Name: Jose A. Garcia
Date: 2/2/23
Class: CS 472

Testing lab

Task 2.1:

Coverage with the 1ST test: **src/main/java/nl/tudelft/jpacman/level/LevelFactory.java@createPellet()**

Coverage before test was added:

I.

Coverage: Tests in 'nl.tudelft.jpacman'

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

Coverage after test was added:

II.

Element	Class, %	Method, %	Line, %
nl	25% (28/110)	12% (80/624)	10% (246/2330)
tudelft	25% (28/110)	12% (80/624)	10% (246/2330)
jpacman	25% (28/110)	12% (80/624)	10% (246/2330)
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	30% (8/26)	11% (18/156)	7% (50/706)
npc	10% (2/20)	2% (2/94)	1% (6/486)
points	100% (4/4)	57% (8/14)	54% (24/44)
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)

Test source code PelletTest.java

III.

```java
public class PelletTest {
//    Create a levelFactory to create a pellet
//        Create a PacmanSprites
        2 usages
        private final PacManSprites sprite_loader = new PacManSprites();
//        Create a GhostFactory
        1 usage
        private GhostFactory ghFactory = new GhostFactory(sprite_loader);
//        Create a pointCalculator
        1 usage
        private PointCalculatorLoader tempCalc = new PointCalculatorLoader();

        1 usage
        private LevelFactory Lvl_factory = new LevelFactory(sprite_loader, ghFactory, tempCalc.load());

        1 usage
        private Pellet testingPellet = Lvl_factory.createPellet();

//        If pellet was created then the value would be set to 10 and nothing else

    no usages  new *
    @Test
        void testPellet() { assertThat(testingPellet.getValue()).isEqualTo( expected: 10); }
}
```

a. Note that whenever I create a new pellet in the constructor a default value of 10 is set to its *value* attribute thus my thinking of testing it the way shown above.
b. Also note that by adding this test my level class coverage incremented by 15%.and also my method coverage incremented by 5%.

Coverage with the 2nd test: **src/main/java/nl/tudelft/jpacman/level/LevelFactory.java@createGhost()**

- Please note I modified the source code to include a getter for ghost index to keep track of the number of ghosts incrementing

Coverage before test was added:

I.

Element	Class, %	Method, %	Line, %
nl	25% (28/110)	12% (80/624)	10% (246/2330)
tudelft	25% (28/110)	12% (80/624)	10% (246/2330)
jpacman	25% (28/110)	12% (80/624)	10% (246/2330)
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	30% (8/26)	11% (18/156)	7% (50/706)
npc	10% (2/20)	2% (2/94)	1% (6/486)
points	100% (4/4)	57% (8/14)	54% (24/44)
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)

Coverage after test was added:

II.

Coverage: Tests in 'jpacman.test'

Element	Class, %	Method, %	Line, %
nl	30% (34/110)	15% (96/626)	12% (290/2332)
tudelft	30% (34/110)	15% (96/626)	12% (290/2332)
jpacman	30% (34/110)	15% (96/626)	12% (290/2332)
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	30% (8/26)	13% (22/158)	8% (60/708)
npc	40% (8/20)	12% (12/94)	6% (34/486)
points	100% (4/4)	57% (8/14)	54% (24/44)
sprite	83% (10/12)	48% (44/90)	55% (144/260)
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)

Test source code GhostTest.java:

III.

```java
public class GhostTest{
    //    Create a levelFactory to create a pellet
    //        Create a PacmanSprites
    2 usages
    private final PacManSprites sprite_loader = new PacManSprites();
    //        Create a GhostFactory
    1 usage
    private GhostFactory ghFactory = new GhostFactory(sprite_loader);
    //        Create a pointCalculator
    1 usage
    private PointCalculatorLoader tempCalc = new PointCalculatorLoader();

    //        Create a level factory
    2 usages
    private LevelFactory Lvl_factory = new LevelFactory(sprite_loader, ghFactory, tempCalc.load());

    //      Create a ghost for the level factory
    no usages
    private Ghost test_ghost = Lvl_factory.createGhost();

    //      If the ghostIndex is greater than -1(default value) that means that a ghost was created
    no usages  new *
    @Test
    void testGhost() { assertThat(Lvl_factory.getGhostIndex()).isGreaterThan( other: -1); }

}
```

a. For this test when creating a new ghost in the LevelFactory class there's an attribute named ghostIndex which keeps track of how many ghost exist in the current object. I simply check after creating one that the ghost index is greater than the default of -1.
b. Also note that by doing this test my npc coverage in class increased from 10% -> 40%.

Coverage for the 3rd test: **src/main/java/nl/tudelft/jpacman/board/BoardFactory@createBoard()**

Coverage before test was added:

I.

