @BeforeEach
    void setUp() {
        Map<Direction, Sprite> spriteMap = new HashMap<>();
        blinky = new Blinky(spriteMap);
        square = mock(Square.class);

        // Mock the behavior of the square to return itself when occupied
        when(square.getOccupants()).thenReturn(Collections.singletonList(blinky));
        when(square.getSquareAt(Direction.NORTH)).thenReturn(mock(Square.class));
        when(square.getSquareAt(Direction.SOUTH)).thenReturn(mock(Square.class));
        when(square.getSquareAt(Direction.EAST)).thenReturn(mock(Square.class));
        when(square.getSquareAt(Direction.WEST)).thenReturn(mock(Square.class));

        blinky.occupy(square);
    }

    /**
     * Test if Blinky can move in a random direction.
     */
    @Test
    void testRandomMove() {
        Square northSquare = mock(Square.class);
        Square southSquare = mock(Square.class);

        when(square.getSquareAt(Direction.NORTH)).thenReturn(northSquare);
        when(square.getSquareAt(Direction.SOUTH)).thenReturn(southSquare);
        when(northSquare.isAccessibleTo(blinky)).thenReturn(true);
        when(southSquare.isAccessibleTo(blinky)).thenReturn(true);

        List<Direction> accessibleDirections = Arrays.asList(Direction.NORTH, Direction.SOUTH);

        Direction move = blinky.randomMove();
        assertThat(accessibleDirections).contains(move);
    }

    /**
     * Test if Blinky returns null when no directions are accessible.
     */
    @Test
    void testRandomMoveNoAccessibleDirections() {
        for (Direction direction : Direction.values()) {
            Square mockSquare = mock(Square.class);
            when(square.getSquareAt(direction)).thenReturn(mockSquare);
            when(mockSquare.isAccessibleTo(blinky)).thenReturn(false);
        }

        Direction move = blinky.randomMove();
        assertThat(move).isNull(); }
}
```

As you can see, compared to the screenshot after the animated sprite test, the npc package (which includes the ghost package) was at 0% coverage. However, after running my tests, we can see that blinky method % is now at 75% and the overall coverage has increased.

Coverage | Tests in 'jpacman.test'

Element	Class, %	Method...	Line, %	Branch,...
nl.tudelft.jpacman	20% (11...	12% (39/...	10% (125...	7% (42/5...
board	30% (3/...	16% (9/53)	17% (25/...	13% (13/...
fuzzer	0% (0/1)	0% (0/6)	0% (0/32)	0% (0/16)
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/...	6% (5/78)	3% (13/3...	0% (0/179)
npc	20% (2/...	8% (4/48)	7% (17/2...	5% (7/126)
ghost	11% (1/9)	6% (3/44)	5% (12/2...	5% (7/126)
Blinky	100% (1...	75% (3/4)	54% (12/...	31% (7/22)
Clyde	0% (0/1)	0% (0/4)	0% (0/29)	0% (0/18)
GhostColor	0% (0/1)	0% (0/1)	0% (0/5)	100% (0/0)
GhostFactory	0% (0/1)	0% (0/5)	0% (0/6)	100% (0/0)
Inky	0% (0/1)	0% (0/5)	0% (0/30)	0% (0/22)
Navigation	0% (0/2)	0% (0/12)	0% (0/58)	0% (0/40)
NavigationTest	0% (0/1)	0% (0/9)	0% (0/56)	0% (0/8)
Pinky	0% (0/1)	0% (0/4)	0% (0/21)	0% (0/16)
Ghost	100% (1...	25% (1/4)	62% (5/8)	100% (0/0)
points	0% (0/2)	0% (0/7)	0% (0/19)	0% (0/12)
sprite	66% (4/6)	46% (21/...	54% (70/...	31% (22/...
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)

Lastly, the SquareTest class is designed to test the remove() method of the Square class. The test class includes two main test methods. The first method, testRemove(), verifies that a unit can be successfully removed from the square. It does this by first adding a unit to the square, checking that the unit is present, then removing the unit, and finally confirming that the unit is no longer present. The second method, testRemoveNonExistentUnit(), ensures that attempting to remove a unit that is not present in the square does not cause any issues. This is done by trying to remove a unit that was never added and verifying that the square's occupants remain unaffected. The BasicSquare class is a simple implementation of the Square class used for testing purposes.

The next page contains my code for the test.

