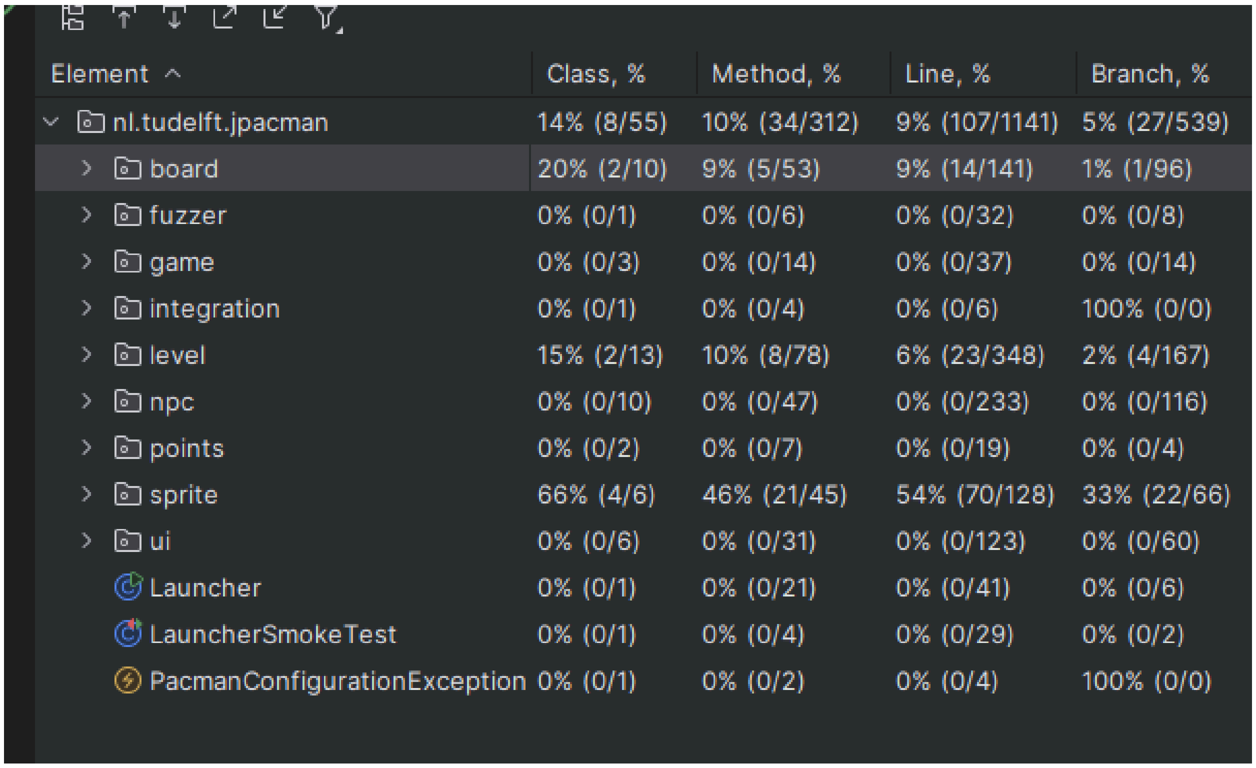
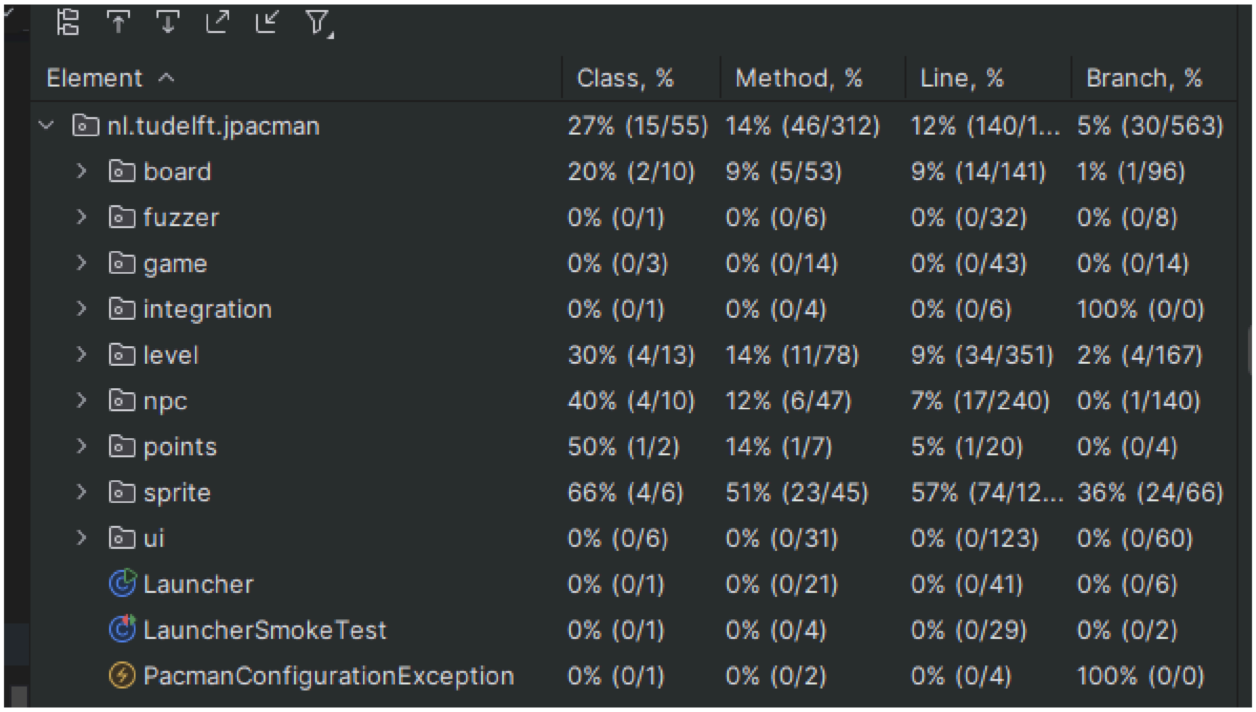
Before any changes:



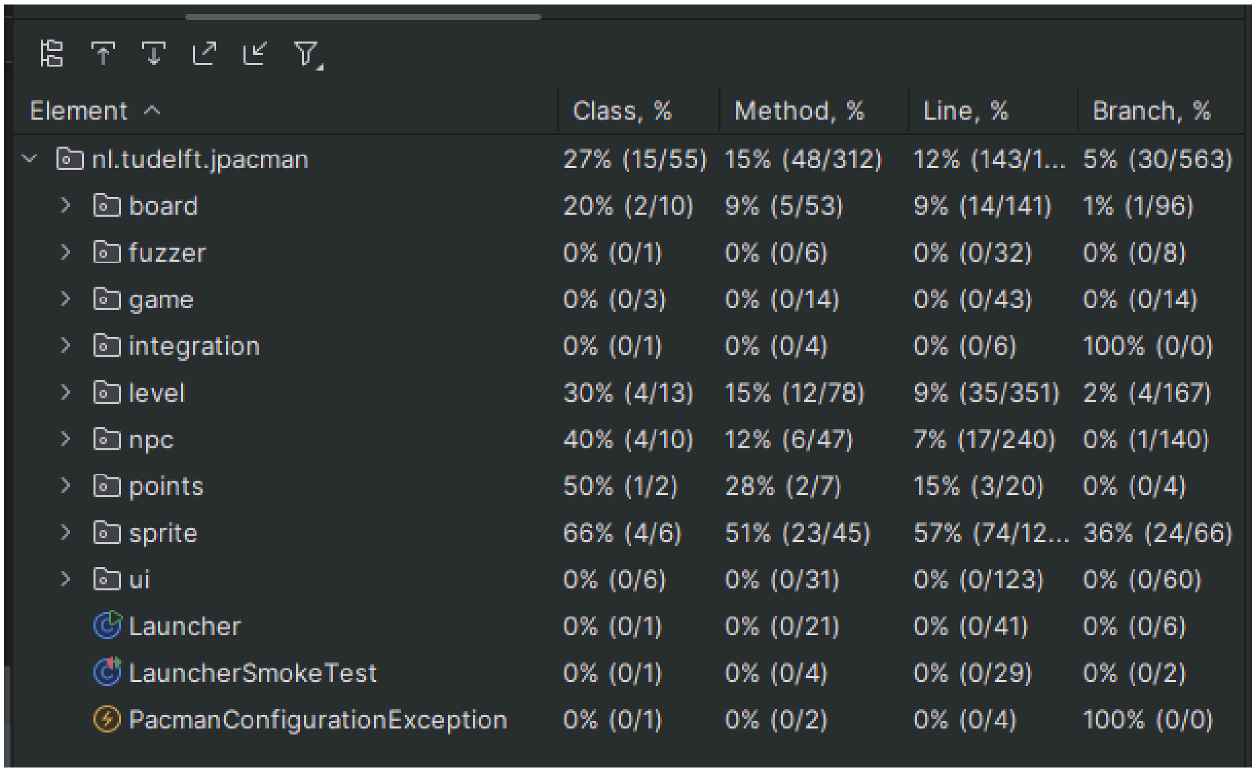
Element	Class, %	Method, %	Line, %	Branch, %
nl.tudelft.jpacman	14% (8/55)	10% (34/312)	9% (107/1141)	5% (27/539)
board	20% (2/10)	9% (5/53)	9% (14/141)	1% (1/96)
fuzzer	0% (0/1)	0% (0/6)	0% (0/32)	0% (0/8)
game	0% (0/3)	0% (0/14)	0% (0/37)	0% (0/14)
integration	0% (0/1)	0% (0/4)	0% (0/6)	100% (0/0)
level	15% (2/13)	10% (8/78)	6% (23/348)	2% (4/167)
npc	0% (0/10)	0% (0/47)	0% (0/233)	0% (0/116)
points	0% (0/2)	0% (0/7)	0% (0/19)	0% (0/4)
sprite	66% (4/6)	46% (21/45)	54% (70/128)	33% (22/66)
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)

