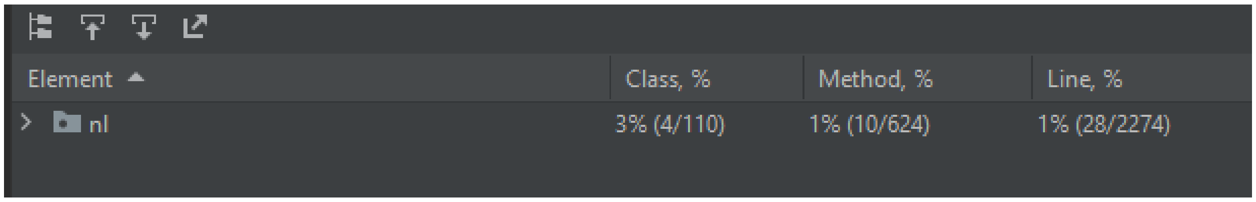
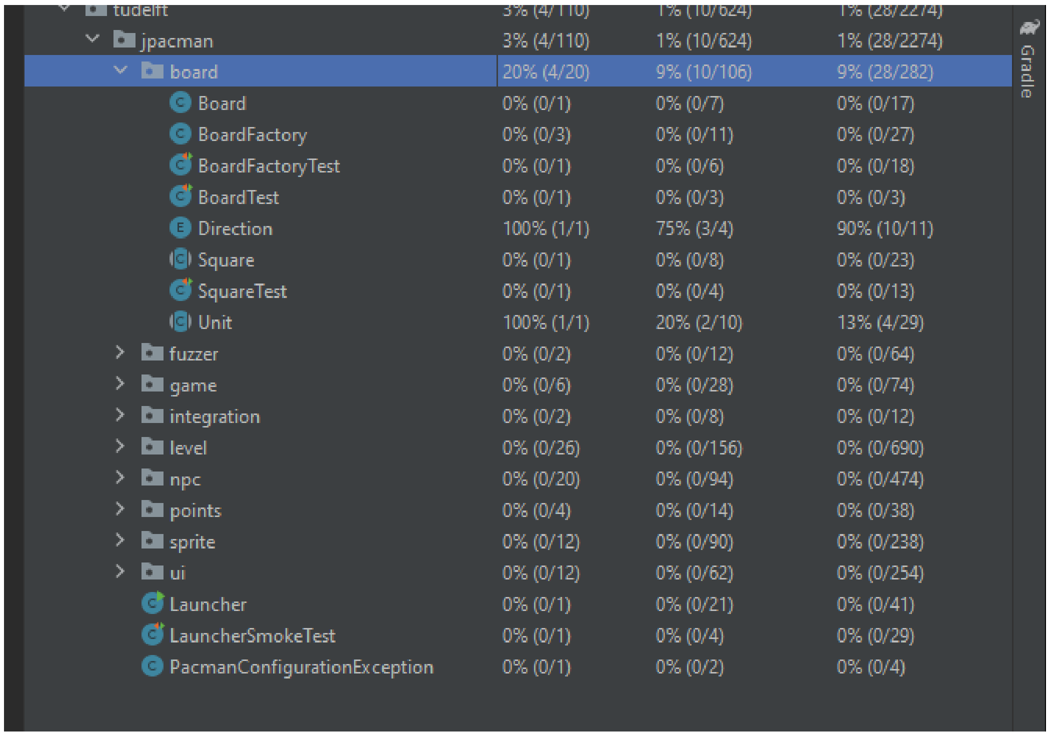
Chris Catechis
Dr. Businge
CS 472
2 February 2023

Unit Testing Lab

Task 1 (Initial test coverage):