```java
package nl.tudelft.jpacman.board;

import nl.tudelft.jpacman.sprite.Sprite;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import java.util.List;

import static org.assertj.core.api.Assertions.assertThat;
import static org.mockito.Mockito.mock;

/**
 * Test class for the Square class.
 */
public class SquareTest {

    private Square square;
    private Unit unit;

    @BeforeEach
    void setUp() {
        square = new BasicSquare();
        unit = mock(Unit.class);
    }

    /**
     * Test if a unit can be removed from the square.
     */
    @Test
    void testRemove() {
        square.put(unit);
        assertThat(square.getOccupants()).contains(unit);

        square.remove(unit);
        assertThat(square.getOccupants()).doesNotContain(unit);
    }

    /**
     * Test if removing a unit that is not present does not cause any issues.
     */
    @Test
    void testRemoveNonExistentUnit() {
        square.remove(unit);
        assertThat(square.getOccupants()).doesNotContain(unit);
    }

    /**
     * Basic implementation of Square for testing purposes.
     */
    private static class BasicSquare extends Square {
        @Override
        public boolean isAccessibleTo(Unit unit) {
            return true;
        }

        @Override
        public Sprite getSprite() {
            return mock(Sprite.class);
        }
    }
}
```

The screenshot below shows that Square only has one method test so far, with SquareTest being at zero coverage.

Coverage | Tests in 'jpacman.test'

Element	Class, %	Method...	Line, %	Branch,...
nl.tudelft.jpacman	20% (11...	12% (39/...	10% (125...	7% (42/5...
board	30% (3/...	16% (9/53)	17% (25/...	13% (13/...
Board	0% (0/1)	0% (0/7)	0% (0/17)	0% (0/22)
BoardFactory	0% (0/3)	0% (0/11)	0% (0/27)	0% (0/12)
BoardFactoryTest	0% (0/1)	0% (0/6)	0% (0/18)	100% (0/0)
BoardTest	0% (0/1)	0% (0/3)	0% (0/3)	100% (0/0)
Direction	100% (1...	75% (3/4)	90% (10/...	100% (0/0)
Square	100% (1...	12% (1/8)	4% (1/23)	3% (1/28)
SquareTest	0% (0/1)	0% (0/4)	0% (0/13)	100% (0/0)
Unit	100% (1...	50% (5/10)	48% (14/...	35% (12/...

After running the tests, we can now see that overall coverage has increased:

Coverage | Tests in 'jpacman.test'

Element	Class, %	Method...	Line, %	Branch,...
nl.tudelft.jpacman	21% (12...	15% (47/3...	13% (151...	8% (51/5...
board	40% (4/...	32% (17/5...	36% (51/...	22% (22/...
Board	0% (0/1)	0% (0/7)	0% (0/17)	0% (0/22)
BoardFactory	0% (0/3)	0% (0/11)	0% (0/27)	0% (0/12)
BoardFactoryTest	0% (0/1)	0% (0/6)	0% (0/18)	100% (0/0)
BoardTest	0% (0/1)	0% (0/3)	0% (0/3)	100% (0/0)
Direction	100% (1...	75% (3/4)	90% (10/...	100% (0/0)
Square	100% (1...	75% (6/8)	69% (16/...	35% (10/...
SquareTest	100% (1...	100% (3/3)	100% (11...	100% (0/0)
Unit	100% (1...	50% (5/10)	48% (14/...	35% (12/...
fuzzer	0% (0/1)	0% (0/6)	0% (0/32)	0% (0/16)
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/...	6% (5/78)	3% (13/3...	0% (0/179)
npc	20% (2/...	8% (4/48)	7% (17/2...	5% (7/126)
points	0% (0/2)	0% (0/7)	0% (0/19)	0% (0/12)
sprite	66% (4/6)	46% (21/...	54% (70/...	31% (22/...
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)
PacmanConfigurationE›	0% (0/1)	0% (0/2)	0% (0/4)	100% (0/0)

Task 3 – JaCoCo Report on JPacman

Below is my JaCoCo report found in build/reports/jacoco/test/html/index.html

jpacman

jpacman

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
nl.tudelft.jpacman.level		66%		57%	74	155	104	344	21	69	4	12
nl.tudelft.jpacman.npc.ghost		77%		59%	52	105	34	181	4	34	0	8
nl.tudelft.jpacman.ui		78%		47%	54	86	21	144	7	31	0	6
default		0%		0%	12	12	21	21	5	5	1	1
nl.tudelft.jpacman.sprite		86%		59%	30	70	11	113	5	38	0	5
nl.tudelft.jpacman.board		86%		58%	44	93	2	110	0	40	0	7
nl.tudelft.jpacman		67%		25%	12	30	18	52	6	24	1	2
nl.tudelft.jpacman.points		59%		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	1,195 of 4,755	74%	287 of 637	54%	289	590	220	1,039	50	268	6	47

- Yes, there is similar information covered between JaCoCo's report and IntelliJ, but the percentages of how much is covered seems to be slightly different between the two tests. I think this is because JaCoCo provides a much more in depth report and is incredibly detailed compared to IntelliJ.
- I think the source code visualization is super helpful! Highlighting exactly what lines are missing tests makes our life so easy because we can easily write tests for those lines to reach 100% coverage.
- I much prefer JaCoCo. The IntelliJ is nice for quick tests because it is built into the IDE, but for big projects like the one we will be working on later in this semester, I definitely prefer the detailed coverage.

Task 4 – Working with Python Test Coverage

The initial results of the pytest is shown in the screenshot below. Our goal is to raise the test coverage to 100%.

