Unit Testing and Coverage Analysis in JPacman

<https://github.com/kzmaybe/2DRogueLikeUnityGame.git>

Task 1

A screenshot of a computer

Description automatically generated

Ans: The unit test only covers 3% of the Class, 1% of the Method and Line, which is not good enough.

Task 2

A screenshot of a computer

Description automatically generated

\*added PlayerTest

Task 2.1

A screenshot of a computer

Description automatically generated\*added Points test

A screen shot of a computer program

Description automatically generated

Test Methods Added:

* testCollidedWithAGhost: Ensures no points are added when colliding with a ghost.
* testConsumedAPellet: Confirms that points are added correctly when a pellet is consumed.
* testPacmanMoved: Vertifies that no points are added when Pacman moves.

Task 3

* Ans1:

The coverage results from the JaCoCo report indicate a 73% instruction coverage, in contrast to the 8%-line coverage observed in the IntelliJ report from Task 2.1. The superior coverage reported by JaCoCo can be attributed to the broader scope of testing it encompasses. In contrast, the tests implemented during Task 2.1 were limited to only four functions, which causes a large portion of the codebase remains untested.

* Ans2

The source code visualization provided by JaCoCo for uncovering branches was indeed helpful. This feature allows for an at-a-glance assessment of which code paths have been exercised by the test suite and which have not, visually distinguishing between executed and missed branches. Such direct feedback within the context of the source code is crucial for identifying untested parts of the codebase, enabling targeted improvements in test coverage.

* Ans3

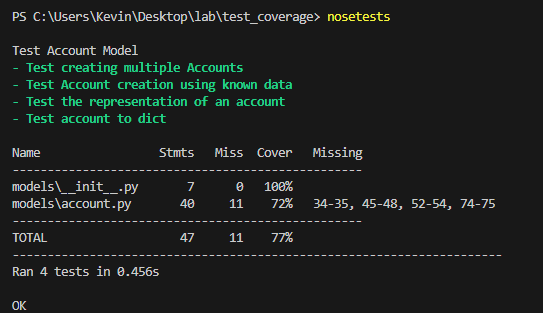
I prefer JaCoCo's report because it provides a comprehensive visual representation of the coverage through charts. Additionally, JaCoCo offers the advantage of displaying the source code with annotations that clearly indicate which sections of code have been covered by tests, enhancing the overall utility of the coverage analysis.

Task 4

Before adding tests:

A screenshot of a computer error

Description automatically generated

Added the provided test in the document:  


Implement the test:  
A screen shot of a computer program

Description automatically generated

Final result:

A screenshot of a computer

Description automatically generated

Task 5

A screenshot of a computer error

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer error

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer error

Description automatically generated

Added test\_update\_a\_counter

A screenshot of a computer program

Description automatically generated

Added update\_counter(name)

A screenshot of a computer program

Description automatically generated

A screen shot of a computer code

Description automatically generated

A screen shot of a computer code

Description automatically generated