Aaron Leondar
CS362 Winter 2017
Assignment 3/Final Project
FinalProject_CS362_001.pdf

Part A:

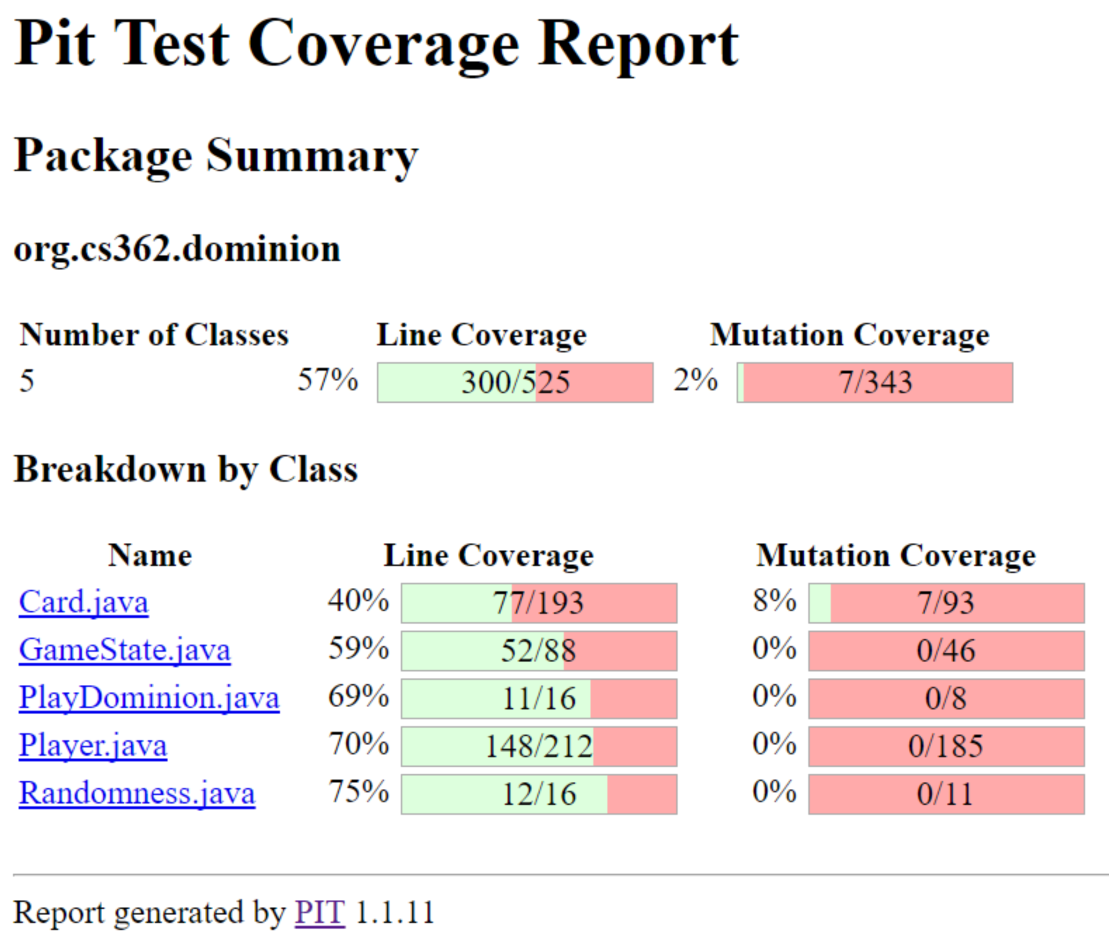
The tool I used to generate the mutation tests for the Final Project was PiTest. I originally thought about looking for other tools, especially since there was a point in testing where I couldn't get PiTest to properly generate the mutation tests. However, I managed to figure out the issues I was having with PiTest before I actually committed to using any other mutation testing tool, and I'm glad about that.

Using PiTest was extremely easy to set up, as it only required inserting a couple lines into the pom.xml file. Though for some reason it took issue with several of my created tests, as well as several of the EvoSuite tests, and I had to omit them so the mutation coverage could be properly generated. Though because of this the mutation coverage was extremely low, so I had to create new test to bring up the mutation coverage. To get it closer to 100% more tests would need to be generated, which could be done with more time.

However, when just testing the EvoSuite tests, there wasn't any mutation coverage, and it would be tough to add in extra tests along with the EvoSuite tests, so EvoSuite, while it does generate a lot of different tests, doesn't generate well-mutatable tests.