```
PS C:\Users\Brandon\Desktop\test_coverage> python -m pytest --cov=.
================================================ test session starts =================================================
platform win32 -- Python 3.11.9, pytest-8.3.3, pluggy-1.5.0 -- C:\Users\Brandon\AppData\Local\Microsoft\WindowsApps\Pyth
onSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\Brandon\Desktop\test_coverage
configfile: pytest.ini
plugins: cov-5.0.0, typeguard-2.13.3
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 win32, python 3.11.9-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
tests\__init__.py             0      0   100%
tests\test_account.py        30      0   100%
-------------------------------------------------------
TOTAL                        77     13    83%


================================================= 2 passed in 1.54s ==================================================
```

After pasting the provided test methods test_repr(), and test_to_dict(), we are left with lines 34-35, 45-48, 52-54, 74-75. Below contains the list of methods at those line numbers, as well as the code I made for the testing.

- Lines 34-35: from_dict

```
def test_from_dict():
    account = Account()
    data = {
        "name": "Brandon Bruno",
        "email": "brunob1@unlv.nevada.edu",
        "phone_number": "1234567890",
        "disabled": False
    }
    account.from_dict(data)
    assert account.name == "Brandon Bruno"
    assert account.email == "brunob1@unlv.nevada.edu"
    assert account.phone_number == "1234567890"
    assert account.disabled is False
```

- Lines 45-48: update

```
def test_update():
    # Test updating an account with a valid ID
    account = Account(name="bruno b")
    account.create()
    account.name = "Brandon Bruno"
    account.update()
    updated_account = Account.find(account.id)
    assert updated_account.name == "Brandon Bruno"

    # Test updating an account without an ID
    account_without_id = Account(name="Brandon Bruno")
    raised_exception = False
    try:
        account_without_id.update()
    except DataValidationError:
        raised_exception = True
    assert raised_exception, "DataValidationError not raised"
```

- Lines 52-54: delete

