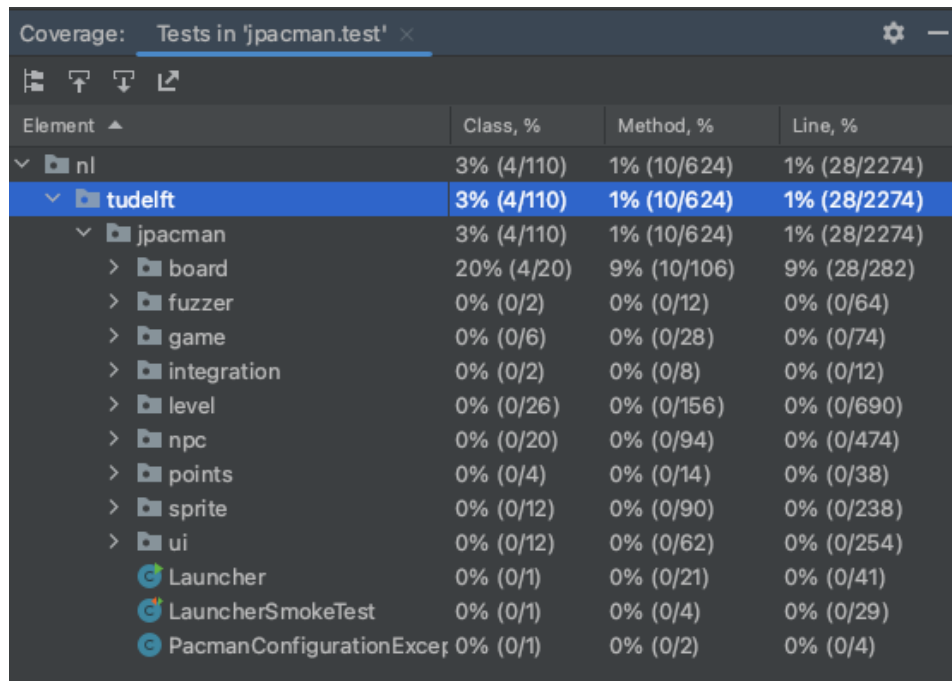


Link to repository: <https://github.com/ks-moss/472-2023-G3.git>

Task 1 Coverage before testing

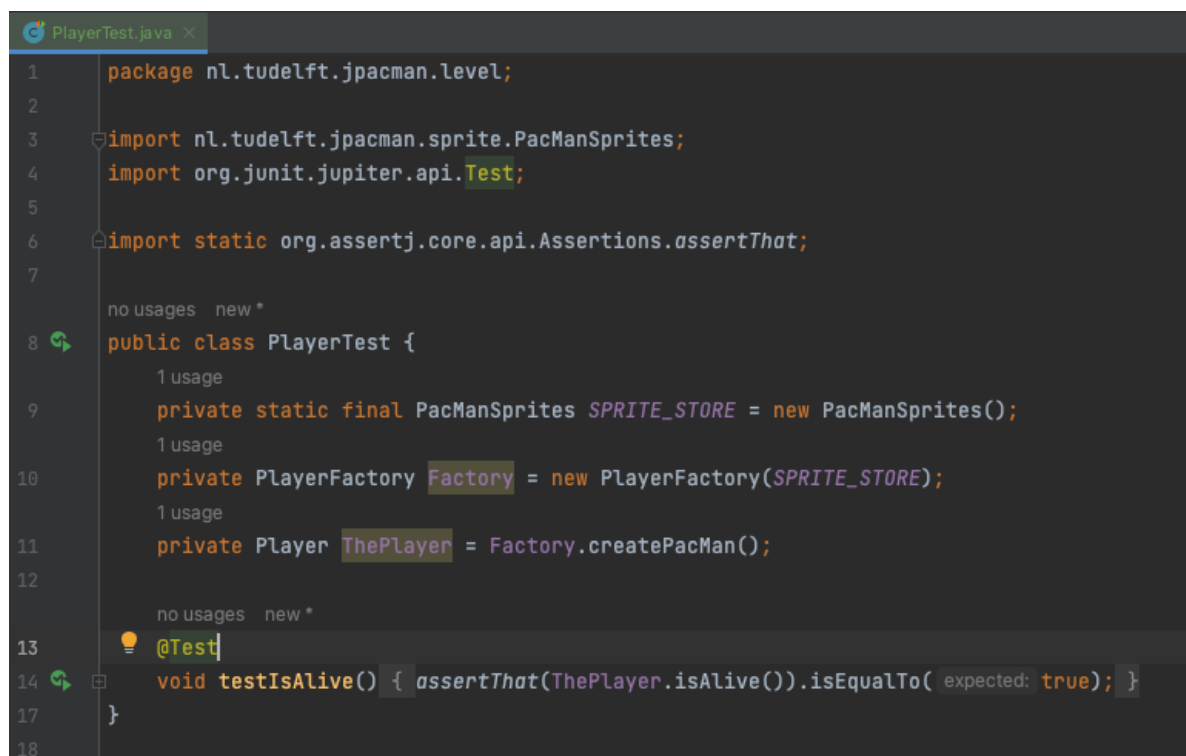


The screenshot shows the 'Coverage' window in IntelliJ IDEA, titled 'Tests in 'jpacman.test''. It displays a tree view of the project structure with coverage percentages for classes, methods, and lines. The 'tudelft' package is highlighted, showing 3% class coverage (4/110), 1% method coverage (10/624), and 1% line coverage (28/2274). The 'jpacman' package is expanded, showing 3% class coverage (4/110), 1% method coverage (10/624), and 1% line coverage (28/2274). The 'board' package is expanded, showing 20% class coverage (4/20), 9% method coverage (10/106), and 9% line coverage (28/282). The 'fuzzer' package is expanded, showing 0% class coverage (0/2), 0% method coverage (0/12), and 0% line coverage (0/64). The 'game' package is expanded, showing 0% class coverage (0/6), 0% method coverage (0/28), and 0% line coverage (0/74). The 'integration' package is expanded, showing 0% class coverage (0/2), 0% method coverage (0/8), and 0% line coverage (0/12). The 'level' package is expanded, showing 0% class coverage (0/26), 0% method coverage (0/156), and 0% line coverage (0/690). The 'npc' package is expanded, showing 0% class coverage (0/20), 0% method coverage (0/94), and 0% line coverage (0/474). The 'points' package is expanded, showing 0% class coverage (0/4), 0% method coverage (0/14), and 0% line coverage (0/38). The 'sprite' package is expanded, showing 0% class coverage (0/12), 0% method coverage (0/90), and 0% line coverage (0/238). The 'ui' package is expanded, showing 0% class coverage (0/12), 0% method coverage (0/62), and 0% line coverage (0/254). The 'Launcher' package is expanded, showing 0% class coverage (0/1), 0% method coverage (0/21), and 0% line coverage (0/41). The 'LauncherSmokeTest' package is expanded, showing 0% class coverage (0/1), 0% method coverage (0/4), and 0% line coverage (0/29). The 'PacmanConfigurationException' package is expanded, showing 0% class coverage (0/1), 0% method coverage (0/2), and 0% line coverage (0/4).

