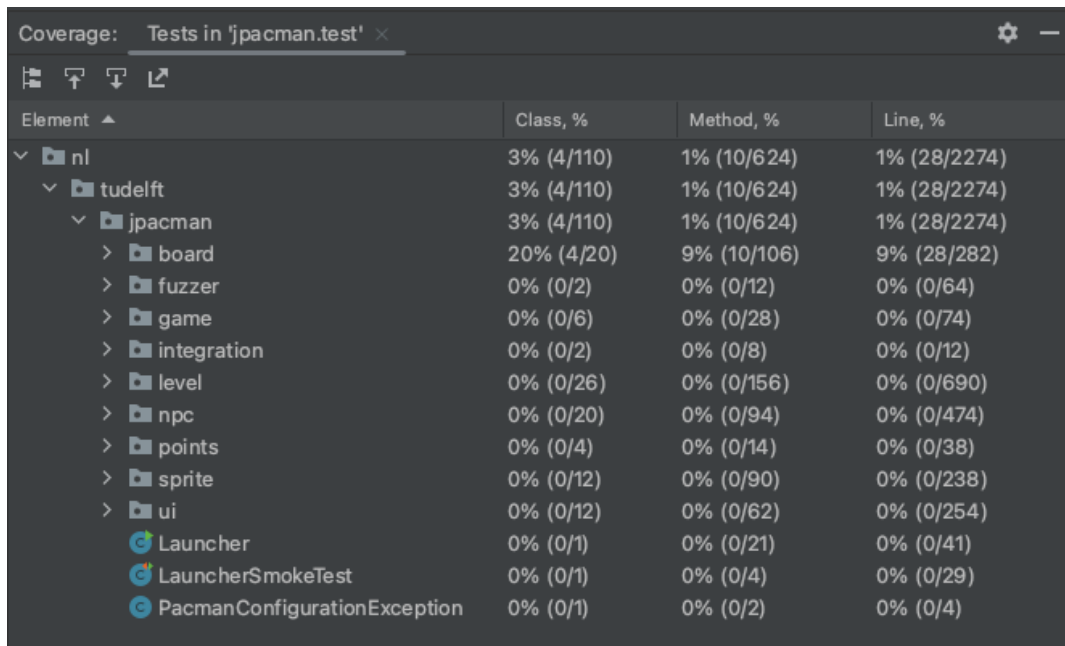


Link to repository: <https://github.com/ks-moss/472-2023-G3.git>

Task 1

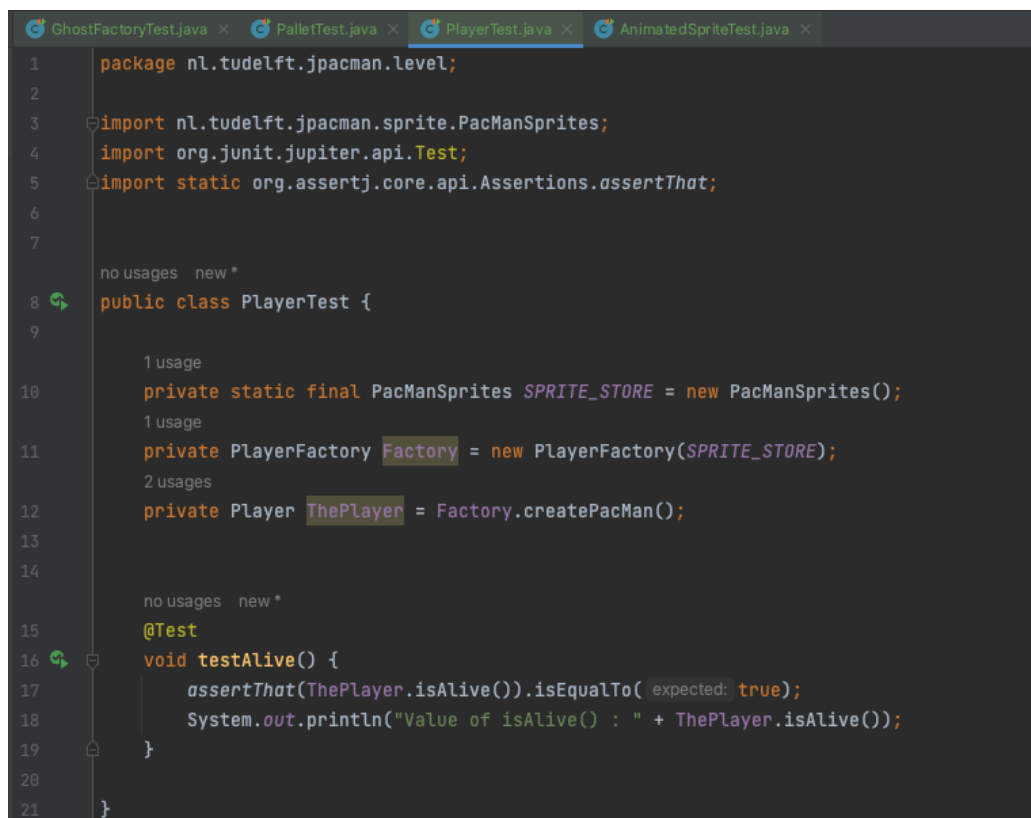
Coverage before testing



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)
LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)
PacmanConfigurationException	0% (0/1)	0% (0/2)	0% (0/4)

Is the coverage good enough? No, it's not good enough.

Task 2



```
1 package nl.tudelft.jpacman.level;
2
3 import nl.tudelft.jpacman.sprite.PacManSprites;
4 import org.junit.jupiter.api.Test;
5 import static org.assertj.core.api.Assertions.assertThat;
6
7
8 public class PlayerTest {
9
10     private static final PacManSprites SPRITE_STORE = new PacManSprites();
11     private PlayerFactory Factory = new PlayerFactory(SPRITE_STORE);
12     private Player ThePlayer = Factory.createPacMan();
13
14
15     @Test
16     void testAlive() {
17         assertThat(ThePlayer.isAlive()).isEqualTo(expected: true);
18         System.out.println("Value of isAlive() : " + ThePlayer.isAlive());
19     }
20
21 }
```

Coverage: Tests in 'jpacman.test' x

Element	Class, %	Method, %	Line, %
nl	16% (18/112)	9% (60/634)	8% (190/2330)
tudelft	16% (18/112)	9% (60/634)	8% (190/2330)
jpacman	16% (18/112)	9% (60/634)	8% (190/2330)
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% (0/22)	0% (0/104)	0% (0/498)
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)

Task 2.1

src/main/java/nl/tudelft/jpacman/level/pallet.java

```

GhostFactoryTest.java x PalletTest.java x PlayerTest.java x AnimatedSpriteTest.java x
1  package nl.tudelft.jpacman.level;
2
3  import nl.tudelft.jpacman.sprite.PacManSprites;
