Tyler Cook
CS 472
Testing Lab

Fork Repository: https://github.com/cookt5/jpacman.git

Board.squareAt()
The unit test for the method squareAt(), for the Board class, tests if the method returns the correct Square from a matrix of Squares. To test the method, a board is created and a single object of type Blinky is stored in the board. The x and y coordinates of the object are passed to the squareAt() method, if the Square returned has the object, the test passes.

Board.squareAt() – Results Before Unit Test

Coverage: Tests in 'jpacman.test'

Element	Class, %	Method, %	Line, %
nl	30% (34...	17% (106/...	12% (290...
tudelft	30% (34...	17% (106/...	12% (290...
jpacman	30% (34...	17% (106/...	12% (290...
board	18% (4/2...	13% (14/1...	12% (34/...
Board	0% (0/1)	0% (0/7)	0% (0/17)
BoardFactory	0% (0/3)	0% (0/11)	0% (0/27)
BoardFactoryTest	0% (0/1)	0% (0/6)	0% (0/18)
BoardTest	0% (0/1)	0% (0/1)	0% (0/1)
Direction	100% (1/1)	75% (3/4)	90% (10/...
Square	0% (0/1)	0% (0/8)	0% (0/23)
SquareTest	0% (0/1)	0% (0/4)	0% (0/13)
Unit	100% (1/1)	40% (4/10)	24% (7/29)
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	23% (6/...	11% (18/1...	5% (42/7...
npc	70% (14/...	31% (30/...	14% (70/...
points	0% (0/4)	0% (0/14)	0% (0/38)
sprite	83% (10/...	48% (44/...	55% (144...
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)
PacmanConfigurationI	0% (0/1)	0% (0/2)	0% (0/4)

Board.squareAt() – Results After Unit Test

Coverage: Tests in 'jpacman.test'

Element	Class, %	Method, %	Line, %
nl	41% (46/112)	24% (156/6...	19% (468/2...
tudelft	41% (46/112)	24% (156/6...	19% (468/2...
jpacman	41% (46/112)	24% (156/6...	19% (468/2...
board	72% (16/22)	57% (62/108)	61% (210/3...
Board	100% (1/1)	100% (7/7)	94% (17/18)
BoardFactory	66% (2/3)	45% (5/11)	72% (21/29)
BoardFactoryTest	0% (0/1)	0% (0/6)	0% (0/18)
BoardTest	100% (1/1)	100% (2/2)	100% (15/15)
Direction	100% (1/1)	100% (4/4)	100% (11/11)
Square	100% (1/1)	75% (6/8)	78% (18/23)
SquareTest	0% (0/1)	0% (0/4)	0% (0/13)
Unit	100% (1/1)	50% (5/10)	27% (8/29)
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	23% (6/26)	11% (18/156)	5% (42/702)
npc	70% (14/20)	31% (30/94)	14% (70/486)
points	0% (0/4)	0% (0/14)	0% (0/38)
sprite	83% (10/12)	51% (46/90)	56% (146/2...
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)
PacmanConfigurationExcepti	0% (0/1)	0% (0/2)	0% (0/4)

Board.squareAt() – Code for Unit Test

```
public class BoardTest {
    2 usages
    private static final PacManSprites sprite_store = new PacManSprites();
    2 usages
    private BoardFactory boardFactory = new BoardFactory(sprite_store);
    1 usage
    private GhostFactory ghostFactory = new GhostFactory(sprite_store);
    3 usages
    private Square[][] square  = new Square[10][10];
    1 usage
    private Ghost testBlinky = ghostFactory.createBlinky();


    no usages
    @Test
    void squareHasOccupantBlinky_squareAt_hasOccupantBlinky(){
        for(int i=0; i<10; i++) {
            for (int j = 0; j < 10; j++) {
                square[i][j] = boardFactory.createGround();
            }
        }
        square[3][4].put(testBlinky);
        Board board = boardFactory.createBoard(square);
        Square testSquare = board.squareAt( x: 3,  y: 4);
        List<Unit> testOccupantList = testSquare.getOccupants();
        assertThat(testOccupantList.get(0)).isExactlyInstanceOf(Blinky.class);
    }
}
```

Player.addPoints()

The unit test for the method addPoints(), for the Player class, tests if the method properly increases the score member variable. To test the method, the current score is stored and the addPoints() method is called and passed a 1. If the new score equals the previously stored score plus one, then the test passes.

Player.addPoints() - Results Before Unit Test

Coverage: Tests in 'jpacman.test'

Element	Class, %	Method, %	Line, %
nl	41% (46/112)	24% (152/6...	19% (462/2...
tudelft	41% (46/112)	24% (152/6...	19% (462/2...
jpacman	41% (46/112)	24% (152/6...	19% (462/2...
board	72% (16/22)	57% (62/108)	61% (210/3...
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	23% (6/26)	8% (14/156)	5% (36/702)
CollisionInteractionMap	0% (0/2)	0% (0/9)	0% (0/41)
CollisionMap	100% (0/0)	100% (0/0)	100% (0/0)
DefaultPlayerInteractionMa	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	100% (1/1)	66% (2/3)	83% (5/6)
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	70% (14/20)	31% (30/94)	14% (70/486)
points	0% (0/4)	0% (0/14)	0% (0/38)
sprite	83% (10/12)	51% (46/90)	56% (146/2...
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)
PacmanConfigurationExcepti	0% (0/1)	0% (0/2)	0% (0/4)

Player.addPoints() – Results After Unit Test

Coverage: Tests in 'jpacman.test'