Element	Class, %	Method, %	Line, %
nl	30% (34/110)	15% (96/626)	12% (290/2332)
tudelft	30% (34/110)	15% (96/626)	12% (290/2332)
jpacman	30% (34/110)	15% (96/626)	12% (290/2332)
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	30% (8/26)	13% (22/158)	8% (60/708)
npc	40% (8/20)	12% (12/94)	6% (34/486)
points	100% (4/4)	57% (8/14)	54% (24/44)
sprite	83% (10/12)	48% (44/90)	55% (144/260)
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)

Coverage after test was added:

II.

Coverage: Tests in 'jpacman.test'

Element	Class, %	Method, %	Line, %
nl	36% (40/110)	18% (116/626)	15% (352/2338)
tudelft	36% (40/110)	18% (116/626)	15% (352/2338)
jpacman	36% (40/110)	18% (116/626)	15% (352/2338)
board	50% (10/20)	28% (30/106)	31% (90/288)
fuzzer	0% (0/2)	0% (0/12)	0% (0/64)
game	0% (0/6)	0% (0/28)	0% (0/74)
integrati	0% (0/2)	0% (0/8)	0% (0/12)
level	30% (8/26)	13% (22/158)	8% (60/708)
npc	40% (8/20)	12% (12/94)	6% (34/486)
points	100% (4/4)	57% (8/14)	54% (24/44)
sprite	83% (10/12)	48% (44/90)	55% (144/260)
ui	0% (0/12)	0% (0/62)	0% (0/254)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)
Launcher	0% (0/1)	0% (0/4)	0% (0/29)
PacmanC	0% (0/1)	0% (0/2)	0% (0/4)

Test source code BoardTest.java

III.

```java
public class BoardTest {
        // Create a sprite_store
        1 usage
        private static final PacManSprites sprite_store = new PacManSprites();
        // Create a board_factory
        1 usage
        private BoardFactory board_factory = new BoardFactory(sprite_store);
        // Create a Square object of 2d
        1 usage
        private Square newSqr[][]={
            {mock(Square.class)},
            {mock(Square.class)}
        };
        // Create a test board
        2 usages
        private Board test_board = board_factory.createBoard(newSqr);

        // If it was created correctly then the board size would exist 2d with a dimension greater than 0
        no usages  new*
        @Test
        void create_board_test(){

            assertThat(test_board.getWidth()).isGreaterThan( other: 0);
            assertThat(test_board.getHeight()).isGreaterThan( other: 0);

        }
```

a. For this test when creating a new board, the default width and height are never set so by immediately creating one of and checking its dimension is a good enough check.
b. My board class package coverage incremented from 30% -> 50%

Task 3:

I. Are the coverage results from `JaCoCo` similar to the ones you got from `IntelliJ` in the last task? Why so or why not?

   a. *The coverage results from JaCoCo are different from the ones produced by IntelliJ. My thoughts for this are probably because JaCoCo also takes into account branches so thus needs a different technique to do so compared to IntelliJ.*

   b.

jpacman — Sessions

jpacman

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
nl.tudelft.jpacman.level		67%		57%	74	156	104	345	21	70	4	12
nl.tudelft.jpacman.npc.ghost		71%		55%	56	105	43	181	5	34	0	8
nl.tudelft.jpacman.ui		77%		47%	54	86	21	144	7	31	0	6
default		0%		0%	12	12	21	21	5	5	1	1
nl.tudelft.jpacman.board		86%		58%	44	93	2	110	0	40	0	7
nl.tudelft.jpacman.sprite		86%		59%	30	70	11	113	5	38	0	5
nl.tudelft.jpacman		69%		25%	12	30	18	52	6	24	1	2
nl.tudelft.jpacman.points		60%		75%	1	11	5	21	0	9	0	2
nl.tudelft.jpacman.game		87%		60%	10	24	4	45	2	14	0	3
nl.tudelft.jpacman.npc		100%		n/a	0	4	0	8	0	4	0	1
Total	1,213 of 4,697	74%	293 of 637	54%	293	591	229	1,040	51	269	6	47

Created with JaCoCo 0.8.3.201901230119

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

   a. *Yes, I did. They added much more detail with including the missed branches section as well as the missed instructions. It's much more detailed than Intellij.*

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

   a. *Personally, I prefer IntelliJ's coverage as it's simpler to follow. However, it's pretty neat that JaCoCo includes so much detail in their report like possibly missed branches. I could make a case for both i.e) like using IntelliJ for the draft project and switching over to JaCoCo to catch all the missed branches.*

Team Repo: https://github.com/justin-negron/SoftwareProductDesign

My Repo: https://github.com/soulos98/jpacman