CSC410 Assignemnt 2

Name: Liu Jin

Student number :1000897657

Name: Siyuan Zheng

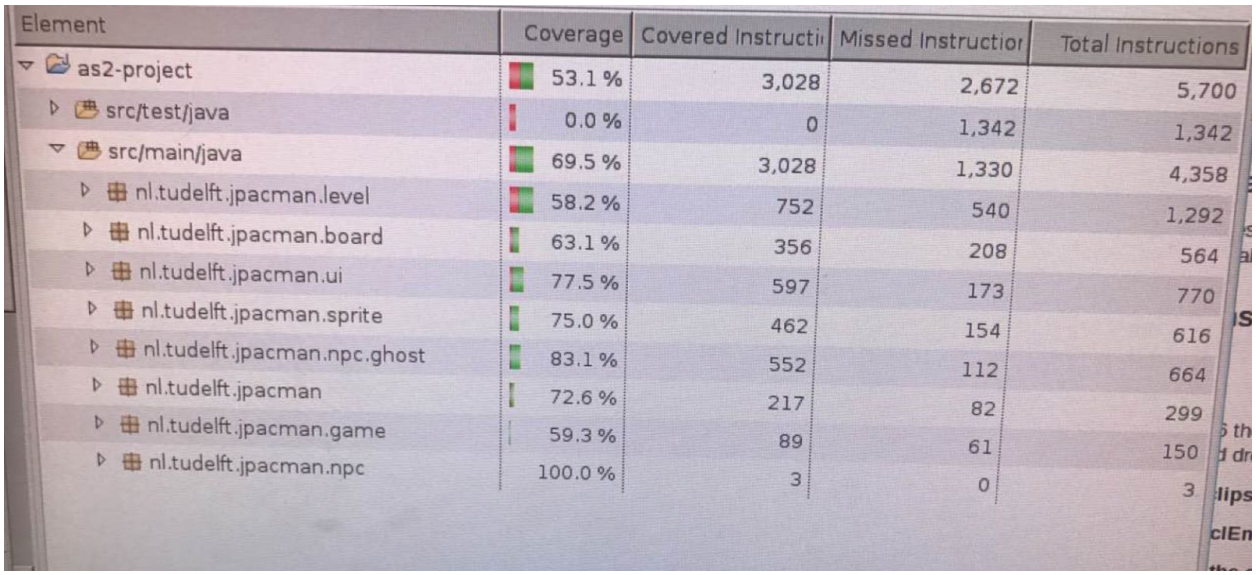
Student Number: 1000726814

Lecture:Tuesday 2 - 4 pm

part 1

1. Three scenarios:

sc1.open the game,press “start” button,let Pacman makes no move and close the window



Element	Coverage	Covered Instructi	Missed Instructior	Total Instructions
as2-project	53.1 %	3,028	2,672	5,700
src/test/java	0.0 %	0	1,342	1,342
src/main/java	69.5 %	3,028	1,330	4,358
nl.tudelft.jpacman.level	58.2 %	752	540	1,292
nl.tudelft.jpacman.board	63.1 %	356	208	564
nl.tudelft.jpacman.ui	77.5 %	597	173	770
nl.tudelft.jpacman.sprite	75.0 %	462	154	616
nl.tudelft.jpacman.npc.ghost	83.1 %	552	112	664
nl.tudelft.jpacman	72.6 %	217	82	299
nl.tudelft.jpacman.game	59.3 %	89	61	150
nl.tudelft.jpacman.npc	100.0 %	3	0	3

sc2 .open the game,press “start” button,let Pacman makes no move ,and the Pacman attacked by a ghost,and close the window

Element	Coverage	Covered Instructi	Missed Instructior	Total Instructions
as2-project	56.2 %	3,204	2,496	5,700
src/test/java	0.0 %	0	1,342	1,342
src/main/java	73.5 %	3,204	1,154	4,358
nl.tudelft.jpacman.level	64.0 %	827	465	1,292
nl.tudelft.jpacman.board	63.1 %	356	208	564
nl.tudelft.jpacman.ui	77.8 %	599	171	770
nl.tudelft.jpacman.sprite	78.7 %	485	131	616
nl.tudelft.jpacman	72.6 %	217	82	299
nl.tudelft.jpacman.npc.ghost	91.4 %	607	57	664
nl.tudelft.jpacman.game	73.3 %	110	40	150
nl.tudelft.jpacman.npc	100.0 %	3	0	3