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. Let's increase the coverage.

Task 2



The screenshot shows the code editor in IntelliJ IDEA, displaying the file 'PlayerTest.java'. The code is as follows:

```
1 package nl.tudelft.jpacman.level;
2
3 import nl.tudelft.jpacman.sprite.PacManSprites;
4 import org.junit.jupiter.api.Test;
5
6 import static org.assertj.core.api.Assertions.assertThat;
7
8 no usages new *
9 public class PlayerTest {
10     1 usage
11     private static final PacManSprites SPRITE_STORE = new PacManSprites();
12     1 usage
13     private PlayerFactory Factory = new PlayerFactory(SPRITE_STORE);
14     1 usage
15     private Player ThePlayer = Factory.createPacMan();
16
17     no usages new *
18     @Test
19     void testIsAlive() { assertThat(ThePlayer.isAlive()).isEqualTo( expected: true); }
20 }
```

Coverage: Tests in 'jpacman.test' x			
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)
PacmanConfigurationExcept	0% (0/1)	0% (0/2)	0% (0/4)

Task 2.1

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

First Method

```

GhostFactoryTest.java x
1  package nl.tudelft.jpacman.npc.ghost;
2
3  import nl.tudelft.jpacman.sprite.PacManSprites;
4  import org.junit.jupiter.api.Test;
5
6  no usages new *
7  public class GhostFactoryTest {
8      1 usage
9      private static final PacManSprites SPRITE_STORE = new PacManSprites();
10     1 usage
11     GhostFactory tempGhost = new GhostFactory(SPRITE_STORE);
12
13     no usages new *
14     @Test
15     void testGhostFactory(){
16         System.out.println("Value of createBlinky() : " + tempGhost.createBlinky());
17         //System.out.println("Value of createClyde() : " + tempGhost.createClyde());
18         //System.out.println("Value of createPinky() : " + tempGhost.createPinky());
19     }
20 }

```

Coverage: Tests in 'jpacman.test' ×			
Element ▲	Class, %	Method, %	Line, %
▼ nl	23% (26/110)	11% (74/624)	9% (230/2318)
▼ tudelft	23% (26/110)	11% (74/624)	9% (230/2318)
▼ jpacman	23% (26/110)	11% (74/624)	9% (230/2318)
> 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	40% (8/20)	12% (12/94)	6% (34/486)
> 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)

