

LAB REPORT 2

TASK 2.1:

Pre-Coverage Test – ButtonPanel()

Coverage: Tests in 'jpacman.test' x			
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)
i Action	100% (0/0)	100% (0/0)	100% (0/0)
c BoardPanel	0% (0/1)	0% (0/5)	0% (0/27)
c ButtonPanel	0% (0/1)	0% (0/3)	0% (0/11)
c PacKeyListener	0% (0/1)	0% (0/5)	0% (0/10)
c PacManUI	0% (0/1)	0% (0/4)	0% (0/24)
c PacManUiBuilder	0% (0/1)	0% (0/9)	0% (0/30)
c ScorePanel	0% (0/1)	0% (0/5)	0% (0/25)
c Launcher	0% (0/1)	0% (0/21)	0% (0/41)
c LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)
c PacmanConfigurationExc	0% (0/1)	0% (0/2)	0% (0/4)

Post-Coverage Test – ButtonPanel

Coverage: Tests in 'jpacman.test' ×			
Element ▲			
Class, %			
Method, %			
Line, %			
▼ nl	7% (8/110)	2% (16/6...	1% (42/2...
▼ tudelft	7% (8/110)	2% (16/6...	1% (42/2...
▼ jpacman	7% (8/110)	2% (16/6...	1% (42/2...
> board	20% (4/...	9% (10/1...	9% (28/2...
> fuzzer	0% (0/2)	0% (0/12)	0% (0/64)
> game	0% (0/6)	0% (0/28)	0% (0/86)
> integration	0% (0/2)	0% (0/8)	0% (0/12)
> level	0% (0/26)	0% (0/156)	0% (0/69...
> 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	33% (4/12)	9% (6/62)	5% (14/2...
I Action	100% (0/...	100% (0/0)	100% (0/0)
C BoardPanel	0% (0/1)	0% (0/5)	0% (0/31)
C ButtonPanel	100% (1/1)	66% (2/3)	46% (6/13)
C PacKeyListener	0% (0/1)	0% (0/5)	0% (0/10)
C PacManUI	100% (1/1)	25% (1/4)	3% (1/27)
C PacManUiBuilder	0% (0/1)	0% (0/9)	0% (0/30)
C ScorePanel	0% (0/1)	0% (0/5)	0% (0/28)
G Launcher	0% (0/1)	0% (0/21)	0% (0/41)
G LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)
⚡ PacmanConfiguratio	0% (0/1)	0% (0/2)	0% (0/4)

Pre-Coverage Test – PacManUI()

Coverage: Tests in 'jpacman.test' ×			
Element ▲	Class, %	Method, %	Line, %
▼ nl	7% (8/110)	2% (16/6...)	1% (42/2...
▼ tudelft	7% (8/110)	2% (16/6...)	1% (42/2...
▼ jpacman	7% (8/110)	2% (16/6...)	1% (42/2...
> board	20% (4/...	9% (10/1...	9% (28/2...
> fuzzer	0% (0/2)	0% (0/12)	0% (0/64)
> game	0% (0/6)	0% (0/28)	0% (0/86)
> integration	0% (0/2)	0% (0/8)	0% (0/12)
> level	0% (0/26)	0% (0/156)	0% (0/69...
> 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	33% (4/12)	9% (6/62)	5% (14/2...
i Action	100% (0/...	100% (0/0)	100% (0/0)
c BoardPanel	0% (0/1)	0% (0/5)	0% (0/31)
c ButtonPanel	100% (1/1)	66% (2/3)	46% (6/13)
c PacKeyListener	0% (0/1)	0% (0/5)	0% (0/10)
c PacManUI	100% (1/1)	25% (1/4)	3% (1/27)
c PacManUiBuilder	0% (0/1)	0% (0/9)	0% (0/30)
c ScorePanel	0% (0/1)	0% (0/5)	0% (0/28)
c Launcher	0% (0/1)	0% (0/21)	0% (0/41)
c LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)
⚡ PacmanConfiguratio	0% (0/1)	0% (0/2)	0% (0/4)

Post-Coverage Test – PacManUI()