```
def test_delete():
    account = Account(name="Brandon Bruno")
    account.create()
    account_id = account.id
    account.delete()
    deleted_account = Account.find(account_id)
    assert deleted_account is None
```

- Lines 74-75: find

```
def test_find():
    account = Account(name="Brandon Bruno")
    account.create()
    found_account = Account.find(account.id)
    assert found_account is not None
    assert found_account.name == "Brandon Bruno"
```

Below is the screenshot showing we are now at 100% coverage:

```
PS C:\Users\Brandon\Desktop\test_coverage> python -m pytest --cov=.
========================================================================================================== test session starts ==========================================================================================================
platform win32 -- Python 3.11.9, pytest-8.3.3, pluggy-1.5.0 -- C:\Users\Brandon\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\Brandon\Desktop\test_coverage
configfile: pytest.ini
plugins: cov-5.0.0, typeguard-2.13.3
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
  C:\Users\Brandon\Desktop\test_coverage\models\account.py:75: LegacyAPIWarning: The Query.get() method is considered legacy as of the 1.x series of SQLAlchemy and becomes a legacy construct in 2.0. The method is now available as Session
.get() (deprecated since: 2.0) (Background on SQLAlchemy 2.0 at: 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 win32, python 3.11.9-final-0 -----------
Name                     Stmts   Miss  Cover   Missing
------------------------------------------------------
models\__init__.py           7      0   100%
models\account.py           40      0   100%
tests\__init__.py            0      0   100%
tests\test_account.py       79      0   100%
------------------------------------------------------
TOTAL                      126      0   100%

========================================================================================================== 8 passed, 3 warnings in 0.58s ==========================================================================================================
```

Task 5 – Test Driven Development (TDD)

For the first part of this task, we were tasked with implementing a RESTful API for a counter web service using Flask. The goal was to create endpoints that allow users to create, update, and read counters.

RED PHASE:

1. Created the update test case in test_counter.py:

```
def test_update_a_counter(client):
    """It should update a counter"""
    # Step 1: Make a call to Create a counter.
    result = client.post('/counters/boo')
    assert result.status_code == status.HTTP_201_CREATED
    # Step 2: Ensure that it returned a successful return code.
    data = result.get_json()
    assert data['boo'] == 0
    # Step 3: Check the counter value as a baseline.
    baseline = data['boo']
    # Step 4: Update the counter
```

```
    result = client.put('/counters/boo')
    assert result.status_code == status.HTTP_200_OK
    # Step 5: Ensure that it returned a successful return code.
    data = result.get_json()
    # Step 6: Check that the counter value is one more than the baseline
              you measured in step 3.
    assert data['boo'] == baseline + 1
```

The test case was designed to create a counter, update it, and verify the updated value.

2. Created the read test Case in test_counter.py

```
def test_read_a_counter(client):
    """It should read a counter"""
    result = client.post('/counters/boo2')
    assert result.status_code == status.HTTP_201_CREATED
    data = result.get_json()
    assert data['boo2'] == 0
    result = client.get('/counters/boo2')
    assert result.status_code == status.HTTP_200_OK
    data = result.get_json()
    assert data['boo2'] == 0
```

The test case will create a counter and then read it to verify the value.

GREEN PHASE:

1. Implemented the update function in counter.py:

```
@app.route('/counters/<name>', methods=['PUT'])
def update_counter(name):
    """Update a counter"""
    global COUNTERS
    if name not in COUNTERS:
        return {"Message": f"Counter {name} not found"},
        status.HTTP_404_NOT_FOUND
    COUNTERS[name] += 1
    return {name: COUNTERS[name]}, status.HTTP_200_OK
```

We implemented the update_counter function to handle the PUT request and increment the counter.

2. Implemented the Read (GET) Function in counter.py:

```
@app.route('/counters/<name>', methods=['GET'])
def get_counter(name):
    """Get a counter"""
    if name not in COUNTERS:
        return {"Message": f"Counter {name} does not exist"},
        status.HTTP_404_NOT_FOUND
```

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

REFACTOR PHASE:

I reviewed the code for any potential improvements.