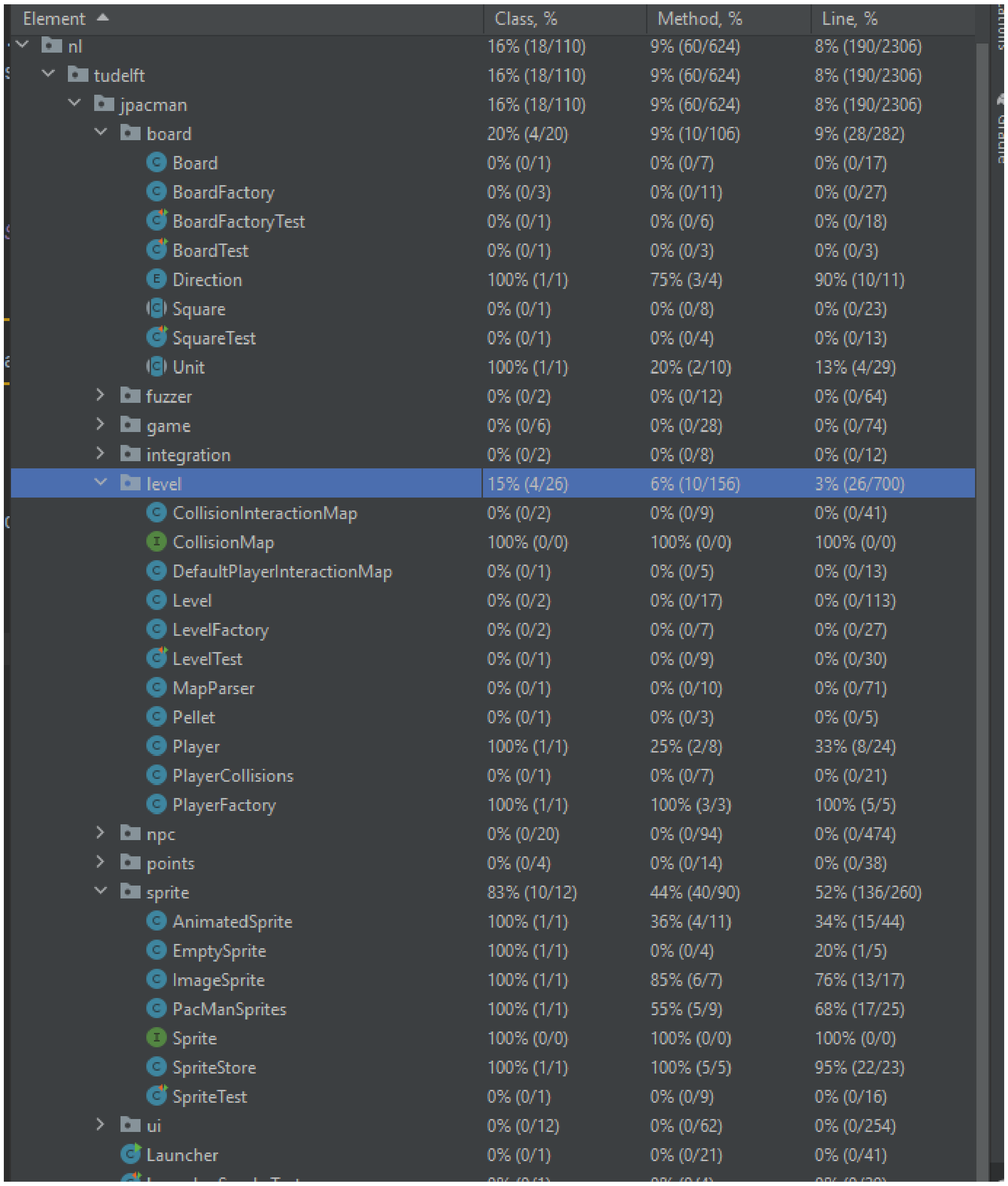


Element	Class, %	Method, %	Line, %
nl	3% (4/110)	1% (10/624)	1% (28/2274)