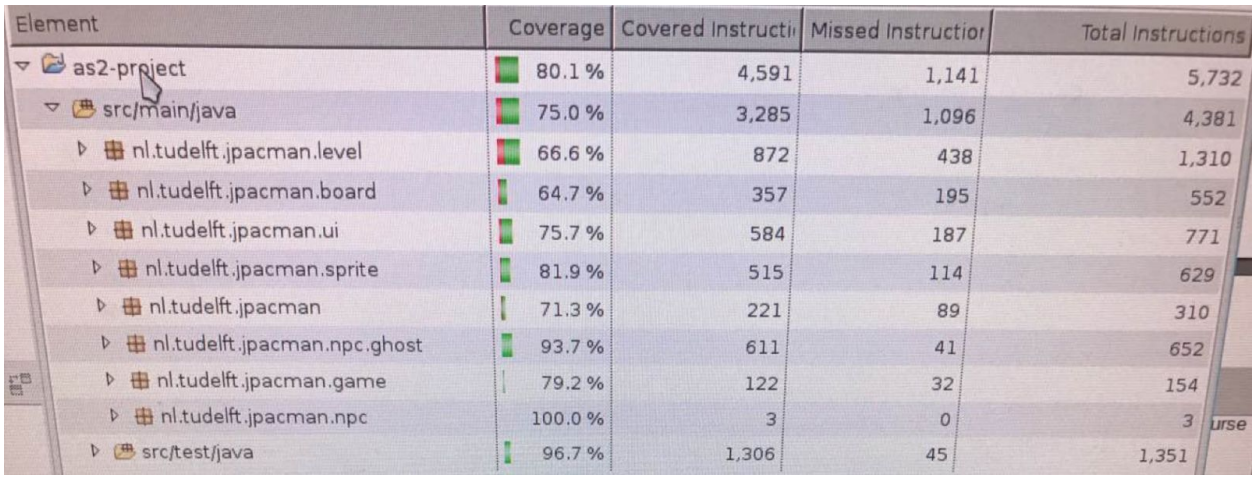
sc3. open the game,press “start” button,then move the Pacman 6 times to the right and eat the beans in the path,then let Pacman be attacked by a ghost,and close the window

Element	Coverage	Covered Instructi	Missed Instructior	Total Instructions
as2-project	57.6 %	3,286	2,414	5,700
src/test/java	0.0 %	0	1,342	1,342
src/main/java	75.4 %	3,286	1,072	4,358
nl.tudelft.jpacman.level	66.7 %	862	430	1,292
nl.tudelft.jpacman.board	65.4 %	369	195	564
nl.tudelft.jpacman.ui	79.9 %	615	155	770
nl.tudelft.jpacman.sprite	78.7 %	485	131	616
nl.tudelft.jpacman	74.9 %	224	75	299
nl.tudelft.jpacman.npc.ghost	91.7 %	609	55	664
nl.tudelft.jpacman.game	79.3 %	119	31	150
nl.tudelft.jpacman.npc	100.0 %	3	0	3

According to the coverage result graph,we can clearly see that the coverage of sc1 < the coverage of sc2 < the coverage of sc3 in general,since in the steps take in sc1 < the steps take in sc2<the steps take in sc3.Therefore,the method needed to use in sc1 < the method needed to use in sc2 < the method needed to use in sc1.

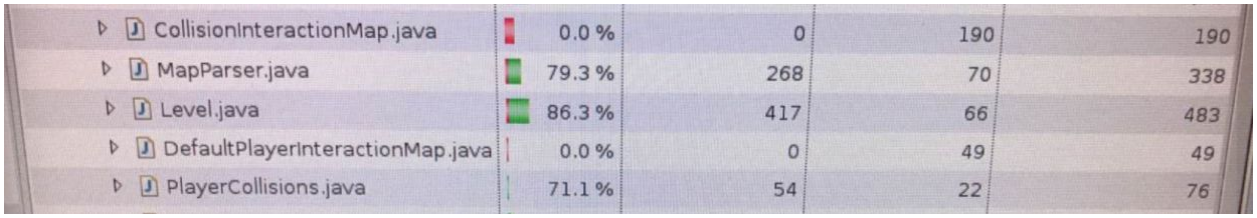
For sc1, we only start the game,and then close the window. Compare to sc2,at least methods about moving the the Pacman are less covered.
For sc2 ,we only start the game, not make Pacman move, and the Pacman attacked by a ghost.and then close the window. Compare to sc3,at least methods about eating the beans and update score are less covered.
For sc3,open the game and then move the Pacman to eat some beans,and then let Pacman be attacked by ghost.and then close the window.

2.



Element	Coverage	Covered Instructi	Missed Instructior	Total Instructions
as2-project	80.1 %	4,591	1,141	5,732
src/main/java	75.0 %	3,285	1,096	4,381
nl.tudelft.jpacman.level	66.6 %	872	438	1,310
nl.tudelft.jpacman.board	64.7 %	357	195	552
nl.tudelft.jpacman.ui	75.7 %	584	187	771
nl.tudelft.jpacman.sprite	81.9 %	515	114	629
nl.tudelft.jpacman	71.3 %	221	89	310
nl.tudelft.jpacman.npc.ghost	93.7 %	611	41	652
nl.tudelft.jpacman.game	79.2 %	122	32	154
nl.tudelft.jpacman.npc	100.0 %	3	0	3
src/test/java	96.7 %	1,306	45	1,351

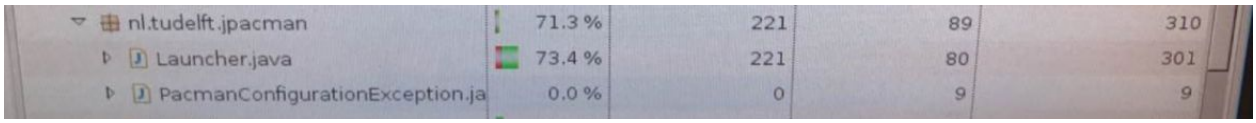
the coverage percentage is for main is 75.0%.
the least covered application classes are :CollisionInteractionMap, DefaultPlayerInteractionMap, PacmanConfigurationException. they are all 0% .
The test is adequate since the functionality of CollisionInteractionMap and DefaultPlayerInteractionMap are are replaced them by PlayerCollisions and only PlayerCollisions are used for executing the Pacman program. Therefore it is enough to just go through PlayerCollisions for the test.