Second Method

```

1  package nl.tudelft.jpacman.npc.ghost;
2
3  import nl.tudelft.jpacman.sprite.PacManSprites;
4  import org.junit.jupiter.api.Test;
5
6  no usages new *
7  public class GhostFactoryTest {
8      1 usage
9      private static final PacManSprites SPRITE_STORE = new PacManSprites();
10     2 usages
11     GhostFactory tempGhost = new GhostFactory(SPRITE_STORE);
12
13     no usages new *
14     @Test
15     void testGhostFactory(){
16         System.out.println("Value of createBlinky() : " + tempGhost.createBlinky());
17         System.out.println("Value of createClyde() : " + tempGhost.createClyde());
18         //System.out.println("Value of createPinky() : " + tempGhost.createPinky());
19     }
20 }

```

Coverage: Tests in 'jpacman.test' ×			
Element ▲	Class, %	Method, %	Line, %
▼ nl	25% (28/110)	12% (80/624)	10% (250/2318)
▼ tudelft	25% (28/110)	12% (80/624)	10% (250/2318)
▼ jpacman	25% (28/110)	12% (80/624)	10% (250/2318)
> 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	50% (10/20)	19% (18/94)	11% (54/486)
> 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)
🌀 PacmanConfigurationExce	0% (0/1)	0% (0/2)	0% (0/4)

Third Method

```

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

```

Coverage: Tests in 'jpacman.test' ×			
Element ▲	Class, %	Method, %	Line, %
▼ nl	27% (30/110)	13% (86/624)	11% (258/2318)
▼ tudelft	27% (30/110)	13% (86/624)	11% (258/2318)
▼ jpacman	27% (30/110)	13% (86/624)	11% (258/2318)
> 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	60% (12/20)	25% (24/94)	12% (62/486)
> 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)

Forth Method

```

GhostFactoryTest.java ×
1  package nl.tudelft.jpacman.npc.ghost;
2
3  import nl.tudelft.jpacman.sprite.PacManSprites;
4  import org.junit.jupiter.api.Test;
5
6  no usages new *
7  public class GhostFactoryTest {
8      1 usage
9      private static final PacManSprites SPRITE_STORE = new PacManSprites();
10     4 usages
11     GhostFactory tempGhost = new GhostFactory(SPRITE_STORE);
12
13     no usages new *
14     @Test
15     void testGhostFactory(){
16         System.out.println("Value of createBlinky() : " + tempGhost.createBlinky());
17         System.out.println("Value of createClyde() : " + tempGhost.createClyde());
18         System.out.println("Value of createPinky() : " + tempGhost.createPinky());
19         System.out.println("Value of createInky() : " + tempGhost.createInky());
20     }
21 }

```

Coverage: Tests in 'jpacman.test' x

Element ▲	Class, %	Method, %	Line, %
▼ 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)

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:


jpacman


jpacman

Element
default
nl.tudelft.jpacman
nl.tudelft.jpacman.board
nl.tudelft.jpacman.game
nl.tudelft.jpacman.level
nl.tudelft.jpacman.npc
nl.tudelft.jpacman.npc.ghost
nl.tudelft.jpacman.points
nl.tudelft.jpacman.sprite
nl.tudelft.jpacman.ui











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

 [Sessions](#)

jpacman

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
 default	<div><div></div></div>	0%	<div><div></div></div>	0%	12	12	21	21	5	5	1	1
 nl.tudelft.jpacman	<div><div></div></div>	69%	<div><div></div></div>	25%	12	30	18	52	6	24	1	2
 nl.tudelft.jpacman.board	<div><div></div></div>	86%	<div><div></div></div>	58%	44	93	2	110	0	40	0	7
 nl.tudelft.jpacman.game	<div><div></div></div>	87%	<div><div></div></div>	60%	10	24	4	45	2	14	0	3
 nl.tudelft.jpacman.level	<div><div></div></div>	67%	<div><div></div></div>	57%	74	155	104	344	21	69	4	12
 nl.tudelft.jpacman.npc	<div><div></div></div>	100%		n/a	0	4	0	8	0	4	0	1
 nl.tudelft.jpacman.npc.ghost	<div><div></div></div>	71%	<div><div></div></div>	55%	56	105	43	181	5	34	0	8
 nl.tudelft.jpacman.points	<div><div></div></div>	60%	<div><div></div></div>	75%	1	11	5	21	0	9	0	2
 nl.tudelft.jpacman.sprite	<div><div></div></div>	87%	<div><div></div></div>	59%	29	70	10	113	4	38	0	5
 nl.tudelft.jpacman.ui	<div><div></div></div>	77%	<div><div></div></div>	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