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)
Board	0% (0/1)	0% (0/7)	0% (0/17)
BoardFactory	0% (0/3)	0% (0/11)	0% (0/27)
BoardFactoryTest	0% (0/1)	0% (0/6)	0% (0/18)
BoardTest	0% (0/1)	0% (0/3)	0% (0/3)
Direction	100% (1/1)	75% (3/4)	90% (10/11)
Square	0% (0/1)	0% (0/8)	0% (0/23)
SquareTest	0% (0/1)	0% (0/4)	0% (0/13)
Unit	100% (1/1)	20% (2/10)	13% (4/29)
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)
LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)
PacmanConfigurationException	0% (0/1)	0% (0/2)	0% (0/4)

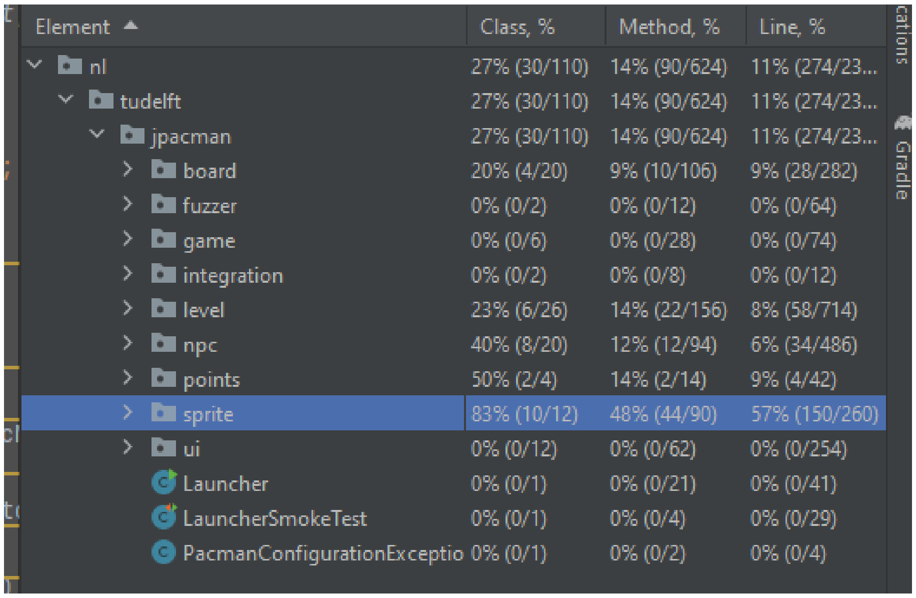
Task 2 (add playerTest):