Mutation Test Coverage:

Both EvoSuite Tests and Self-Created tests:



Pit Test Coverage Report

Package Summary

org.cs362.dominion

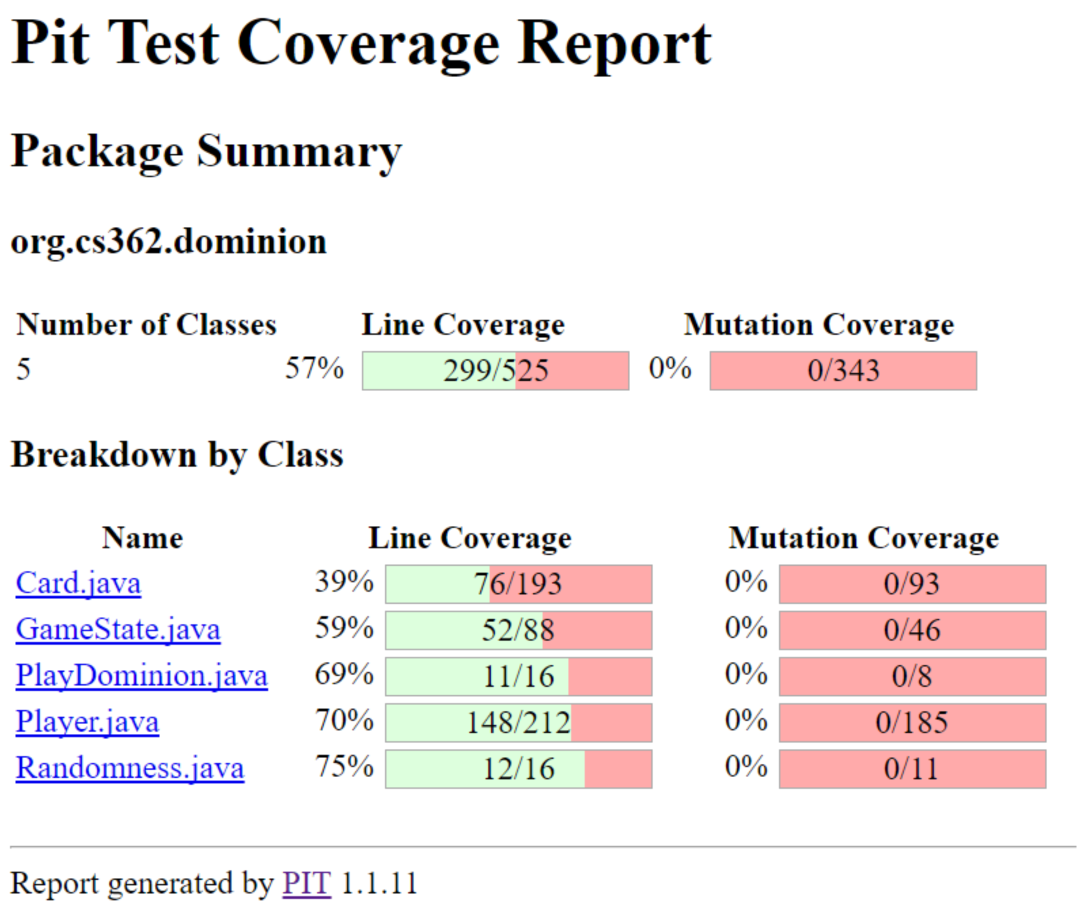
Number of Classes	Line Coverage		Mutation Coverage	
5	57%	300/525	2%	7/343

Breakdown by Class

Name	Line Coverage		Mutation Coverage	
Card.java	40%	77/193	8%	7/93
GameState.java	59%	52/88	0%	0/46
PlayDominion.java	69%	11/16	0%	0/8
Player.java	70%	148/212	0%	0/185
Randomness.java	75%	12/16	0%	0/11