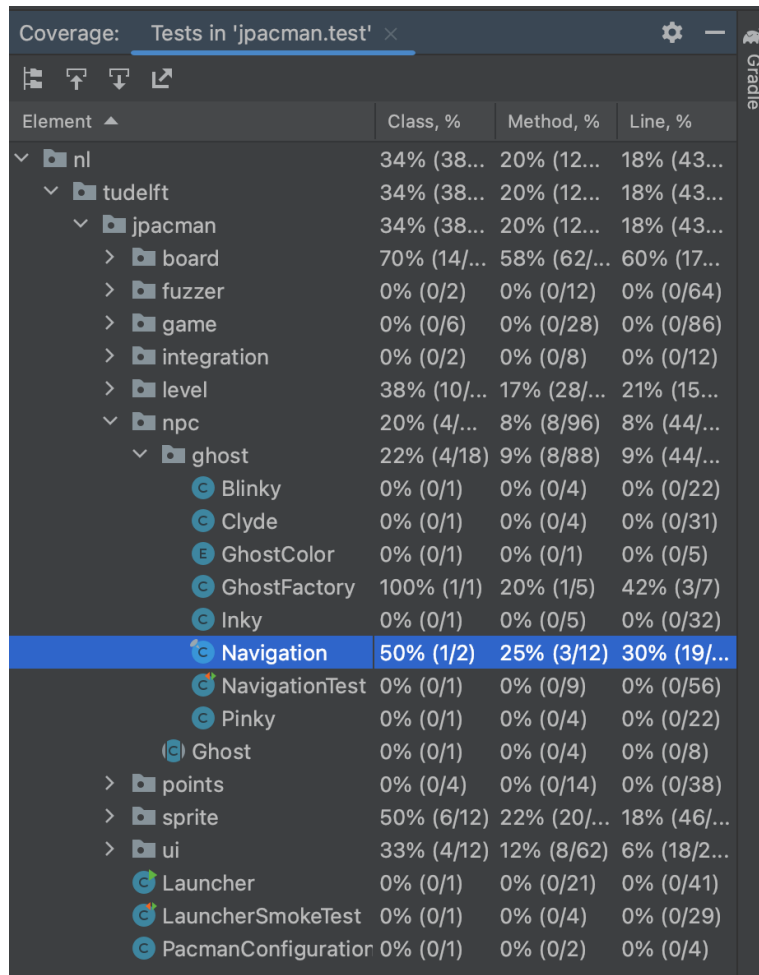
Element ▲	Class, %	Method, %	Line, %
▼ nl	7% (8/110)	2% (18/6...)	1% (46/2...
▼ tudelft	7% (8/110)	2% (18/6...)	1% (46/2...
▼ jpacman	7% (8/110)	2% (18/6...)	1% (46/2...
> board	20% (4/...	9% (10/1...	9% (28/2...
> fuzzer	0% (0/2)	0% (0/12)	0% (0/64)
> game	0% (0/6)	0% (0/28)	0% (0/86)
> integration	0% (0/2)	0% (0/8)	0% (0/12)
> level	0% (0/26)	0% (0/156)	0% (0/69...
> 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% (0/23...
▼ ui	33% (4/12)	12% (8/62)	6% (18/2...
I Action	100% (0/...	100% (0/0)	100% (0/0)
C BoardPanel	0% (0/1)	0% (0/5)	0% (0/31)
C ButtonPanel	100% (1/1)	66% (2/3)	46% (6/13)
C PacKeyListener	0% (0/1)	0% (0/5)	0% (0/10)
C PacManUI	100% (1/1)	50% (2/4)	11% (3/27)
C PacManUiBuilder	0% (0/1)	0% (0/9)	0% (0/30)
C ScorePanel	0% (0/1)	0% (0/5)	0% (0/28)
C Launcher	0% (0/1)	0% (0/21)	0% (0/41)
C LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)
C PacmanConfiguratio	0% (0/1)	0% (0/2)	0% (0/4)

Gradle

Pre-Coverage Test – findNearest()