After testCollidedWithGhost:



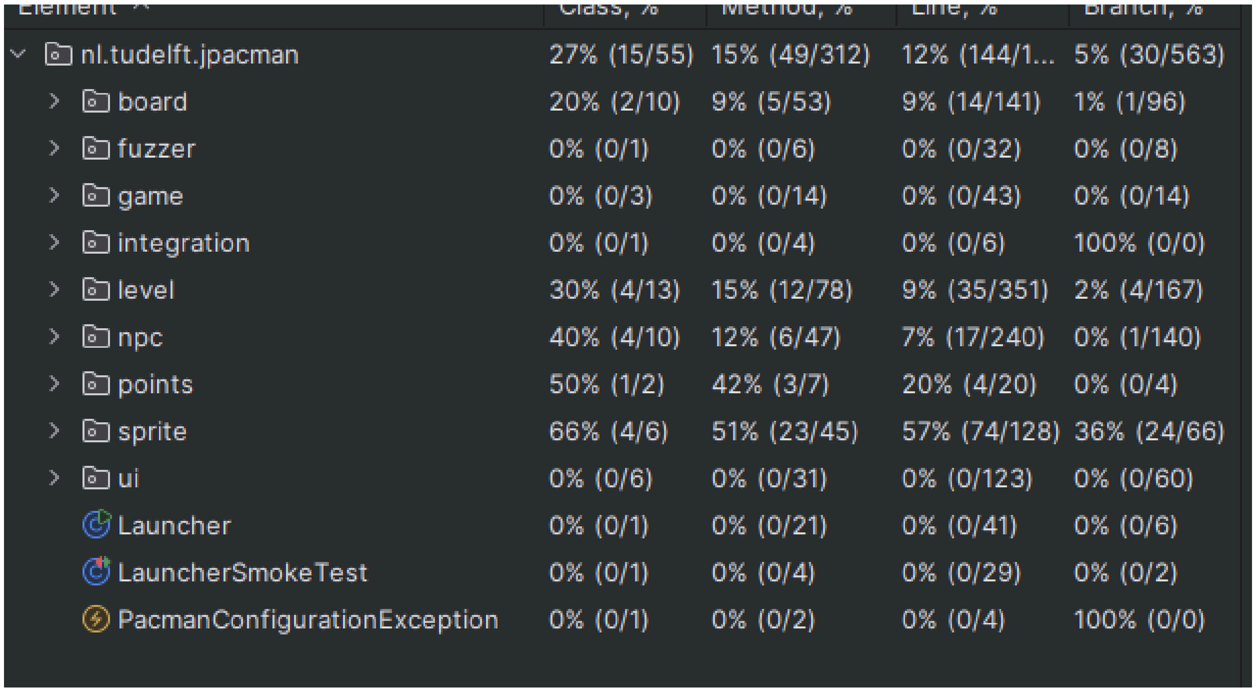
Element	Class, %	Method, %	Line, %	Branch, %
nl.tudelft.jpacman	27% (15/55)	14% (46/312)	12% (140/1...	5% (30/563)
board	20% (2/10)	9% (5/53)	9% (14/141)	1% (1/96)
fuzzer	0% (0/1)	0% (0/6)	0% (0/32)	0% (0/8)
game	0% (0/3)	0% (0/14)	0% (0/43)	0% (0/14)
integration	0% (0/1)	0% (0/4)	0% (0/6)	100% (0/0)
level	30% (4/13)	14% (11/78)	9% (34/351)	2% (4/167)
npc	40% (4/10)	12% (6/47)	7% (17/240)	0% (1/140)
points	50% (1/2)	14% (1/7)	5% (1/20)	0% (0/4)
sprite	66% (4/6)	51% (23/45)	57% (74/12...	36% (24/66)
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)