Report generated by PIT 1.1.11

Just EvoSuite Tests mutation coverage



Pit Test Coverage Report

Package Summary

org.cs362.dominion

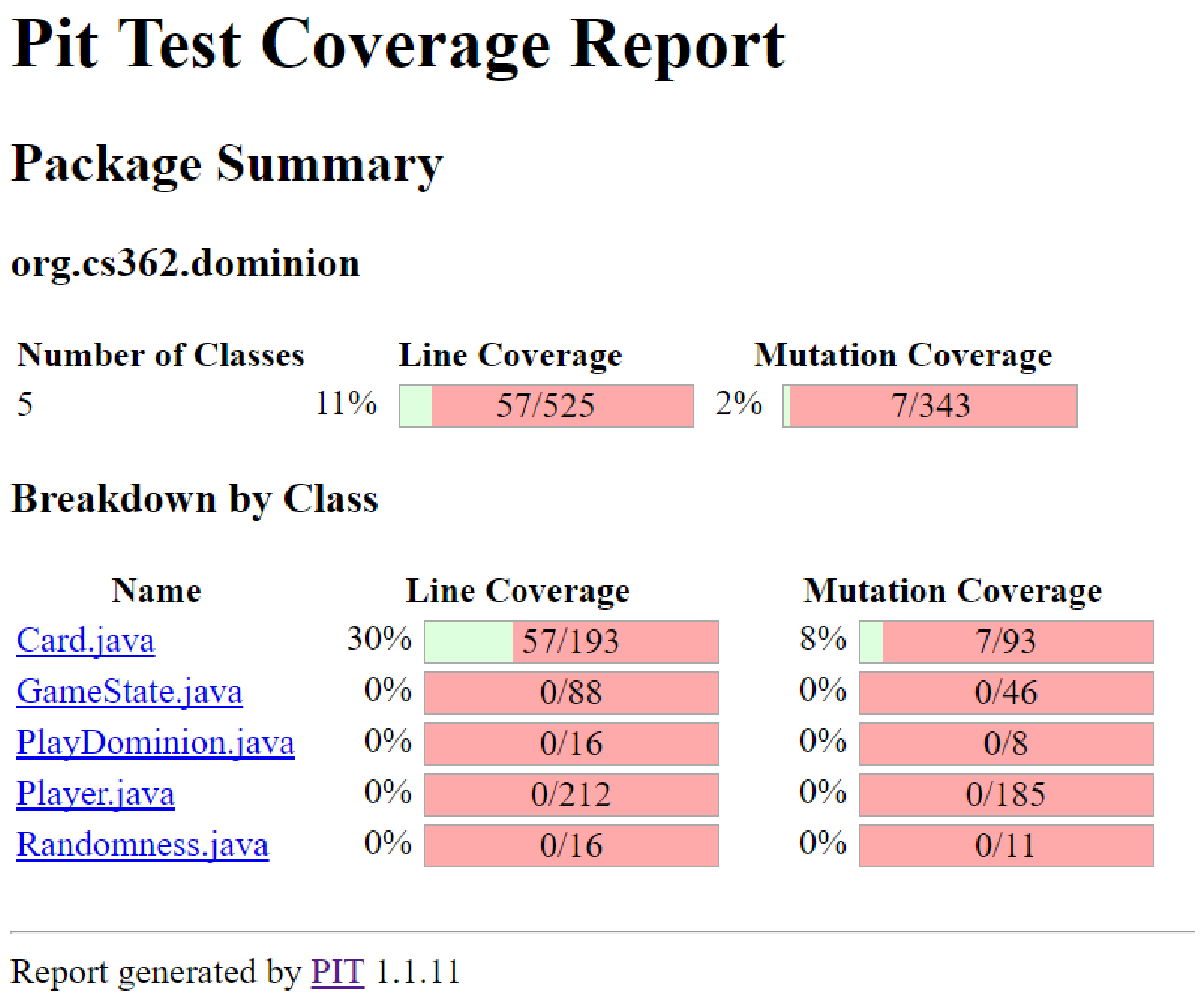
Number of Classes	Line Coverage		Mutation Coverage	
5	57%	299/525	0%	0/343

Breakdown by Class

Name	Line Coverage		Mutation Coverage	
Card.java	39%	76/193	0%	0/93
GameState.java	59%	52/88	0%	0/46
PlayDominion.java	69%	11/16	0%	0/8
Player.java	70%	148/212	0%	0/185
Randomness.java	75%	12/16	0%	0/11

Report generated by PIT 1.1.11

Just Self-made Test mutation Coverage:



Pit Test Coverage Report

Package Summary

org.cs362.dominion

Number of Classes	Line Coverage		Mutation Coverage	
5	11%	57/525	2%	7/343

Breakdown by Class

Name	Line Coverage		Mutation Coverage	
Card.java	30%	57/193	8%	7/93
GameState.java	0%	0/88	0%	0/46
PlayDominion.java	0%	0/16	0%	0/8
Player.java	0%	0/212	0%	0/185
Randomness.java	0%	0/16	0%	0/11

Report generated by PIT 1.1.11

The reason I was unable to get the mutation rate to a higher percentage is because a lot of the tests I attempted to implement and run either wouldn't test correctly, or did not play nice with PiTest and caused PiTest to say it needed a green suite and that some of those particular tests caused the test suite to not be "green".