CollisionInteractionMap.java	0.0 %	0	190	190
MapParser.java	79.3 %	268	70	338
Level.java	86.3 %	417	66	483
DefaultPlayerInteractionMap.java	0.0 %	0	49	49
PlayerCollisions.java	71.1 %	54	22	76

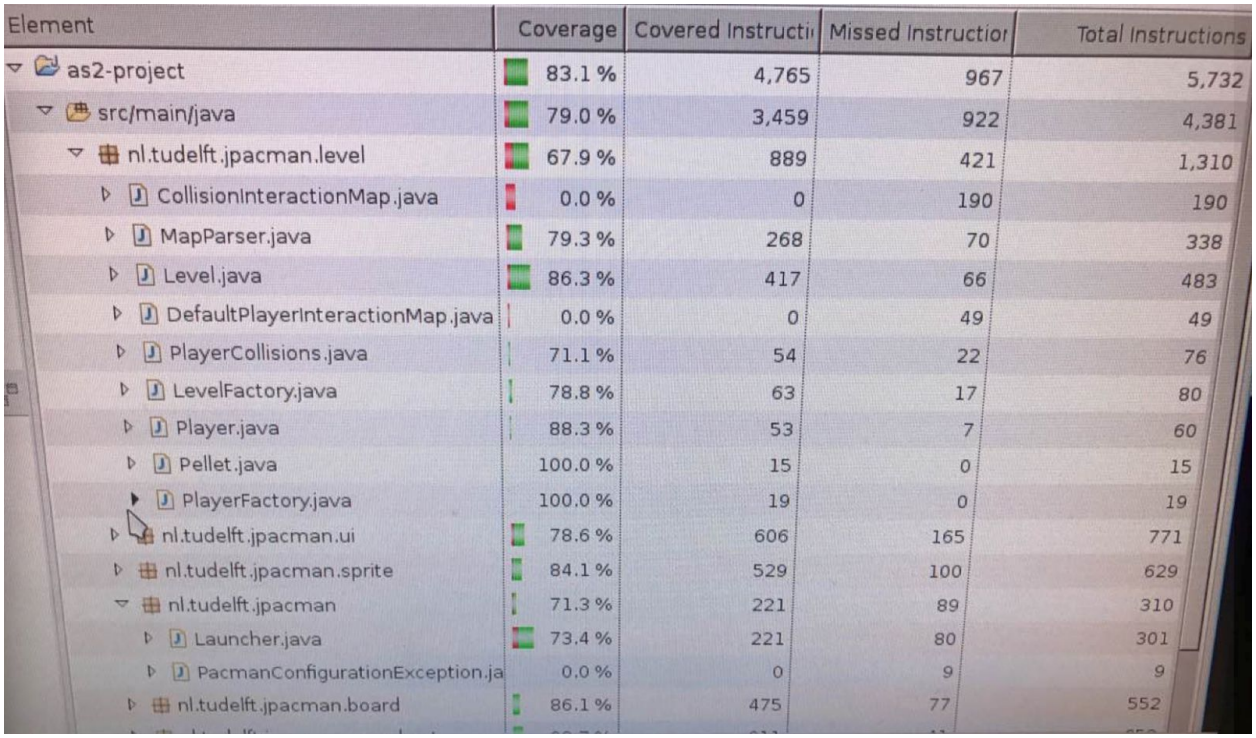
And for PacmanConfigurationException, it throws error message when the Pacman can not be loaded properly. However, our program loads properly everytime. And we can add a test case to cause some error to make the loading unsuccessful.For example,

give a wrong DEFAULT_MAP(wrong map file address) and it will casue getLevelMap() to fail and makeLevel will cacth the IOException Error.



nl.tudelft.jpacman	71.3 %	221	89	310
Launcher.java	73.4 %	221	80	301
PacmanConfigurationException.ja	0.0 %	0	9	9

3.