After testConsumedPellet:



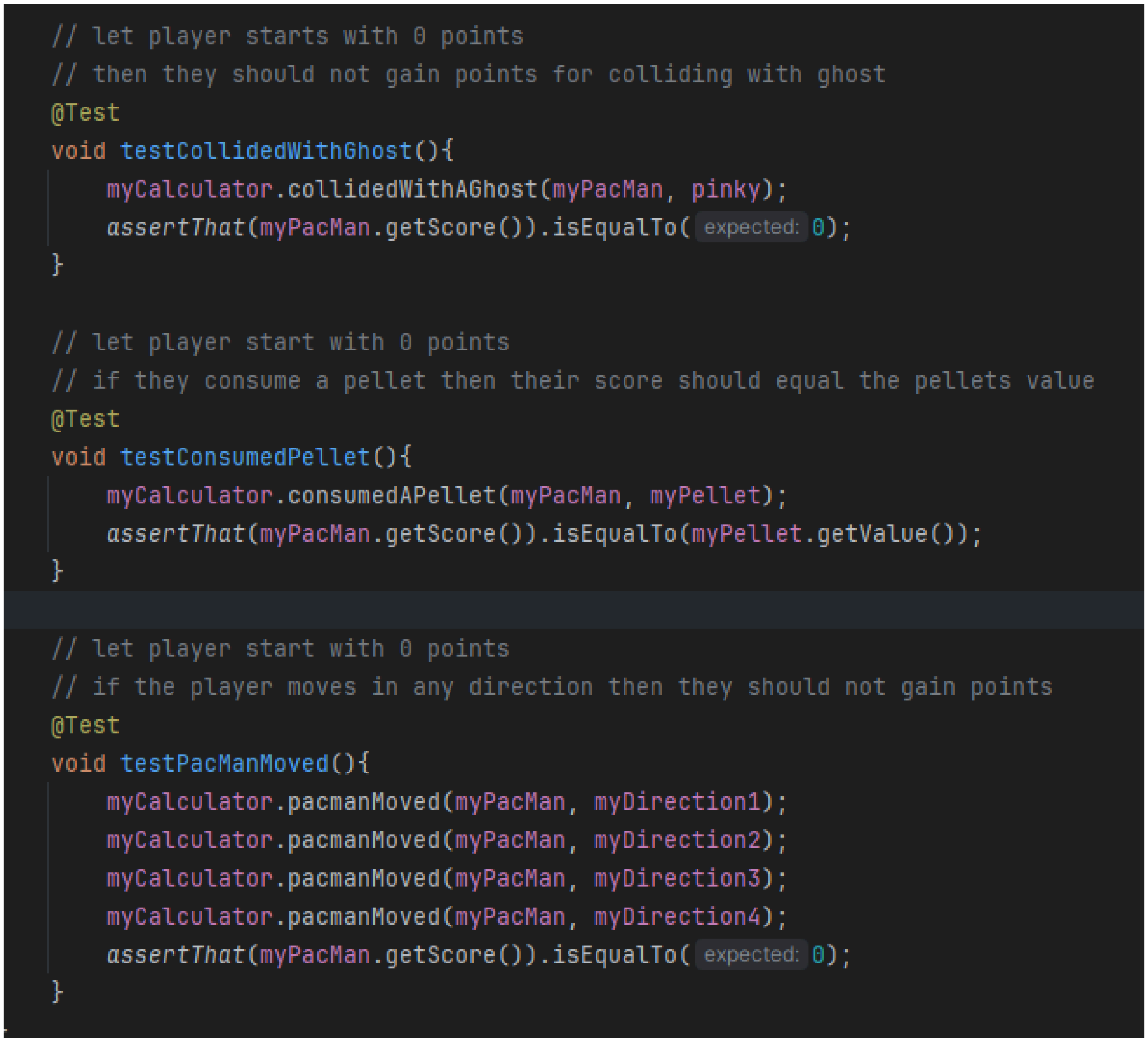
Element	Class, %	Method, %	Line, %	Branch, %
nl.tudelft.jpacman	27% (15/55)	15% (48/312)	12% (143/1...	5% (30/563)
board	20% (2/10)	9% (5/53)	9% (14/141)	1% (1/96)
fuzzer	0% (0/1)	0% (0/6)	0% (0/32)	0% (0/8)
game	0% (0/3)	0% (0/14)	0% (0/43)	0% (0/14)
integration	0% (0/1)	0% (0/4)	0% (0/6)	100% (0/0)
level	30% (4/13)	15% (12/78)	9% (35/351)	2% (4/167)
npc	40% (4/10)	12% (6/47)	7% (17/240)	0% (1/140)
points	50% (1/2)	28% (2/7)	15% (3/20)	0% (0/4)
sprite	66% (4/6)	51% (23/45)	57% (74/12...	36% (24/66)
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)