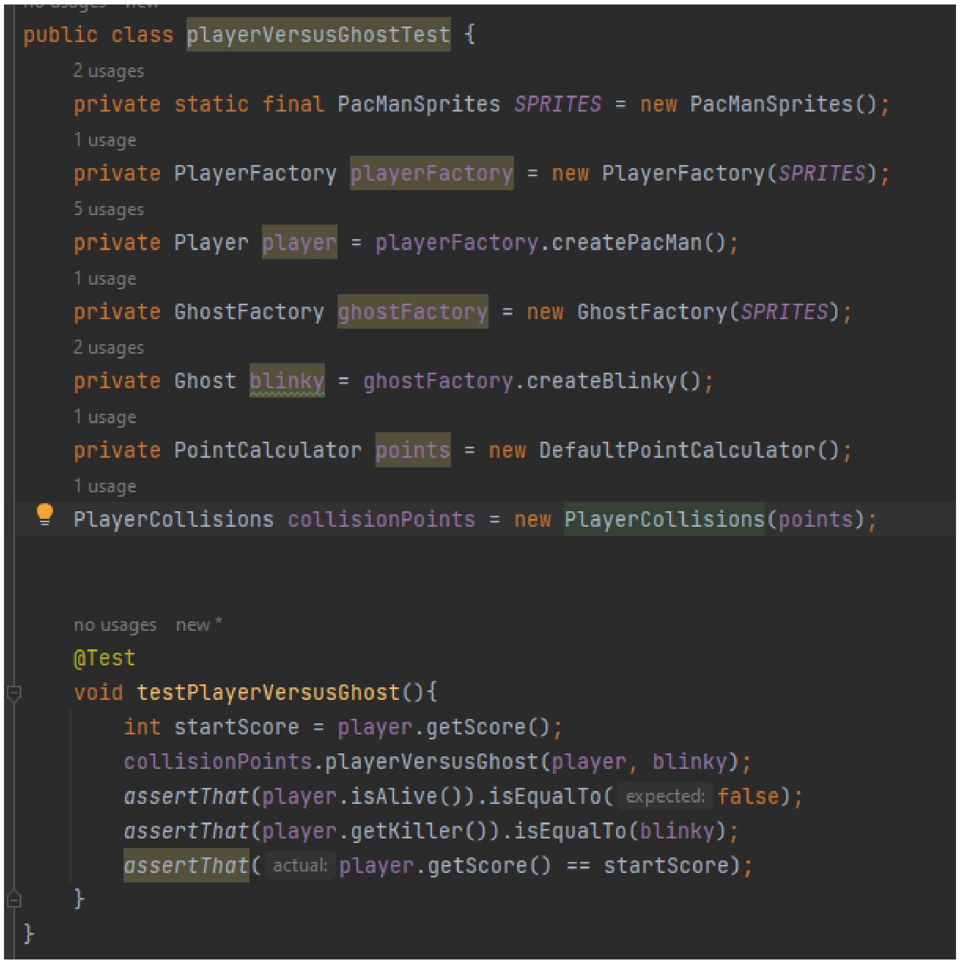
Element	Class, %	Method, %	Line, %
nl	16% (18/110)	9% (60/624)	8% (190/2306)
tudelft	16% (18/110)	9% (60/624)	8% (190/2306)
jpacman	16% (18/110)	9% (60/624)	8% (190/2306)
board	20% (4/20)	9% (10/106)	9% (28/282)
Board	0% (0/1)	0% (0/7)	0% (0/17)
BoardFactory	0% (0/3)	0% (0/11)	0% (0/27)
BoardFactoryTest	0% (0/1)	0% (0/6)	0% (0/18)
BoardTest	0% (0/1)	0% (0/3)	0% (0/3)
Direction	100% (1/1)	75% (3/4)	90% (10/11)
Square	0% (0/1)	0% (0/8)	0% (0/23)
SquareTest	0% (0/1)	0% (0/4)	0% (0/13)
Unit	100% (1/1)	20% (2/10)	13% (4/29)
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)
CollisionInteractionMap	0% (0/2)	0% (0/9)	0% (0/41)
CollisionMap	100% (0/0)	100% (0/0)	100% (0/0)
DefaultPlayerInteractionMap	0% (0/1)	0% (0/5)	0% (0/13)
Level	0% (0/2)	0% (0/17)	0% (0/113)
LevelFactory	0% (0/2)	0% (0/7)	0% (0/27)
LevelTest	0% (0/1)	0% (0/9)	0% (0/30)
MapParser	0% (0/1)	0% (0/10)	0% (0/71)
Pellet	0% (0/1)	0% (0/3)	0% (0/5)
Player	100% (1/1)	25% (2/8)	33% (8/24)
PlayerCollisions	0% (0/1)	0% (0/7)	0% (0/21)
PlayerFactory	100% (1/1)	100% (3/3)	100% (5/5)
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)
AnimatedSprite	100% (1/1)	36% (4/11)	34% (15/44)
EmptySprite	100% (1/1)	0% (0/4)	20% (1/5)
ImageSprite	100% (1/1)	85% (6/7)	76% (13/17)
PacManSprites	100% (1/1)	55% (5/9)	68% (17/25)
Sprite	100% (0/0)	100% (0/0)	100% (0/0)
SpriteStore	100% (1/1)	100% (5/5)	95% (22/23)
SpriteTest	0% (0/1)	0% (0/9)	0% (0/16)
ui	0% (0/12)	0% (0/62)	0% (0/254)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)