4  import org.junit.jupiter.api.Test;
5
6  no usages new *
7  public class PalletTest {
8      1 usage
9      private static final PacManSprites SPRITE_STORE = new PacManSprites();
10     2 usages
11     private PlayerFactory Factory = new PlayerFactory(SPRITE_STORE);
12     no usages
13     private Player ThePlayer = Factory.createPacMan();
14
15     no usages new *
16     @Test
17     void testGetValue() { System.out.println("Value of getSprite() : " + Factory.createPacMan()); }
18 }

```

Coverage: Tests in 'jpacman.test' ×			
Element ▲	Class, %	Method, %	Line, %
▼ nl	16% (18/112)	10% (64/634)	8% (196/2330)
▼ tudelft	16% (18/112)	10% (64/634)	8% (196/2330)
▼ jpacman	16% (18/112)	10% (64/634)	8% (196/2330)
> board	20% (4/20)	11% (12/106)	10% (30/282)
> 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)	7% (12/156)	4% (30/700)
> npc	0% (0/22)	0% (0/104)	0% (0/498)
> 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)

src/main/java/nl/tudelft/jpacman/npc/ghost/GhostFactory.java

```

GhostFactoryTest.java × PalletTest.java × PlayerTest.java × AnimatedSpriteTest.java ×
1  package nl.tudelft.jpacman.ghost;
2
3  import nl.tudelft.jpacman.level.PlayerFactory;
4  import nl.tudelft.jpacman.sprite.PacManSprites;
5  import org.junit.jupiter.api.Test;
6
7  no usages new *
8  public class GhostFactoryTest {
9
10     1 usage
11     private static final PacManSprites SPRITE_STORE = new PacManSprites();
12     1 usage
13     private PlayerFactory Factory = new PlayerFactory(SPRITE_STORE);
14
15     no usages new *
16     @Test
17     void testGetSprites() {
18         System.out.println("Value of createPacMan() :" + Factory.createPacMan());
19     }
20 }