After testPacManMoved:



Element	Class, %	Method, %	Line, %	Branch, %
nl.tudelft.jpacman	27% (15/55)	15% (49/312)	12% (144/1...	5% (30/563)
board	20% (2/10)	9% (5/53)	9% (14/141)	1% (1/96)
fuzzer	0% (0/1)	0% (0/6)	0% (0/32)	0% (0/8)
game	0% (0/3)	0% (0/14)	0% (0/43)	0% (0/14)
integration	0% (0/1)	0% (0/4)	0% (0/6)	100% (0/0)
level	30% (4/13)	15% (12/78)	9% (35/351)	2% (4/167)
npc	40% (4/10)	12% (6/47)	7% (17/240)	0% (1/140)
points	50% (1/2)	42% (3/7)	20% (4/20)	0% (0/4)
sprite	66% (4/6)	51% (23/45)	57% (74/128)	36% (24/66)
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)

Code:



```
// let player starts with 0 points
// then they should not gain points for colliding with ghost
@Test
void testCollidedWithGhost(){
    myCalculator.collidedWithAGhost(myPacMan, pinky);
    assertThat(myPacMan.getScore()).isEqualTo( expected: 0);
}

// let player start with 0 points
// if they consume a pellet then their score should equal the pellets value
@Test
void testConsumedPellet(){
    myCalculator.consumedAPellet(myPacMan, myPellet);
    assertThat(myPacMan.getScore()).isEqualTo(myPellet.getValue());
}

// let player start with 0 points
// if the player moves in any direction then they should not gain points
@Test
void testPacManMoved(){
    myCalculator.pacmanMoved(myPacMan, myDirection1);
    myCalculator.pacmanMoved(myPacMan, myDirection2);
    myCalculator.pacmanMoved(myPacMan, myDirection3);
    myCalculator.pacmanMoved(myPacMan, myDirection4);
    assertThat(myPacMan.getScore()).isEqualTo( expected: 0);
}
```

Are the coverage results from JaCoCo similar to the ones you got from IntelliJ in the last task? Why so or why not?
The results are very similar, but JaCoCo seemed to not pick up on some abstract classes such as board.Unit or board.Square.
Did you find helpful the source code visualization from JaCoCo on uncovered branches?
I don't think that the visualization from JaCoCo was any better than the visualization from IntelliJ. It may have been prettier, but I don't think it helped me determine where any test cases should have gone over the way IntelliJ does it.
Which visualization did you prefer and why? IntelliJ's coverage window or JaCoCo's report?
I preferred the IntelliJ's coverage window because it was built into the code editor and was able to show me where there needed to be test coverage in the same application that I am writing the

tests with. Which for me is a plus considering that I don’t have to switch tabs to see where tests need to be.