Task 2.1:

Test 1 (playerVersusGhost):



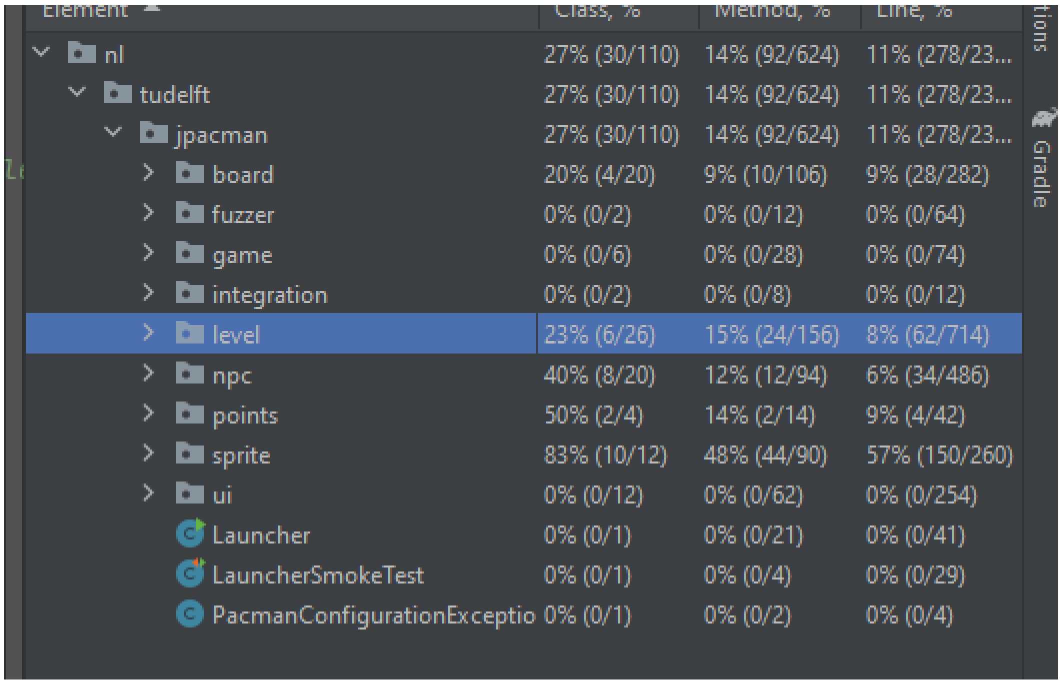
Element	Class, %	Method, %	Line, %
nl	27% (30/110)	14% (90/624)	11% (274/23...
tudelft	27% (30/110)	14% (90/624)	11% (274/23...
jpacman	27% (30/110)	14% (90/624)	11% (274/23...
board	20% (4/20)	9% (10/106)	9% (28/282)
fuzzer	0% (0/2)	0% (0/12)	0% (0/64)
game	0% (0/6)	0% (0/28)	0% (0/74)
integration	0% (0/2)	0% (0/8)	0% (0/12)
level	23% (6/26)	14% (22/156)	8% (58/714)
npc	40% (8/20)	12% (12/94)	6% (34/486)
points	50% (2/4)	14% (2/14)	9% (4/42)
sprite	83% (10/12)	48% (44/90)	57% (150/260)
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)
PacmanConfigurationExceptio	0% (0/1)	0% (0/2)	0% (0/4)