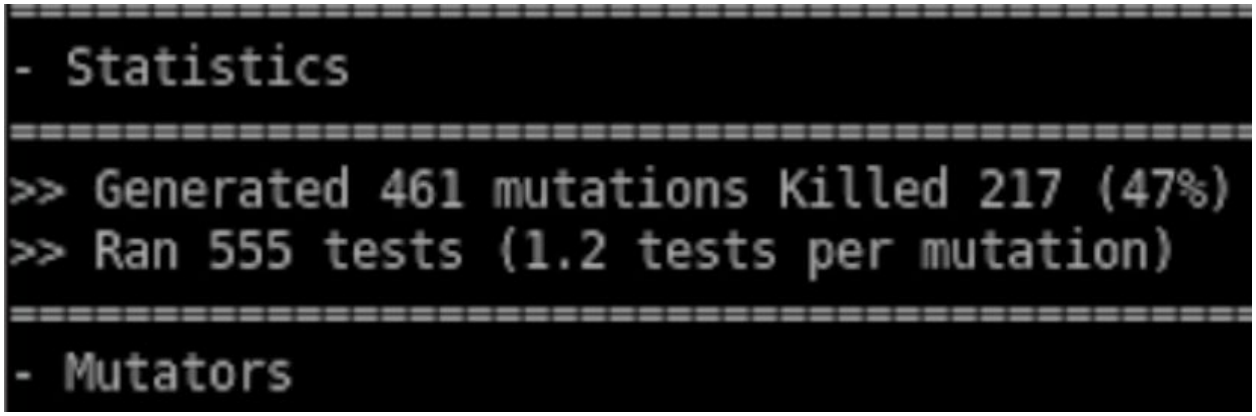


Element	Coverage	Covered Instructi	Missed Instructior	Total Instructions
as2-project	83.1 %	4,765	967	5,732
src/main/java	79.0 %	3,459	922	4,381
nl.tudelft.jpacman.level	67.9 %	889	421	1,310
CollisionInteractionMap.java	0.0 %	0	190	190
MapParser.java	79.3 %	268	70	338
Level.java	86.3 %	417	66	483
DefaultPlayerInteractionMap.java	0.0 %	0	49	49
PlayerCollisions.java	71.1 %	54	22	76
LevelFactory.java	78.8 %	63	17	80
Player.java	88.3 %	53	7	60
Pellet.java	100.0 %	15	0	15
PlayerFactory.java	100.0 %	19	0	19
nl.tudelft.jpacman.ui	78.6 %	606	165	771
nl.tudelft.jpacman.sprite	84.1 %	529	100	629
nl.tudelft.jpacman	71.3 %	221	89	310
Launcher.java	73.4 %	221	80	301
PacmanConfigurationException.ja	0.0 %	0	9	9
nl.tudelft.jpacman.board	86.1 %	475	77	552

Generally coverage percentage increases , since some assertions are enabled after adding -ea as VM arguments, more line will be covered by the tests.

part 2

1.

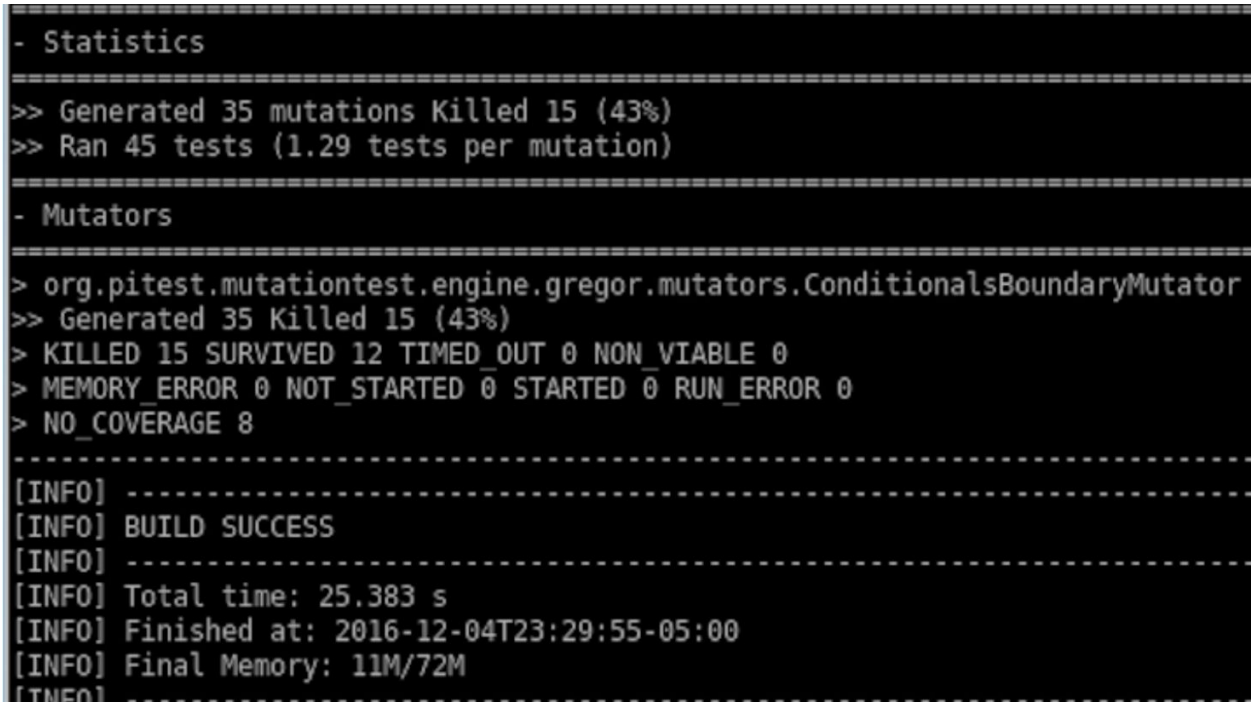


the mutation coverage is 47%,it is smaller than line coverage,
since mutation coverage is a very advanced which tests features and quality of the test cases, and line coverage is very basic which simply test how many lines are executed. Therefore, more underlying problems of code will be detected by the mutation testing, and higher the mutation coverage means more number of mutants killed and it means less underlying problems problems which can be gauged as better quality of the test cases.

