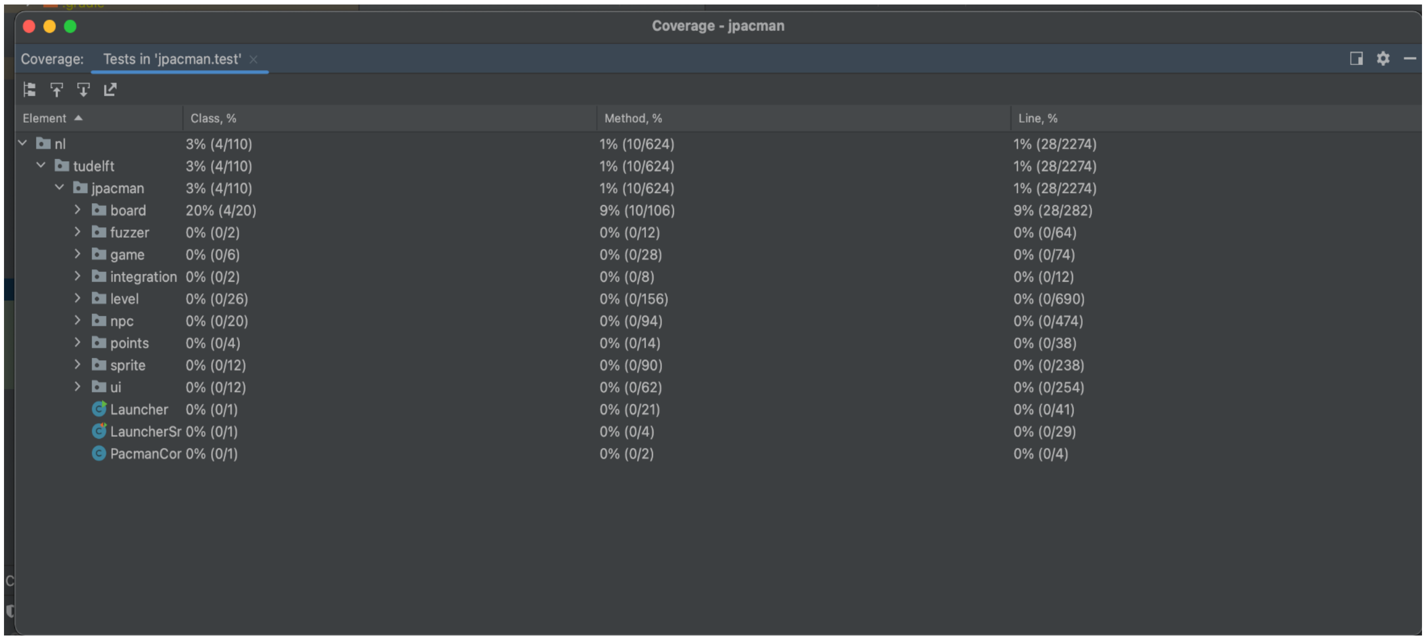
Task 2.1:

This is the IntelliJ coverage prior to any changes made:

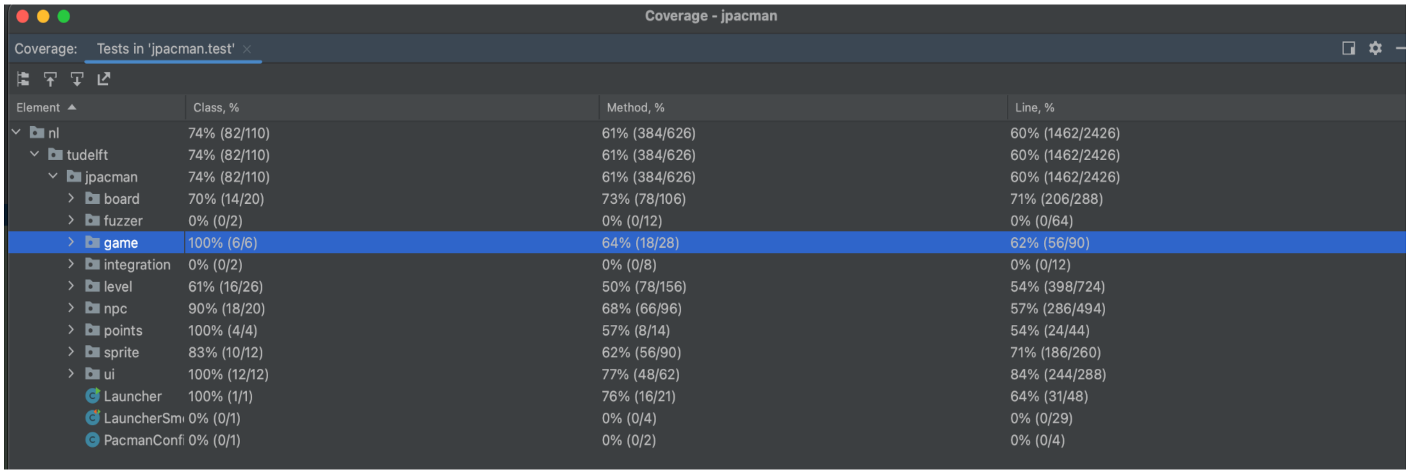


Coverage - jpacman

Coverage: Tests in 'jpacman.test'

Element	Class, %	Method, %	Line, %
nl	3% (4/110)	1% (10/624)	1% (28/2274)
tudelft	3% (4/110)	1% (10/624)	1% (28/2274)
jpacman	3% (4/110)	1% (10/624)	1% (28/2274)
board	20% (4/20)	9% (10/106)	9% (28/282)
fuzzer	0% (0/2)	0% (0/12)	0% (0/64)
game	0% (0/6)	0% (0/28)	0% (0/74)
integration	0% (0/2)	0% (0/8)	0% (0/12)
level	0% (0/26)	0% (0/156)	0% (0/690)
npc	0% (0/20)	0% (0/94)	0% (0/474)
points	0% (0/4)	0% (0/14)	0% (0/38)
sprite	0% (0/12)	0% (0/90)	0% (0/238)
ui	0% (0/12)	0% (0/62)	0% (0/254)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)
LauncherSr	0% (0/1)	0% (0/4)	0% (0/29)
PacmanCor	0% (0/1)	0% (0/2)	0% (0/4)

This is the IntelliJ coverage after I developed a test for isInProgress() for the Game.Java file:



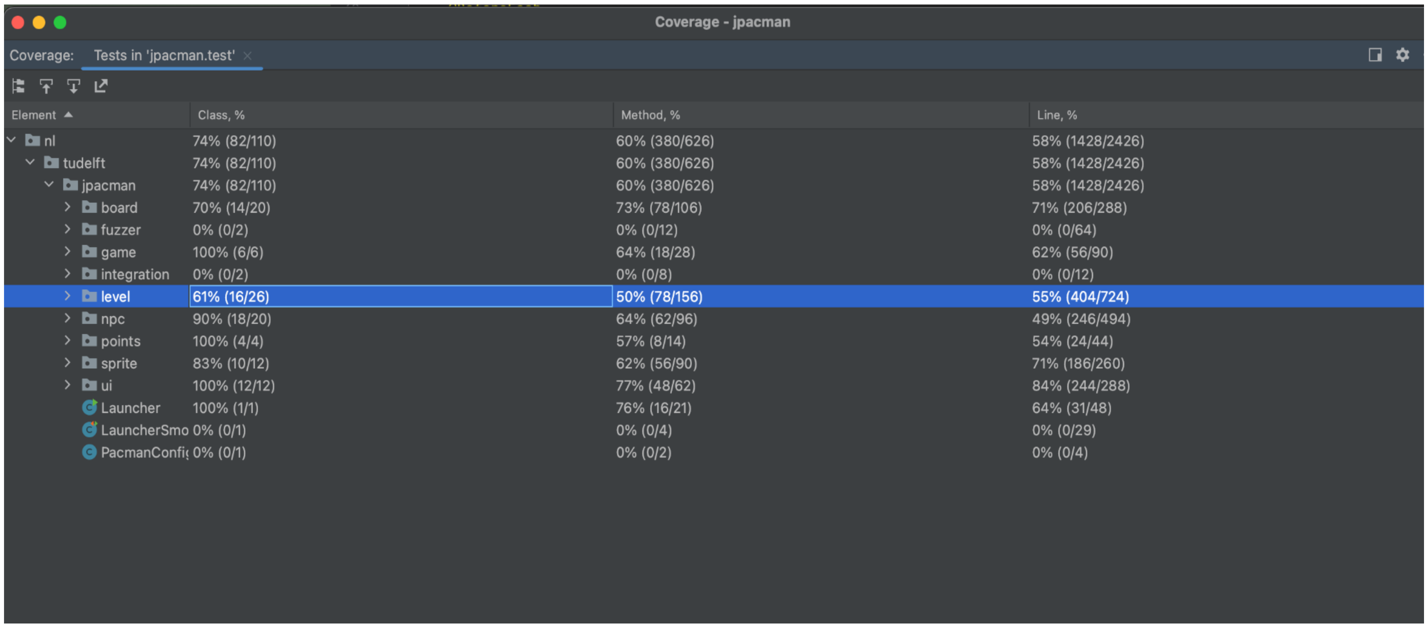
Element	Class, %	Method, %	Line, %
nl	74% (82/110)	61% (384/626)	60% (1462/2426)
tudelft	74% (82/110)	61% (384/626)	60% (1462/2426)
jpacman	74% (82/110)	61% (384/626)	60% (1462/2426)
board	70% (14/20)	73% (78/106)	71% (206/288)
fuzzer	0% (0/2)	0% (0/12)	0% (0/64)
game	100% (6/6)	64% (18/28)	62% (56/90)
integration	0% (0/2)	0% (0/8)	0% (0/12)
level	61% (16/26)	50% (78/156)	54% (398/724)
npc	90% (18/20)	68% (66/96)	57% (286/494)
points	100% (4/4)	57% (8/14)	54% (24/44)
sprite	83% (10/12)	62% (56/90)	71% (186/260)
ui	100% (12/12)	77% (48/62)	84% (244/288)
Launcher	100% (1/1)	76% (16/21)	64% (31/48)
LauncherSm	0% (0/1)	0% (0/4)	0% (0/29)
PacmanConfi	0% (0/1)	0% (0/2)	0% (0/4)