```
PS C:\Users\Brandon\Desktop\tdd> python -m pytest --cov=counter
================================================= test session starts =================================================
platform win32 -- Python 3.11.9, pytest-8.3.3, pluggy-1.5.0 -- C:\Users\Brandon\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFou
cachedir: .pytest_cache
rootdir: C:\Users\Brandon\Desktop\tdd
configfile: pytest.ini
plugins: cov-5.0.0, typeguard-2.13.3
collected 4 items

tests/test_counter.py::TestCounterEndPoints::test_create_a_counter PASSED                                          [ 25%]
tests/test_counter.py::TestCounterEndPoints::test_duplicate_a_counter PASSED                                       [ 50%]
tests/test_counter.py::TestCounterEndPoints::test_update_a_counter PASSED                                          [ 75%]
tests/test_counter.py::test_read_a_counter PASSED                                                                  [100%]C:\Users\Brand
ython311\site-packages\coverage\inorout.py:504: CoverageWarning: Module counter was never imported. (module-not-imported)
  self.warn(f"Module {pkg} was never imported.", slug="module-not-imported")


---------- coverage: platform win32, python 3.11.9-final-0 -----------
Name              Stmts   Miss  Cover   Missing
-----------------------------------------------
src\__init__.py       0      0   100%
src\counter.py       22      2    91%   27, 35
src\status.py         6      0   100%
-----------------------------------------------
TOTAL                28      2    93%


================================================= 4 passed in 0.23s =================================================
```

As you can see, lines 27 and 35 were missing. These lines correspond to the error handling for when a counter is not found. I added two more test cases to trigger those conditions:

```
    def test_update_non_existent_counter(self, client):
        """It should return 404 for updating a non-existent counter"""
        result = client.put('/counters/nonexistent')
        assert result.status_code == status.HTTP_404_NOT_FOUND

def test_read_nonexistent_counter(client):
    """It should return 404 for reading a nonexistent counter"""
    result = client.get('/counters/nonexistent')
    assert result.status_code == status.HTTP_404_NOT_FOUND
```

As seen in the screenshot below, this put me at 100% coverage.

```
PS C:\Users\Brandon\Desktop\tdd> python -m pytest --cov=counter
========================================================================================================== test session starts ==========
platform win32 -- Python 3.11.9, pytest-8.3.3, pluggy-1.5.0 -- C:\Users\Brandon\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundati
cachedir: .pytest_cache
rootdir: C:\Users\Brandon\Desktop\tdd
configfile: pytest.ini
plugins: cov-5.0.0, typeguard-2.13.3
collected 6 items

tests/test_counter.py::TestCounterEndPoints::test_create_a_counter PASSED
tests/test_counter.py::TestCounterEndPoints::test_duplicate_a_counter PASSED
tests/test_counter.py::TestCounterEndPoints::test_update_a_counter PASSED
tests/test_counter.py::TestCounterEndPoints::test_update_non_existent_counter PASSED
tests/test_counter.py::test_read_a_counter PASSED
tests/test_counter.py::test_read_nonexistent_counter PASSED
:\Users\Brandon\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-package
)
  self.warn(f"Module {pkg} was never imported.", slug="module-not-imported")


---------- coverage: platform win32, python 3.11.9-final-0 -----------
Name              Stmts   Miss  Cover   Missing
-----------------------------------------------
src\__init__.py       0      0   100%
src\counter.py       22      0   100%
src\status.py         6      0   100%
-----------------------------------------------
TOTAL                28      0   100%


========================================================================================================== 6 passed in 0.23s ==========
PS C:\Users\Brandon\Desktop\tdd>
```

Exceptions Encountered:

- None - The implementation was straightforward and passed the coverage tests.