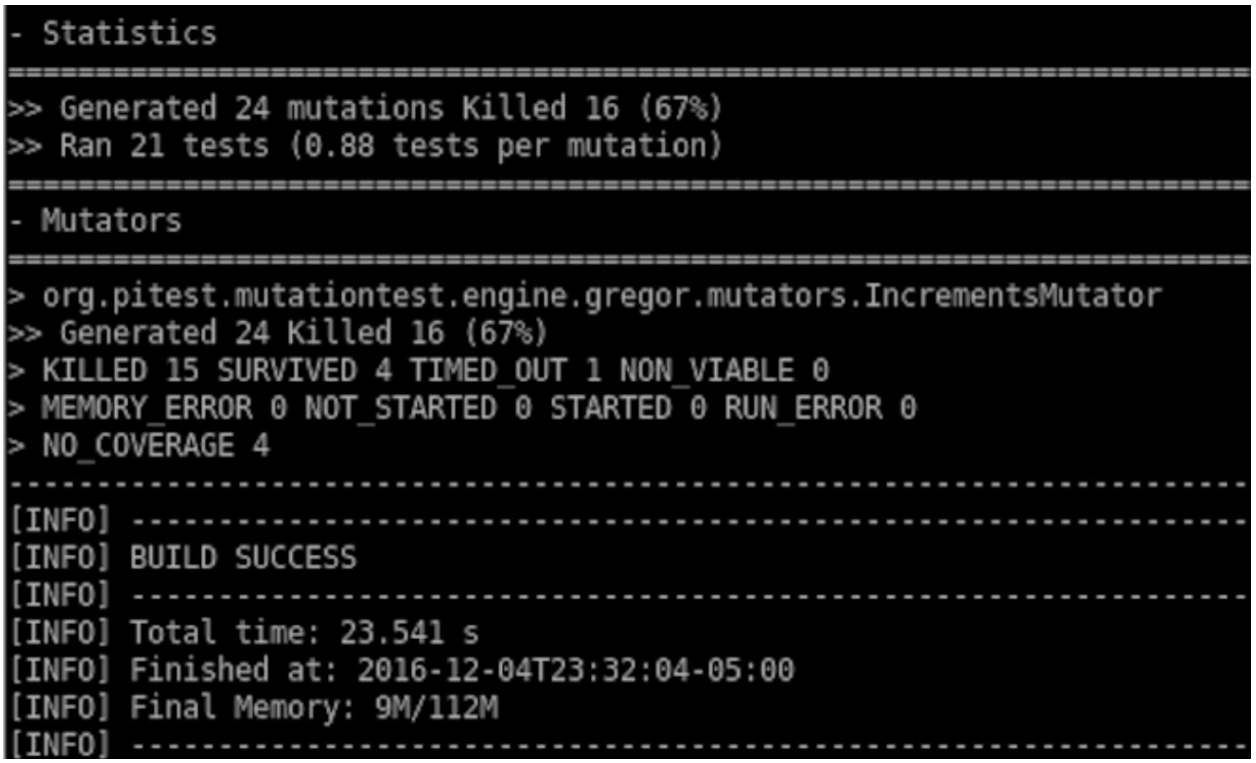
```
> org.pitest.mutationtest.engine.gregor.mutators.ConditionalsBoundaryMutator
>> Generated 35 Killed 15 (43%)
> KILLED 15 SURVIVED 12 TIMED_OUT 0 NON_VIABLE 0
> MEMORY_ERROR 0 NOT_STARTED 0 STARTED 0 RUN_ERROR 0
> NO_COVERAGE 8
--------------------------------------------------------------------------------
> org.pitest.mutationtest.engine.gregor.mutators.IncrementsMutator
>> Generated 24 Killed 16 (67%)
> KILLED 15 SURVIVED 4 TIMED_OUT 1 NON_VIABLE 0
> MEMORY_ERROR 0 NOT_STARTED 0 STARTED 0 RUN_ERROR 0
> NO_COVERAGE 4
--------------------------------------------------------------------------------
> org.pitest.mutationtest.engine.gregor.mutators.VoidMethodCallMutator
>> Generated 95 Killed 22 (23%)
> KILLED 22 SURVIVED 44 TIMED_OUT 0 NON_VIABLE 0
> MEMORY_ERROR 0 NOT_STARTED 0 STARTED 0 RUN_ERROR 0
> NO_COVERAGE 29
--------------------------------------------------------------------------------
> org.pitest.mutationtest.engine.gregor.mutators.ReturnValsMutator
>> Generated 137 Killed 81 (59%)
> KILLED 81 SURVIVED 22 TIMED_OUT 0 NON_VIABLE 0
> MEMORY_ERROR 0 NOT_STARTED 0 STARTED 0 RUN_ERROR 0
> NO_COVERAGE 34
--------------------------------------------------------------------------------
> org.pitest.mutationtest.engine.gregor.mutators.MathMutator
>> Generated 34 Killed 16 (47%)
> KILLED 15 SURVIVED 18 TIMED_OUT 1 NON_VIABLE 0
> MEMORY_ERROR 0 NOT_STARTED 0 STARTED 0 RUN_ERROR 0
> NO_COVERAGE 0
```

2.