Here is the code for this test:

```
public class InProgressTest {

    private Launcher launcher;

    @Test
    void testIsInProgress() {

        launcher = new Launcher();
        launcher.launch();

        Game game = launcher.getGame();
        game.start();

        assertThat(game.isInProgress()).isEqualTo(true);
    }
}
```

This test launches the game and checks if it is running to confirm the in progress function. This test created a big difference in coverage moving up from 3% to 74% for class, from 1% to 61% for method, and from 1% to 60% for lines covered.

This is the IntelliJ coverage after I developed a test for isInProgress() for the Game.Java file:


Coverage - jpacman

Coverage: Tests in 'jpacman.test'

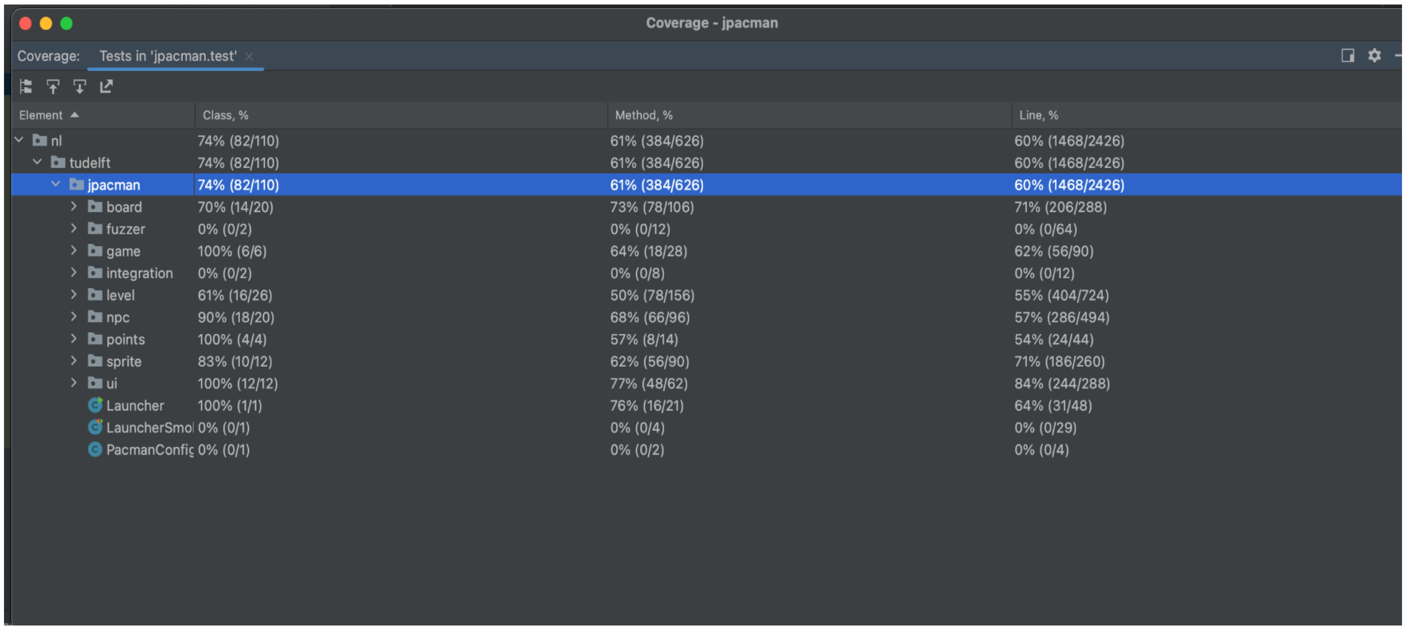
Element	Class, %	Method, %	Line, %
nl	74% (82/110)	60% (380/626)	58% (1428/2426)
tudelft	74% (82/110)	60% (380/626)	58% (1428/2426)
jpacman	74% (82/110)	60% (380/626)	58% (1428/2426)
board	70% (14/20)	73% (78/106)	71% (206/288)
fuzzer	0% (0/2)	0% (0/12)	0% (0/64)
game	100% (6/6)	64% (18/28)	62% (56/90)
integration	0% (0/2)	0% (0/8)	0% (0/12)
level	61% (16/26)	50% (78/156)	55% (404/724)
npc	90% (18/20)	64% (62/96)	49% (246/494)
points	100% (4/4)	57% (8/14)	54% (24/44)
sprite	83% (10/12)	62% (56/90)	71% (186/260)
ui	100% (12/12)	77% (48/62)	84% (244/288)
Launcher	100% (1/1)	76% (16/21)	64% (31/48)
LauncherSmo	0% (0/1)	0% (0/4)	0% (0/29)
PacmanConfi	0% (0/1)	0% (0/2)	0% (0/4)