Element	Class, %	Method, %	Line, %
nl	41% (46/112)	24% (156/6...	19% (468/2...
tudelft	41% (46/112)	24% (156/6...	19% (468/2...
jpacman	41% (46/112)	24% (156/6...	19% (468/2...
board	72% (16/22)	57% (62/108)	61% (210/3...
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	23% (6/26)	11% (18/156)	5% (42/702)
CollisionInteractionMap	0% (0/2)	0% (0/9)	0% (0/41)
CollisionMap	100% (0/0)	100% (0/0)	100% (0/0)
DefaultPlayerInteractionMa	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	100% (1/1)	66% (2/3)	83% (5/6)
Player	100% (1/1)	50% (4/8)	45% (11/24)
PlayerCollisions	0% (0/1)	0% (0/7)	0% (0/21)
PlayerFactory	100% (1/1)	100% (3/3)	100% (5/5)
npc	70% (14/20)	31% (30/94)	14% (70/486)
points	0% (0/4)	0% (0/14)	0% (0/38)
sprite	83% (10/12)	51% (46/90)	56% (146/2...
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)
PacmanConfigurationExcepti	0% (0/1)	0% (0/2)	0% (0/4)

Player.addPoints() – Code for Unit Test

```
no usages
public class PlayerTest_addPoints {
    1 usage
    private static final PacManSprites sprite_store = new PacManSprites();
    1 usage
    private PlayerFactory playerFactory = new PlayerFactory(sprite_store);
    3 usages
    private Player player = playerFactory.createPacMan();

    no usages
    @Test
    void initialScore_addPoints_scoreIncrementByOne(){
        int score = player.getScore();
        player.addPoints(1);
        assertThat(player.getScore()).isEqualTo( expected: score + 1);
    }
}
```

GhostFactory.createBlinky()

The unit test for the method createBlinky(), for the GhostFactory class, tests if the method returns the correct type. To test the method, an object of type Ghost is initialized with the method createBlinky(), which should return an object of type Blinky. An assertion checks if the returned type is equal to the Blinky class. If the function returns any object that is not of type Blinky, the test fails.

GhostFactory.createBlinky() – Results Before Unit Test

Coverage: Tests in 'jpacman.test'

Element	Class, %	Method, %	Line, %
nl	17% (20/112)	11% (74/622)	9% (214/23...
tudelft	17% (20/112)	11% (74/622)	9% (214/23...
jpacman	17% (20/112)	11% (74/622)	9% (214/23...
board	18% (4/22)	13% (14/104)	12% (34/280)
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	23% (6/26)	11% (18/156)	5% (42/702)
npc	0% (0/20)	0% (0/94)	0% (0/474)
ghost	0% (0/18)	0% (0/86)	0% (0/458)
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	83% (10/12)	46% (42/90)	53% (138/2...
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)
PacmanConfigurationExcepti	0% (0/1)	0% (0/2)	0% (0/4)

GhostFactory.createBlinky() – Results After Unit Test

Coverage: Tests in 'jpacman.test'

Element	Class, %	Method, %	Line, %
nl	25% (28/11...	14% (88/622)	10% (254/2...
tudelft	25% (28/11...	14% (88/622)	10% (254/2...
jpacman	25% (28/11...	14% (88/622)	10% (254/2...
board	18% (4/22)	13% (14/104)	12% (34/280)
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	23% (6/26)	11% (18/156)	5% (42/702)
npc	40% (8/20)	12% (12/94)	6% (34/486)
ghost	33% (6/18)	11% (10/86)	5% (24/470)
Blinky	100% (1/1)	50% (2/4)	13% (3/22)
Clyde	0% (0/1)	0% (0/4)	0% (0/31)
GhostColor	100% (1/1)	100% (1/1)	100% (5/5)
GhostFactory	100% (1/1)	40% (2/5)	57% (4/7)
Inky	0% (0/1)	0% (0/5)	0% (0/32)
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% (0/22)
Ghost	100% (1/1)	25% (1/4)	62% (5/8)
points	0% (0/4)	0% (0/14)	0% (0/38)
sprite	83% (10/12)	48% (44/90)	55% (144/2...
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)
PacmanConfigurationExcepti	0% (0/1)	0% (0/2)	0% (0/4)

GhostFactory.createBlinky() – Code for Unit Test

```
public class GhostFactoryTest {
    1 usage
    private static final PacManSprites sprite_store = new PacManSprites();
    1 usage
    private GhostFactory ghostFactory = new GhostFactory(sprite_store);

    no usages
    @Test
    void initialGhostClass_createBlinky_returnTypeBlinky(){
        Ghost blinky = ghostFactory.createBlinky();
        assertThat(blinky).isExactlyInstanceOf(Blinky.class);
    }
}
```

Task 3 – JaCoCo Report on JPacman

The coverage results from JaCoCo and IntelliJ were different for two of the three classes. Starting with the GhostFactory class, IntelliJ reported 2 of 5 methods and 4 of 7 lines were covered, however, JaCoCo reported that GhostFactory had 100% coverage. The Player class also had differing results with IntelliJ reporting 4 of 8 methods and 11 of 24 lines covered. JaCoCo reported 7 of 8 methods and 20 of 24 lines covered. Results for the Board class matched with 7 of 7 methods and 17 of 18 lines covered.

The source code visualization from JaCoCo is very helpful as it allows me to quickly locate what methods or branches in a method are not covered in the unit test.

IntelliJ and JaCoCo were both useful for creating unit tests. IntelliJ provided high level results immediately without having to open a new window, which helped in determining if the unit test being developed was providing any coverage. JaCoCo is useful for providing more detailed results and visualizes what methods and branches are not covered. As a preference, JaCoCo would be the unit test tool I would use most often.