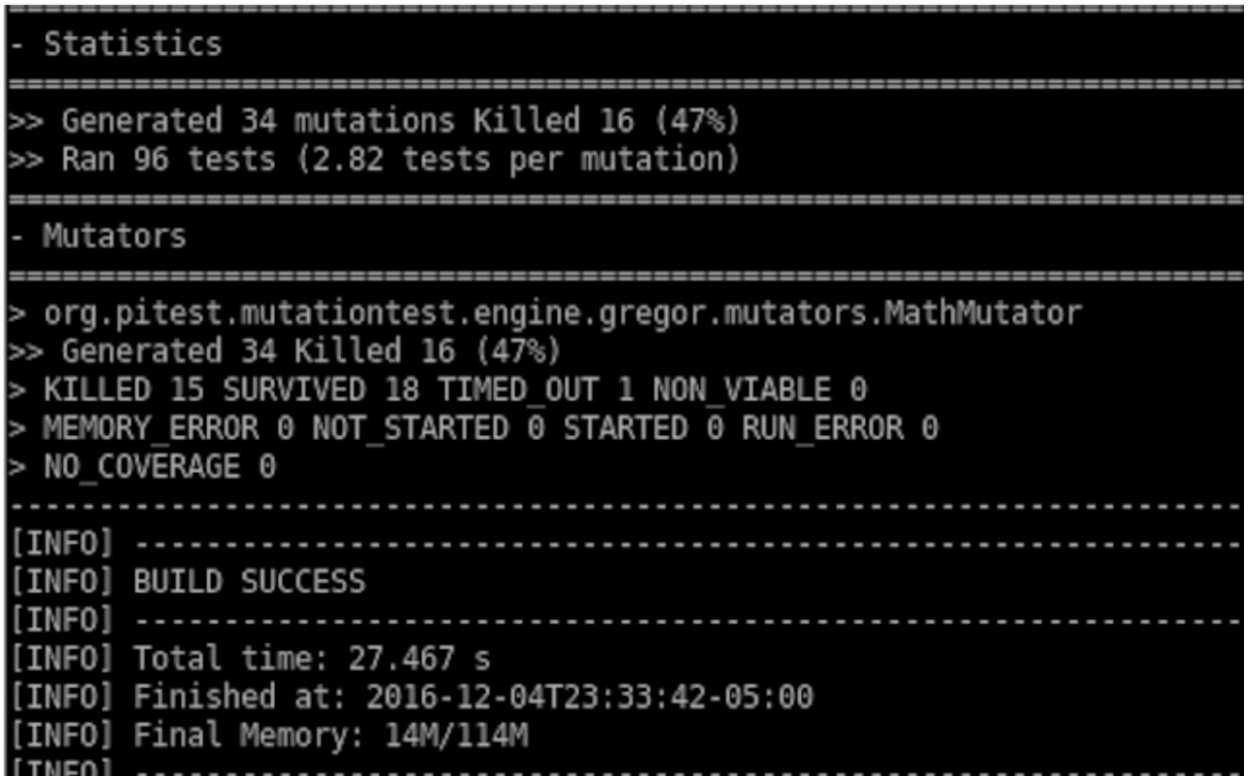
“Conditionals Boundary Mutator”:



“Increments Mutator”:



“Math Mutator”:

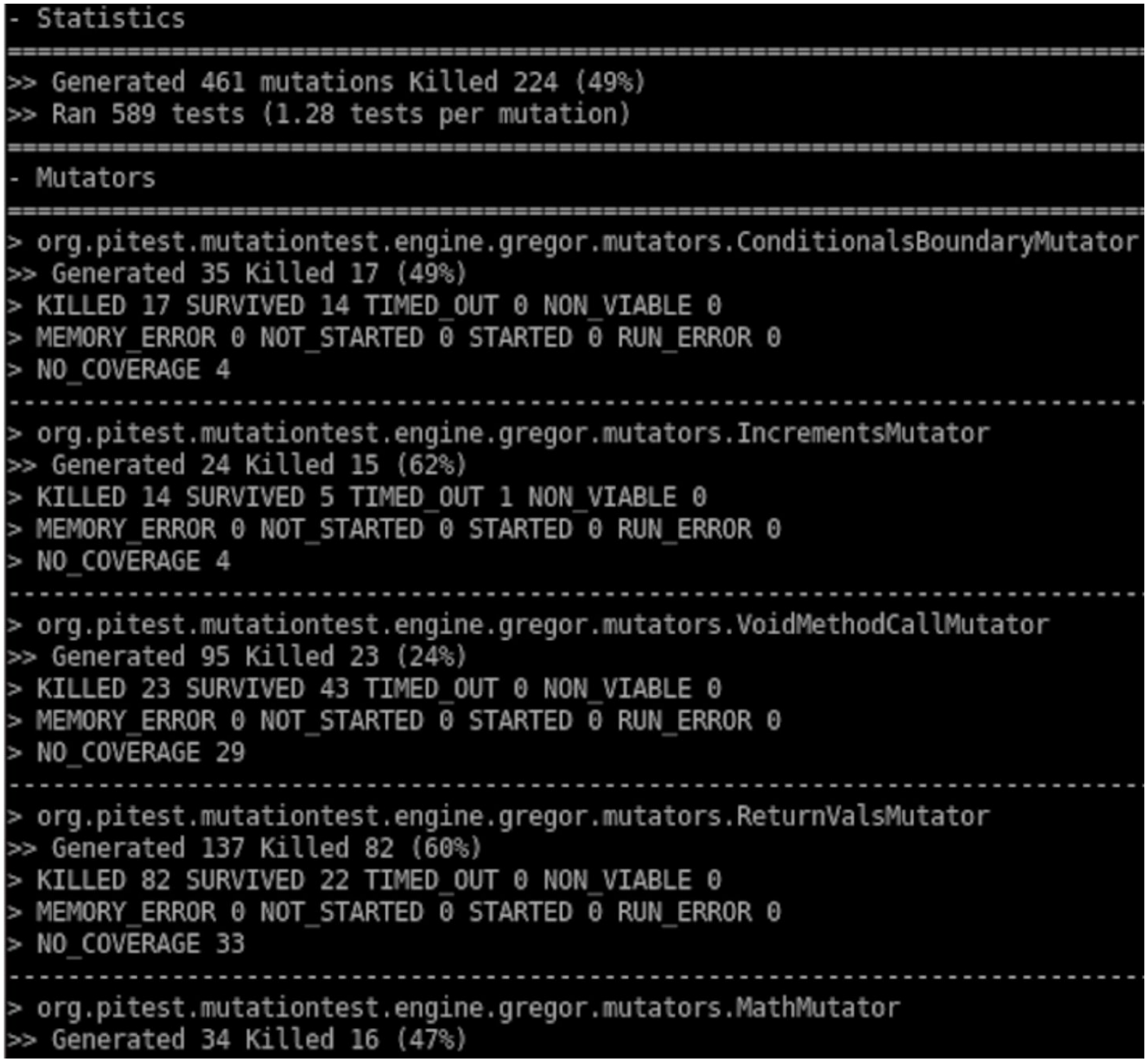


By definitions,
The increments mutator will mutate increments, decrements and assignment increments and decrements of local variables. It will replace increments with decrements and vice versa.For example, for integer i, i++ change to i-- and i-- change to i++.
The math mutator replaces binary arithmetic operations for either integer or floating-point arithmetic with another operation. For example, for integer i and b, i*a change to i/a and i change to i++.And unlike increments, it also test for non-local variable.
And Conditional Boundary Mutators are the mutants of relational operators.For example, from a > b to a >=b and a <=b to a < b.

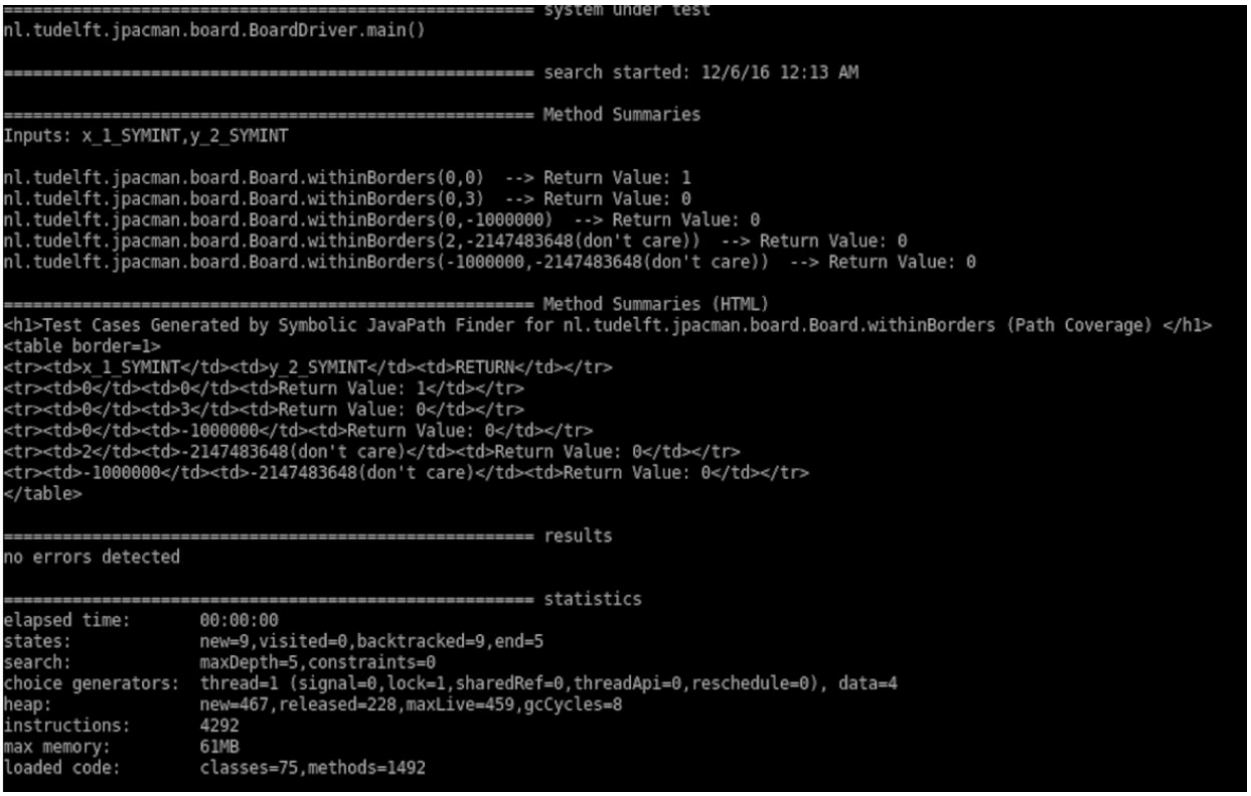
In the Pacman program, the “Conditionals Boundary Mutator” mutation score is 43% and “Math Mutator” mutation score is 47% which are lower than incremental mutator mutation coverage 67%, and it reflects that in this program, higher percentage of incremental mutators causes more unit tests from our test/java to fail.