Here is the code for this test:

```
void testInProgress() {

        level = new Level(board, Lists.newArrayList(ghost),
Lists.newArrayList(
            square1, square2), collisions);
        level.start();

        assertThat(level.isInProgress()).isEqualTo(true);
}
```

This test creates a new level and then runs the test to check if the level is in progress. This test did not have as big of an impact. Level increased by 1% for lines, and that was the only improvement.

This is the IntelliJ coverage after I developed a test for getInterval() for the Ghost.Java file:


Coverage - jpacman

Coverage: Tests in 'jpacman.test'

Element	Class, %	Method, %	Line, %
nl	74% (82/110)	61% (384/626)	60% (1468/2426)
tudelft	74% (82/110)	61% (384/626)	60% (1468/2426)
jpacman	74% (82/110)	61% (384/626)	60% (1468/2426)
board	70% (14/20)	73% (78/106)	71% (206/288)
fuzzer	0% (0/2)	0% (0/12)	0% (0/64)
game	100% (6/6)	64% (18/28)	62% (56/90)
integration	0% (0/2)	0% (0/8)	0% (0/12)
level	61% (16/26)	50% (78/156)	55% (404/724)
npc	90% (18/20)	68% (66/96)	57% (286/494)
points	100% (4/4)	57% (8/14)	54% (24/44)
sprite	83% (10/12)	62% (56/90)	71% (186/260)
ui	100% (12/12)	77% (48/62)	84% (244/288)
Launcher	100% (1/1)	76% (16/21)	64% (31/48)
LauncherSmol	0% (0/1)	0% (0/4)	0% (0/29)
PacmanConfig	0% (0/1)	0% (0/2)	0% (0/4)

Here is the code for this test:

```
void testInProgress() {

        level = new Level(board, Lists.newArrayList(ghost),
Lists.newArrayList(
            square1, square2), collisions);
        level.start();

        assertThat(ghost.getInterval()).isNotNull();
}
```

This test first establishes a new level and then confirms that the getInterval function returns a non null value. This test added slight improvements with a 1% increase in method and a 2% increase in lines covered.

Task 3.0:

The coverage from JaCoCo is not similar to the stats that I received from IntelliJ. The one main similarity is that both list the coverage at 74% for the main. Other than that, every other percentage shown is different. On JaCoCo, the information is listed out more visually and I can have an easier time understanding what branches and instructions were covered.

I think that the visualization of the uncovered branches is particularly useful, because it is information that I did not easily have through IntelliJ.

I prefere INtelliJ’s coverage window. I think that the reason that I like this window is because it presents the information in a more clear way. With the JaCoCo, the vizualization is helpful, but I’m not entirely sure what all of it means. I think that with the IntelliJ coverage it is easier to see how your tests are performing.

Here is a link to my repo:
https://github.com/alexfox361/jpacman.