

Overall

We start our test by doing unit testing, where each of us can start working on any folder (entity, main, tile, searchAI) that we want and just let the group know, since we are not working at the same time. So, the others can continue the part that is not done yet. Originally, we implemented all unit test for each class that we created, then went deep into integration tests, making sure those tests that need to be done across components are working properly. Lastly, we look at the coverage result and check which part that dragged our percentage down and start working on it to improve the overall coverage, either line or branch coverage.














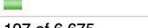
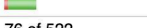
Readme:

1. Clone the project
2. Next, compile the project using: `mvn compile`
3. Then run the project using: `java -cp target/classes Documents.main.Main`
4. After opening the game page, press enter to start and navigate using the arrow key for moving the character.

Test Quality and Coverage

We reached around 97% for line coverage and 88% for branch coverage. We try to aim for the coverage percentage as high as possible. However, there is still a part that is hard for us to implement, which are classes with high dependencies. Another thing is due to `@Override` method, where the parent method is not implemented since the implementation will be in the child method. So, there is no way to do test coverage for the parent method.

Documents

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
 Documents.main		97%		86%	32	170	35	611	2	46	1	8
 Documents.entity		95%		84%	28	143	21	350	0	26	0	5
 Documents.object		94%		62%	3	11	10	58	0	7	0	6
 Documents.tile		99%		86%	3	19	2	83	0	8	0	2
 Documents.SearchAI		99%		86%	7	35	1	104	0	9	0	2
Total	197 of 6,675	97%	76 of 522	85%	73	378	69	1,206	2	96	1	23

What we learnt

In the beginning, we think that doing testing is easy since you only need to call the function and do some basic assertions. However, after working on it, we realize that some class is just nearly impossible to implement the testing as it has too many components connected to each other. Code that has high dependencies is hard to implement for testing, so it is important to do refactoring just like what we learnt from the lecture. But overall, we succeeded in doing the testing and reached our coverage percentage goal.

Changes

During our testing, we do some changes on our main code due to the wrong logic which we catch during the testing. We realized that testing is very important to help us check parts that are hard to measure. We also do some small changes like the encapsulation where applied to improve our code quality.

