Task 2.1 Report:

Test coverage after task 2:

Element	Class, %	Method, %	Line, %
nl	16% (18/110)	9% (60/624)	8% (190/2306)
tudelft	16% (18/110)	9% (60/624)	8% (190/2306)
jpacman	16% (18/110)	9% (60/624)	8% (190/2306)
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	15% (4/26)	6% (10/156)	3% (26/700)
npc	0% (0/20)	0% (0/94)	0% (0/474)
points	0% (0/4)	0% (0/14)	0% (0/38)
sprite	83% (10/12)	44% (40/90)	52% (136/260)
ui	0% (0/12)	0% (0/62)	0% (0/254)
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)

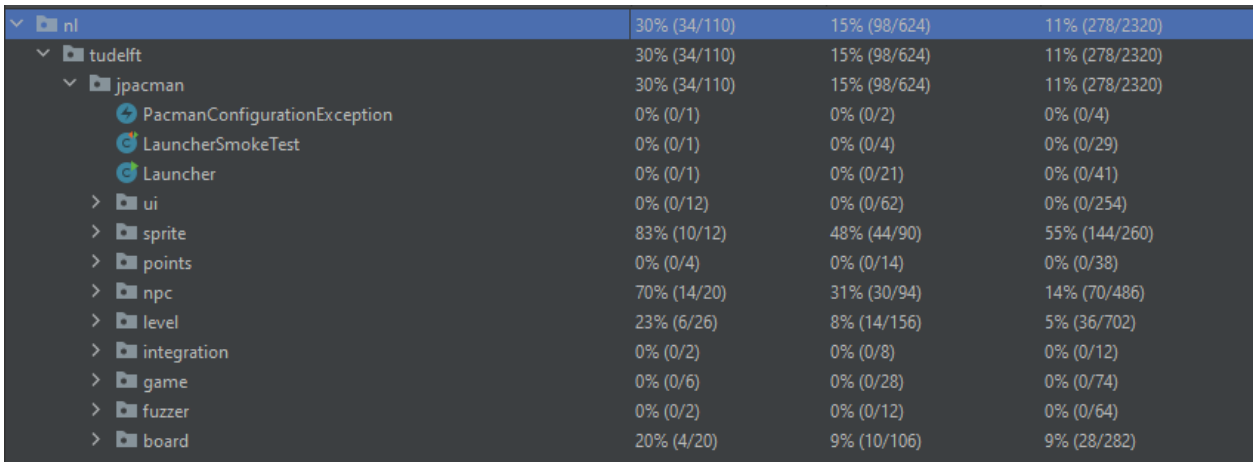
After adding ghost factory tests:

nl	29% (32/110)	14% (92/624)	11% (266/2318)
tudelft	29% (32/110)	14% (92/624)	11% (266/2318)
jpacman	29% (32/110)	14% (92/624)	11% (266/2318)
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	15% (4/26)	6% (10/156)	3% (26/700)
npc	70% (14/20)	31% (30/94)	14% (70/486)
points	0% (0/4)	0% (0/14)	0% (0/38)
sprite	83% (10/12)	46% (42/90)	54% (142/260)
ui	0% (0/12)	0% (0/62)	0% (0/254)
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)

```java
public class GhostTest {
    2 usages
    private static final PacManSprites SPRITE_STORE = new PacManSprites();
    no usages
    private PlayerFactory Factory = new PlayerFactory(SPRITE_STORE);
    4 usages
    private GhostFactory ghostFactory = new GhostFactory(SPRITE_STORE);


    no usages
    @Test
    void testGhostCreation() {
        assert(ghostFactory.createBlinky() instanceof Blinky);
        assert(ghostFactory.createClyde() instanceof Clyde);
        assert(ghostFactory.createInky() instanceof Inky);
        assert(ghostFactory.createPinky() instanceof Pinky);
    }
```

After adding pellet tests:

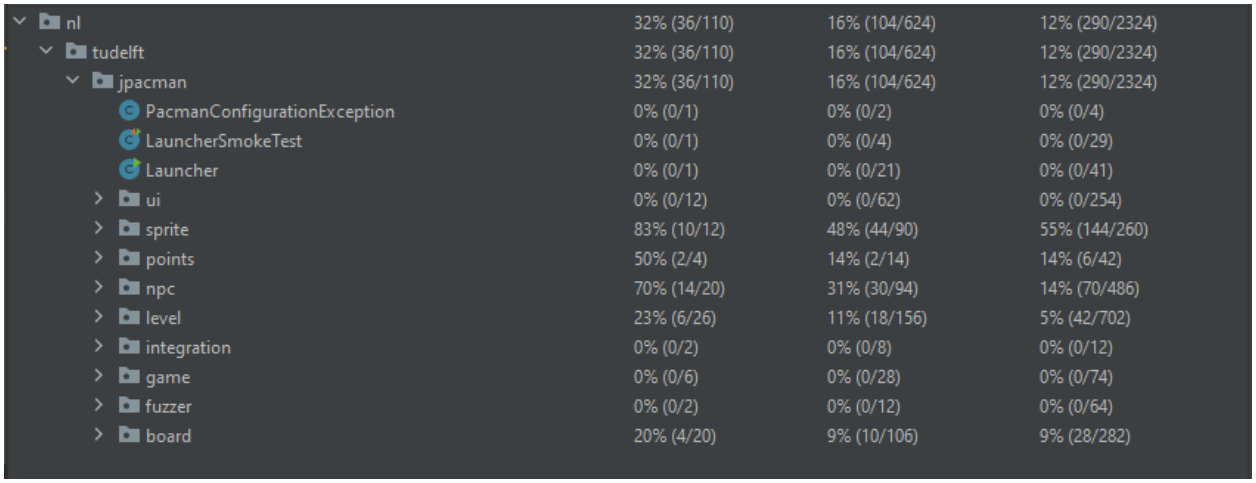


nl	30% (34/110)	15% (98/624)	11% (278/2320)
tudelft	30% (34/110)	15% (98/624)	11% (278/2320)
jpacman	30% (34/110)	15% (98/624)	11% (278/2320)
PacmanConfigurationException	0% (0/1)	0% (0/2)	0% (0/4)
LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)
ui	0% (0/12)	0% (0/62)	0% (0/254)
sprite	83% (10/12)	48% (44/90)	55% (144/260)
points	0% (0/4)	0% (0/14)	0% (0/38)
npc	70% (14/20)	31% (30/94)	14% (70/486)
level	23% (6/26)	8% (14/156)	5% (36/702)
integration	0% (0/2)	0% (0/8)	0% (0/12)
game	0% (0/6)	0% (0/28)	0% (0/74)
fuzzer	0% (0/2)	0% (0/12)	0% (0/64)
board	20% (4/20)	9% (10/106)	9% (28/282)

```java
public class PelletTest {
    1 usage
    private static final PacManSprites SPRITE_STORE = new PacManSprites();
    2 usages
    private static Pellet pellet;

    no usages
    @Test
    void testPellet() {
        pellet = new Pellet( points: 22, SPRITE_STORE.getPelletSprite());
        assertThat(pellet.getValue()).isEqualTo( expected: 22);
    }
}
```

After adding point calculator tests:



nl	32% (36/110)	16% (104/624)	12% (290/2324)
tudelft	32% (36/110)	16% (104/624)	12% (290/2324)
jpacman	32% (36/110)	16% (104/624)	12% (290/2324)
PacmanConfigurationException	0% (0/1)	0% (0/2)	0% (0/4)
LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)
ui	0% (0/12)	0% (0/62)	0% (0/254)
sprite	83% (10/12)	48% (44/90)	55% (144/260)
points	50% (2/4)	14% (2/14)	14% (6/42)
npc	70% (14/20)	31% (30/94)	14% (70/486)
level	23% (6/26)	11% (18/156)	5% (42/702)
integration	0% (0/2)	0% (0/8)	0% (0/12)
game	0% (0/6)	0% (0/28)	0% (0/74)
fuzzer	0% (0/2)	0% (0/12)	0% (0/64)
board	20% (4/20)	9% (10/106)	9% (28/282)

```
private int points;
1 usage
private Pellet pellet = new Pellet(points, SPRITE_STORE.getPelletSprite());

1 usage
private PointCalculator pointCalculator = new DefaultPointCalculator();

no usages
@Test
void testDefaultPointCalculator() {
    pointCalculator.consumedAPellet(ThePlayer, pellet);
    assertThat(ThePlayer.getScore()).isEqualTo(points);
}
```

The three screenshots above this show the coverage after adding my own unit tests. The first one tests the Ghost factory and ensures that each of the separate ghosts is instantiated correctly. The pellet test ensures that the pellets behave as expected after exiting the constructor. And the point calculator ensures that the player consuming pellet function works as intended.

As can be seen from the reports, these small code snippets add a lot of test coverage very simply. It might be more difficult to test the functionality of classes that utilize a lot of dependencies or have void functions. It is still important to test these classes, but as this was just a demo I found optimizing for more coverage to be more important.

Below is the JaCoco report. Jacoco has some advantages in that you can use the html links to quickly look at the test results. However, it is hard to argue against the integrated interface within the IDE for JUnit. Perhaps I would use this if I were in a minimal environment where I was using just the command line and a web browser.

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