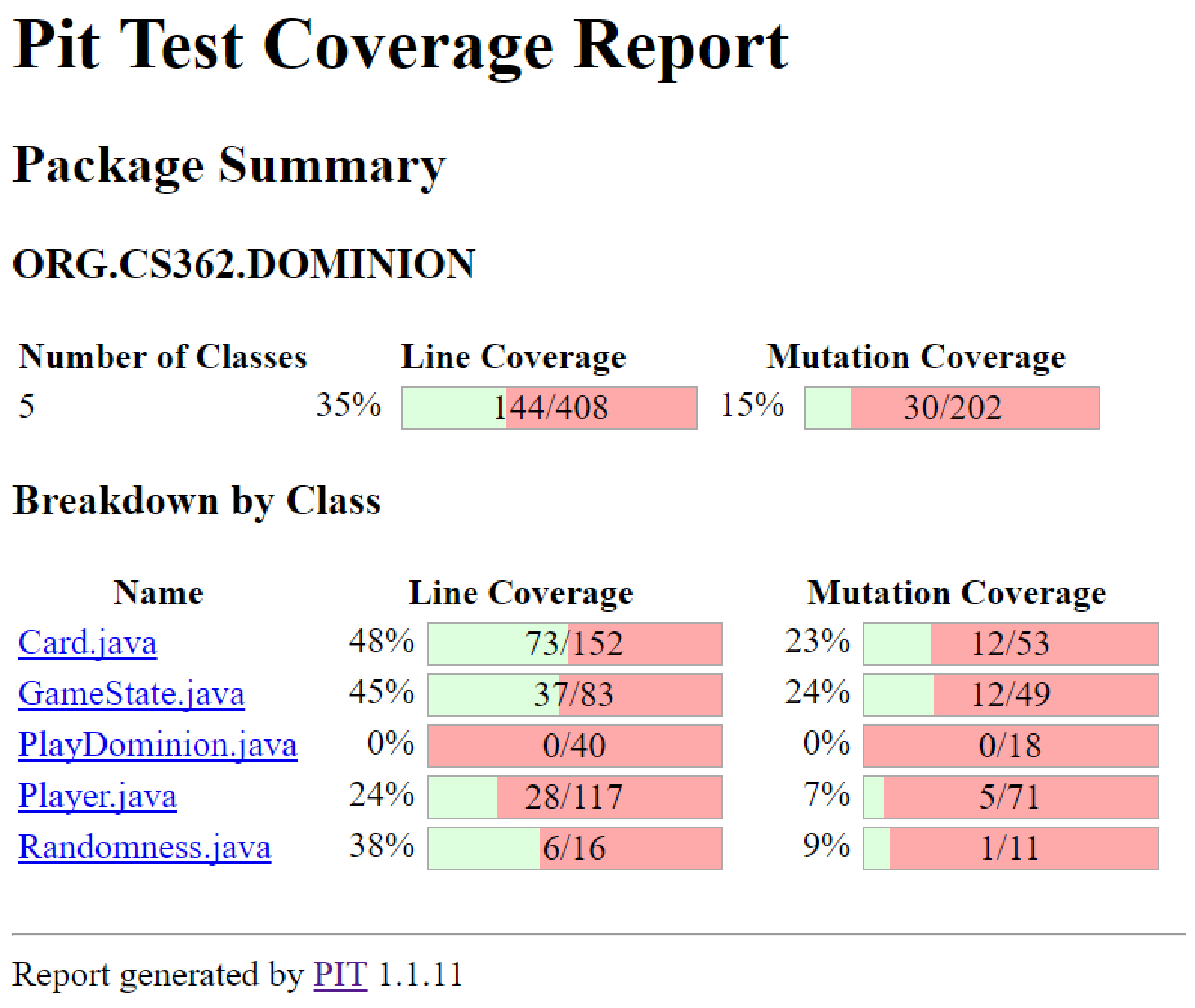
My overall experience with using mutation tools was positive. Mutation seems like an interesting tool in order to test code and see how diverse it is and how diverse it can be.

Part B:

Working with Dominion in general was fun. There was definitely many times when it was frustrating to work with, mainly because Java as a language I have not had any experience in prior to this term, and testing with Maven, EvoSuite, and PiTest were all completely foreign to me. In addition, Dominion was a game I had never played, nor even heard of before this term, so getting to learn that and learn what was required for the programs to work was a challenge in itself.

Student whose code I used: Peter Dorich (dorichp on GitHub)

The code coverage information for the other student whose code I tested is below:



Pit Test Coverage Report

Package Summary

ORG.CS362.DOMINION

Number of Classes	Line Coverage		Mutation Coverage	
5	35%	144/408	15%	30/202

Breakdown by Class

Name	Line Coverage		Mutation Coverage	
Card.java	48%	73/152	23%	12/53
GameState.java	45%	37/83	24%	12/49
PlayDominion.java	0%	0/40	0%	0/18
Player.java	24%	28/117	7%	5/71
Randomness.java	38%	6/16	9%	1/11

Report generated by PIT 1.1.11

As can be seen, the mutation coverage isn't 100%, but it still covers a decent amount of the tests that the other student created, which is pretty good.

I faced absolutely no problems running the mutation test on the other student's code, and the only troubles I faced while generating tests was that the other student's pom.xml file did not include EvoSuite dependencies criteria, so I had to manually add it in. After that however, it worked just fine. All of their tests tested correctly and did not break, I was able to generate tests on their code using EvoSuite, and I was able to obtain the mutation coverage of their unit tests without much trouble.