













Link to my github repository <https://github.com/sonicspeed123/CS472>

Element ^	Class, %	Method, %	Line, %
✓  nl	3% (2/55)	1% (5/312)	1% (14/1137)
✓  tudelft	3% (2/55)	1% (5/312)	1% (14/1137)
✓  jpacman	3% (2/55)	1% (5/312)	1% (14/1137)
>  board	20% (2/10)	9% (5/53)	9% (14/141)
>  fuzzer	0% (0/1)	0% (0/6)	0% (0/32)
>  game	0% (0/3)	0% (0/14)	0% (0/37)
>  integration	0% (0/1)	0% (0/4)	0% (0/6)
>  level	0% (0/13)	0% (0/78)	0% (0/345)
>  npc	0% (0/10)	0% (0/47)	0% (0/237)
>  points	0% (0/2)	0% (0/7)	0% (0/19)
>  sprite	0% (0/6)	0% (0/45)	0% (0/119)
>  ui	0% (0/6)	0% (0/31)	0% (0/127)
© Launcher	0% (0/1)	0% (0/21)	0% (0/41)
© LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)
⚡ PacmanConfigurationE>	0% (0/1)	0% (0/2)	0% (0/4)

This is the test coverage at the start of the assignment.

Considering that there are little to no tests covering anything, no, this is not good coverage.

Coverage jpacman [test] x			
Element ^	Class, %	Method, %	Line, %
nl	16% (9/55)	9% (30/312)	8% (95/1153)
tudelft	16% (9/55)	9% (30/312)	8% (95/1153)
jpacman	16% (9/55)	9% (30/312)	8% (95/1153)
board	20% (2/10)	9% (5/53)	9% (14/141)
fuzzer	0% (0/1)	0% (0/6)	0% (0/32)
game	0% (0/3)	0% (0/14)	0% (0/37)
integration	0% (0/1)	0% (0/4)	0% (0/6)
level	15% (2/13)	6% (5/78)	3% (13/350)
CollisionInteractionM	0% (0/2)	0% (0/9)	0% (0/41)
CollisionMap	100% (0/0)	100% (0/0)	100% (0/0)
DefaultPlayerInterac	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% (0/10)	0% (0/47)	0% (0/237)
points	0% (0/2)	0% (0/7)	0% (0/19)
sprite	83% (5/6)	44% (20/45)	52% (68/130)
ui	0% (0/6)	0% (0/31)	0% (0/127)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)
LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)

This is coverage after adding assert for the Player class. Notice how the level package has gone up to 15%











This also before adding tests for the GhostFactory class.

Element ^	Class, %	Method, %	Line, %
▼ nl	29% (16/55)	14% (46/312)	11% (133/11...
▼ tudelft	29% (16/55)	14% (46/312)	11% (133/11...
▼ jpacman	29% (16/55)	14% (46/312)	11% (133/11...
> board	20% (2/10)	9% (5/53)	9% (14/141)
> fuzzer	0% (0/1)	0% (0/6)	0% (0/32)
> game	0% (0/3)	0% (0/14)	0% (0/37)
> integration	0% (0/1)	0% (0/4)	0% (0/6)
> level	15% (2/13)	6% (5/78)	3% (13/350)
> npc	70% (7/10)	31% (15/47)	14% (35/243)
> points	0% (0/2)	0% (0/7)	0% (0/19)
> sprite	83% (5/6)	46% (21/45)	54% (71/130)
> ui	0% (0/6)	0% (0/31)	0% (0/127)
Ⓢ Launcher	0% (0/1)	0% (0/21)	0% (0/41)
Ⓢ LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)
⚡ PacmanConfigurationE	0% (0/1)	0% (0/2)	0% (0/4)

▼ npc	70% (7/10)	31% (15/47)	14% (35/243)
▼ ghost	66% (6/9)	32% (14/43)	12% (30/235)
Ⓢ Blinky	100% (1/1)	50% (2/4)	13% (3/22)
Ⓢ Clyde	100% (1/1)	50% (2/4)	29% (9/31)
Ⓔ GhostColor	100% (1/1)	100% (1/1)	100% (5/5)
Ⓢ GhostFactory	100% (1/1)	100% (5/5)	100% (7/7)
Ⓢ Inky	100% (1/1)	40% (2/5)	9% (3/32)
Ⓢ Navigation	0% (0/2)	0% (0/11)	0% (0/60)
Ⓢ NavigationTest	0% (0/1)	0% (0/9)	0% (0/56)
Ⓢ Pinky	100% (1/1)	50% (2/4)	13% (3/22)
Ⓢ Ghost	100% (1/1)	25% (1/4)	62% (5/8)

This is the coverage after adding 4 asserts for ghost creation in the GhostFactory class, note the NPC package now has 70%

Player

Element	Missed Instructions	Cov.	Missed Branches	Cov.
● setAlive(boolean)		61%		50%
● getSprite()		76%		50%
● getKiller()		0%		n/a
● Player(Map, AnimatedSprite)		100%		n/a
● addPoints(int)		100%		n/a
● setKiller(Unit)		100%		n/a
● isAlive()		100%		n/a
● getScore()		100%		n/a

```
public Sprite getSprite() {  
    if (isAlive()) {  
        return sprites.get(getDirection());  
    }  
    return deathSprite;  
}
```

The results from JaCoCo are similar to the ones in IntelliJ because they both give a percentage of how many parts of the code have been tested, with JaCoCo being a little more detailed with the branches.

I think that the visualization from JaCoCo is very helpful for me to better understand just which pieces of code need to be tested.

I think that the report from JaCoCo is better when it comes to the finer details of setting up tests for coverage. It literally shows you what lines of code have/haven't been tested, as well as what code still needs to be looked at.