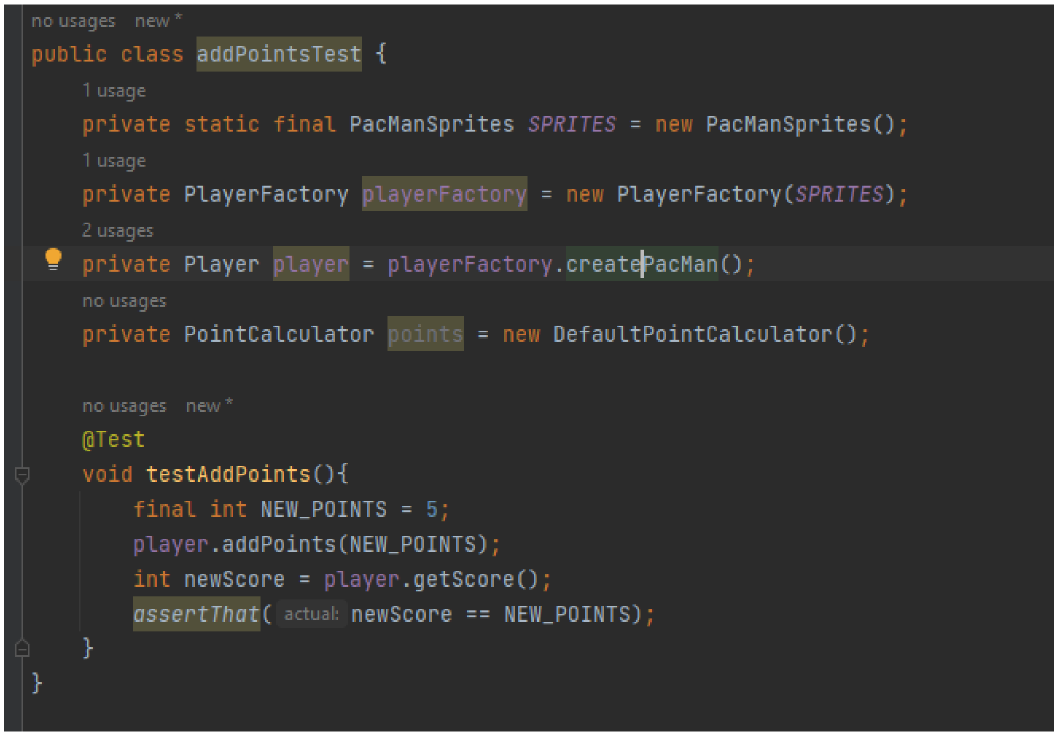
```java
public class playerVersusGhostTest {
    2 usages
    private static final PacManSprites SPRITES = new PacManSprites();
    1 usage
    private PlayerFactory playerFactory = new PlayerFactory(SPRITES);
    5 usages
    private Player player = playerFactory.createPacMan();
    1 usage
    private GhostFactory ghostFactory = new GhostFactory(SPRITES);
    2 usages
    private Ghost blinky = ghostFactory.createBlinky();
    1 usage
    private PointCalculator points = new DefaultPointCalculator();
    1 usage
    PlayerCollisions collisionPoints = new PlayerCollisions(points);


    no usages   new *
    @Test
    void testPlayerVersusGhost(){
        int startScore = player.getScore();
        collisionPoints.playerVersusGhost(player, blinky);
        assertThat(player.isAlive()).isEqualTo( expected: false);
        assertThat(player.getKiller()).isEqualTo(blinky);
        assertThat( actual: player.getScore() == startScore);
    }
}
```

Test 2 (addPoints):