Coverage: Tests in 'jpacman.test' x			
Element ^	Class, %	Method, %	Line, %
▼ nl	7% (8/110)	2% (18/6...)	1% (46/2...
▼ tudelft	7% (8/110)	2% (18/6...)	1% (46/2...
▼ jpacman	7% (8/110)	2% (18/6...)	1% (46/2...
> board	20% (4/...	9% (10/1...	9% (28/2...
> fuzzer	0% (0/2)	0% (0/12)	0% (0/64)
> game	0% (0/6)	0% (0/28)	0% (0/86)
> integration	0% (0/2)	0% (0/8)	0% (0/12)
> level	0% (0/26)	0% (0/156)	0% (0/69...
▼ npc	0% (0/20)	0% (0/94)	0% (0/474)
▼ ghost	0% (0/18)	0% (0/86)	0% (0/45...
Blinky	0% (0/1)	0% (0/4)	0% (0/21)
Clyde	0% (0/1)	0% (0/4)	0% (0/29)
GhostColor	0% (0/1)	0% (0/1)	0% (0/5)
GhostFactory	0% (0/1)	0% (0/5)	0% (0/6)
Inky	0% (0/1)	0% (0/5)	0% (0/31)
Navigation	0% (0/2)	0% (0/11)	0% (0/60)
NavigationTest	0% (0/1)	0% (0/9)	0% (0/56)
Pinky	0% (0/1)	0% (0/4)	0% (0/21)
Ghost	0% (0/1)	0% (0/4)	0% (0/8)
> points	0% (0/4)	0% (0/14)	0% (0/38)
> sprite	0% (0/12)	0% (0/90)	0% (0/23...
> ui	33% (4/12)	12% (8/62)	6% (18/2...
Launcher	0% (0/1)	0% (0/21)	0% (0/41)
LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)
PacmanConfigurator	0% (0/1)	0% (0/2)	0% (0/4)

Post-Coverage Test – findNearest()



The screenshot shows the 'Coverage: Tests in 'jpacman.test'' window in IntelliJ IDEA. The table displays coverage metrics for various elements in the project. The 'Navigation' class is highlighted with a blue background, showing 50% class coverage (1/2), 25% method coverage (3/12), and 30% line coverage (19/63).

Element	Class, %	Method, %	Line, %
nl	34% (38/111)	20% (12/60)	18% (43/239)
tudelft	34% (38/111)	20% (12/60)	18% (43/239)
jpacman	34% (38/111)	20% (12/60)	18% (43/239)
board	70% (14/20)	58% (62/107)	60% (17/28)
fuzzer	0% (0/2)	0% (0/12)	0% (0/64)
game	0% (0/6)	0% (0/28)	0% (0/86)
integration	0% (0/2)	0% (0/8)	0% (0/12)
level	38% (10/26)	17% (28/164)	21% (15/71)
npc	20% (4/20)	8% (8/96)	8% (44/544)
ghost	22% (4/18)	9% (8/88)	9% (44/488)
Blinky	0% (0/1)	0% (0/4)	0% (0/22)
Clyde	0% (0/1)	0% (0/4)	0% (0/31)
GhostColor	0% (0/1)	0% (0/1)	0% (0/5)
GhostFactory	100% (1/1)	20% (1/5)	42% (3/7)
Inky	0% (0/1)	0% (0/5)	0% (0/32)
Navigation	50% (1/2)	25% (3/12)	30% (19/63)
NavigationTest	0% (0/1)	0% (0/9)	0% (0/56)
Pinky	0% (0/1)	0% (0/4)	0% (0/22)
Ghost	0% (0/1)	0% (0/4)	0% (0/8)
points	0% (0/4)	0% (0/14)	0% (0/38)
sprite	50% (6/12)	22% (20/91)	18% (46/256)
ui	33% (4/12)	12% (8/62)	6% (18/299)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)
LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)
PacmanConfiguration	0% (0/1)	0% (0/2)	0% (0/4)

TASK 3:

Q1: The coverage results are different from JaCoCo to IntelliJ. The difference can be caused by the different code that is used to test the methods, as different testing code will test different parts and thus cover different amounts of the code.

Q2: Yes, the source code visualization from JaCoCo was very helpful. It showed, line by line, directly what was covered and also not covered. It's more helpful than only having a coverage percentage.

Q3: I prefer the JaCoCo visualization, as it provides a more in depth look at the coverage of the code. Even so, it also provides a broad view just like IntelliJ, leaving more options for the user to view coverage.

Link to jpacman fork repo: <https://github.com/deleoa5/jpacman-Deleon/tree/master/src/test/java/nl/tudelft/jpacman>