```

Coverage: Tests in 'jpacman.test' x

Element	Class, %	Method, %	Line, %
nl	16% (18/110)	10% (64/624)	8% (196/2306)
tudelft	16% (18/110)	10% (64/624)	8% (196/2306)
jpacman	16% (18/110)	10% (64/624)	8% (196/2306)
board	20% (4/20)	11% (12/106)	10% (30/282)
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)	7% (12/156)	4% (30/700)
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)

src/main/java/nl/tudelft/jpacman/sprite/AnimatedSpriteTest.java

```

1 package nl.tudelft.jpacman.sprite;
2
3 import org.junit.jupiter.api.Test;
4
5 no usages new *
6 public class AnimatedSpriteTest {
7
8     1 usage
9     private static final Sprite SPRITE = new EmptySprite();
10
11     no usages new *
12     @Test
13     void testAnimatedSprite() { System.out.println("Value of getWidth() : " + SPRITE.getWidth()); }
14 }

```

Coverage: Tests in 'jpacman.test' x

Element	Class, %	Method, %	Line, %
nl	16% (18/110)	9% (62/624)	8% (192/2306)
tudelft	16% (18/110)	9% (62/624)	8% (192/2306)
jpacman	16% (18/110)	9% (62/624)	8% (192/2306)
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)	46% (42/90)	53% (138/260)
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)

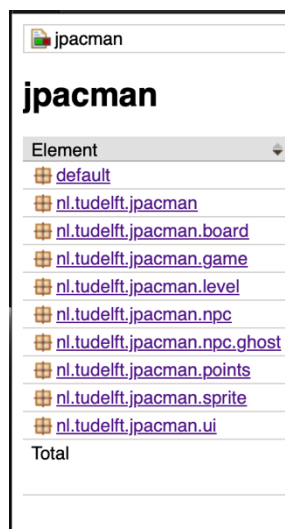
Task 3

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 obtained from the JaCoCo coverage tool differ from those previously obtained through IntelliJ in a prior task. The coverage results within IntelliJ are contingent upon various factors such as the type of coverage tool used, its configuration, the build process utilized, and the extent of test coverage within the codebase.

Did you find helpful the source code visualization from JaCoCo on uncovered branches?

Affirmative, I discovered that the JaCoCo coverage tool provides a visualization of the source code, specifically highlighting any uncovered branches. This information can be readily accessed as demonstrated in the following illustration:



Which visualization did you prefer and why? IntelliJ's coverage window or JaCoCo's report?

I found the JaCoCo coverage report to be preferable due to its convenient accessibility of information regarding uncovered branches, as well as its well-structured format. The report also employs a clear and effective color-coding scheme, utilizing red and white to display the coverage percentage as demonstrated below:

jpacman												
Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
default		0%		0%	12	12	21	21	5	5	1	1
nl.tudelft.jpacman		69%		25%	12	30	18	52	6	24	1	2
nl.tudelft.jpacman.board		86%		58%	44	93	2	110	0	40	0	7
nl.tudelft.jpacman.game		87%		60%	10	24	4	45	2	14	0	3
nl.tudelft.jpacman.level		67%		57%	74	155	104	344	21	69	4	12
nl.tudelft.jpacman.npc		100%		n/a	0	4	0	8	0	4	0	1
nl.tudelft.jpacman.npc.ghost		71%		55%	56	105	43	181	5	34	0	8
nl.tudelft.jpacman.points		60%		75%	1	11	5	21	0	9	0	2
nl.tudelft.jpacman.sprite		87%		59%	29	70	10	113	4	38	0	5
nl.tudelft.jpacman.ui		77%		47%	54	86	21	144	7	31	0	6
Total	1,211 of 4,694	74%	293 of 637	54%	292	590	228	1,039	50	268	6	47

Created with JaCoCo 0.8.3.201901230119