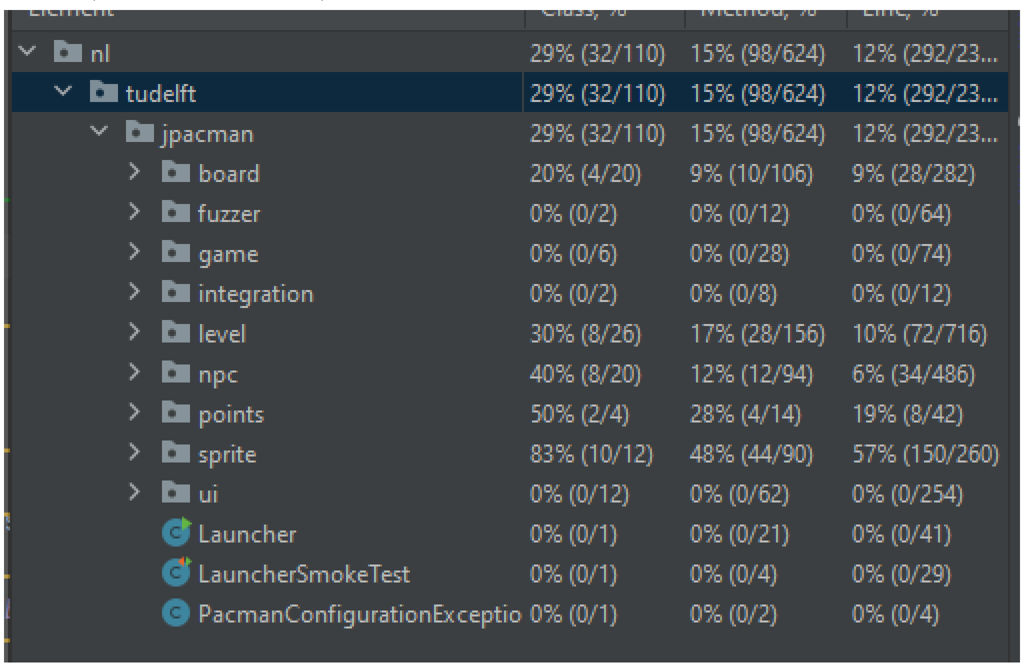
Element	Class, %	Method, %	Line, %
nl	27% (30/110)	14% (92/624)	11% (278/23...
tudelft	27% (30/110)	14% (92/624)	11% (278/23...
jpacman	27% (30/110)	14% (92/624)	11% (278/23...
board	20% (4/20)	9% (10/106)	9% (28/282)
fuzzer	0% (0/2)	0% (0/12)	0% (0/64)
game	0% (0/6)	0% (0/28)	0% (0/74)
integration	0% (0/2)	0% (0/8)	0% (0/12)
level	23% (6/26)	15% (24/156)	8% (62/714)
npc	40% (8/20)	12% (12/94)	6% (34/486)
points	50% (2/4)	14% (2/14)	9% (4/42)
sprite	83% (10/12)	48% (44/90)	57% (150/260)
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)
PacmanConfigurationExceptio	0% (0/1)	0% (0/2)	0% (0/4)



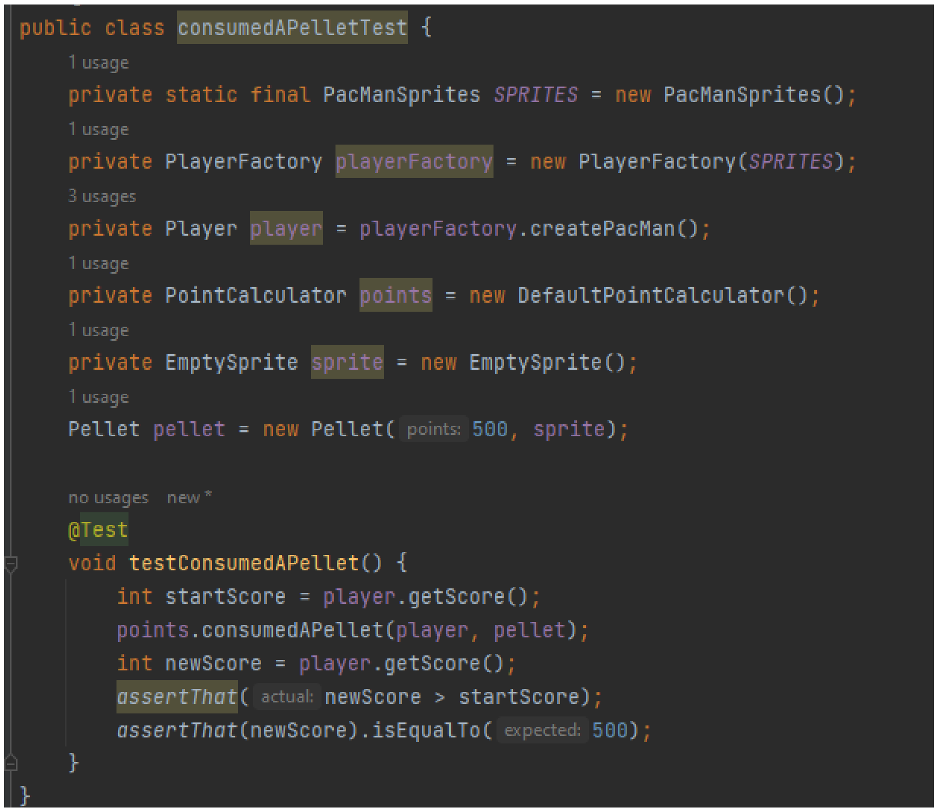
```
no usages   new *
public class addPointsTest {
    1 usage
    private static final PacManSprites SPRITES = new PacManSprites();
    1 usage
    private PlayerFactory playerFactory = new PlayerFactory(SPRITES);
    2 usages
    private Player player = playerFactory.createPacMan();
    no usages
    private PointCalculator points = new DefaultPointCalculator();

    no usages   new *
    @Test
    void testAddPoints(){
        final int NEW_POINTS = 5;
        player.addPoints(NEW_POINTS);
        int newScore = player.getScore();
        assertThat( actual: newScore == NEW_POINTS);
    }
}
```

Test 3 (consumedAPellet):



Element	Class, %	Method, %	Line, %
nl	29% (32/110)	15% (98/624)	12% (292/23...
tudelft	29% (32/110)	15% (98/624)	12% (292/23...
jpacman	29% (32/110)	15% (98/624)	12% (292/23...
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	30% (8/26)	17% (28/156)	10% (72/716)
npc	40% (8/20)	12% (12/94)	6% (34/486)
points	50% (2/4)	28% (4/14)	19% (8/42)
sprite	83% (10/12)	48% (44/90)	57% (150/260)
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)
PacmanConfigurationExceptio	0% (0/1)	0% (0/2)	0% (0/4)



```
public class consumedAPelletTest {
    1 usage
    private static final PacManSprites SPRITES = new PacManSprites();
    1 usage
    private PlayerFactory playerFactory = new PlayerFactory(SPRITES);
    3 usages
    private Player player = playerFactory.createPacMan();
    1 usage
    private PointCalculator points = new DefaultPointCalculator();
    1 usage
    private EmptySprite sprite = new EmptySprite();
    1 usage
    Pellet pellet = new Pellet( points: 500, sprite);

    no usages   new *
    @Test
    void testConsumedAPellet() {
        int startScore = player.getScore();
        points.consumedAPellet(player, pellet);
        int newScore = player.getScore();
        assertThat( actual: newScore > startScore);
        assertThat(newScore).isEqualTo( expected: 500);
    }
}
```

Task 3

The results that I got from JaCoCo were significantly higher than that of intelliJ. I believe it’s because JaCoCo can see the entire repository and understand what has been tested directly or indirectly, whereas intelliJ can only notice what lines of code have been directly tested by a unit test. I did find JaCoCo to be nicely formatted but difficult to understand, it isn’t very readable; in my opinion intelliJ is more user friendly. IntelliJ is very direct and shows what lines have been tested, as well as it’s very simple to run within the application. Additionally, I find simplicity better in this case. Numbers are all I need.