2.3
a method testWithinBorders() to test the withinBorders(int, int) is added in the BoardTest.java file.
testWithinBorders() will test the example withinBorders(0,1), and during the mutation test, some conditional boundary mutators will be killed and increase the mutation score, For example, “x >= 0 && x < getWidth() && y >= 0 && y < getHeight(); “will be mutated to” return x > 0 && x < getWidth() && y > 0 && y < getHeight(); ”, and this mutant will be killed, since 0 >= 0 && 0 < getWidth() && 1 >= 0 && 1 < getHeight() and 0 > 0 && 0 < getWidth() && 1 > 0 && 1 < getHeight() is false.

As the results shown below , the conditional boundary mutation coverage improves from 43% to 49%.The overall coverage improves.



part 3
1 and 2.
//test cases generated below:




```
package nl.tudelft.jpacman.board;

import static org.junit.Assert.*;
import static org.mockito.Mockito.mock;

import org.junit.Before;
import org.junit.Test;

public class JPFBoardTest {
	private Board board;

	private final Square x0y0 = mock(Square.class);
	private final Square x0y1 = mock(Square.class);
	private final Square x0y2 = mock(Square.class);
	private final Square x1y0 = mock(Square.class);
	private final Square x1y1 = mock(Square.class);
	private final Square x1y2 = mock(Square.class);

	private static final int MAX_WIDTH = 2;
```

```
	private static final int MAX_HEIGHT = 3;

	/**
	 * Setup a board that can be used for testing.
	 */
	@Before
	public void setUp() {
		Square[][] grid = new Square[MAX_WIDTH][MAX_HEIGHT];
		grid[0][0] = x0y0;
		grid[0][1] = x0y1;
		grid[0][2] = x0y2;
		grid[1][0] = x1y0;
		grid[1][1] = x1y1;
		grid[1][2] = x1y2;
		board = new Board(grid);
	}


//testwithinBorders() with return value of true , on board
	@Test
	public void testWithinBorders1(){
		assertEquals(true, board.withinBorders(0, 0));
	}


	 //test withinBorders() with return value of false, off board

	@Test
	public void testWithinBorders2(){
		assertEquals(false, board.withinBorders(0, 3));
	}


	 //test withinBorders() with return value of false, off board
	@Test
	public void testWithinBorders3(){
		assertEquals(false, board.withinBorders(0, -1000000));
	}
```

```
	 //test withinBorders() with return value of false , off board
	@Test
	public void testWithinBorders4(){
		assertEquals(false, board.withinBorders(2,-2147483648));
	}

	//test withinBorders() with return value of false, off board
	@Test
	public void sasa(){
		assertEquals(false, board.withinBorders(-1000000,-2147483648));
	}
//part3.question2
   //the test only contains off board and on board but no in board
   //test withinBorders() with return value of true , in board
	@Test
	public void testWithinBorders6(){
		assertEquals(true, board.withinBorders(1,1));
	}
}
```

Part4:
Question 1: Please see the code changes in Level.java, Game.java as well as PacManUiBuilder.java.

The simply nested relationship diagram is like:
stop/start NPCs functions are within Level.java ----> they are implemented and are able to called within Game.java ----> they are eventually able to called at PacManUiBuilder.java

Question 2: No, we are unable to catch any bugs by our implementation of the test suit.

Question 3:
We basically used three different techniques and read the corresponding analysis results.

1, Test Cases
Command :
'Create a new file called Part4Test.java'
mvn compile
mvn test -Dtest=Part4Test
Documentation:
We wrote a new file that is named Part4Test.java, which is located at Assignment2/as2-project/src/test/java/nl/tudelft/jpacman, it is basically junit test. We created some test cases by setting some movement of the player, pressing the (Un)Freeze button , and made some assertions of boolen, value(of scores, isAlive() etc.) to assert whether the actual output match the theory. We made test cases from different scenarios,(1) before pressing the freeze button for the first time, the scores should still be generated if hitting the yellow dots. (2)and until the player spontaneously encounters the ghost, the player's status will be alive and is able to keep earning scores while hitting the yellow dots.(3) if the player encounters the ghost, the isAlive() should be False and the player is no longer able to get further scores. (4) Now the (Un)Freeze button is pressed for the second time, further movements would not generate any scores(player dead), the scores remain same as the previous state somewhere. Thus, our test cases are able to partially ensure that the newly implemented code is correct and have no bugs.

2, Line/Code Coverage
Command:
mvn compile
mvn test -Dtest=Part4Test
mvn jacoco:report
Documentation:
After certain codes are newly added to three java files. We run the jacoco test to test the code/line coverages.We looked at the reports that locate at Assignment2/as2-project/target/site/jacoco.

(1)We first looked at the file called Game.java.html, almost every line of codes are executed while running the test, including the one we added to implement the freezing button. This great portion of green highlight area indicates a good line/code coverage of this certain java file, Game.java.
(2) We then looked at the file called Level.java.html, compared to the former jacoco report while using LaunchSmokerTest as the argument of mvn test command, now there are even greater green hightlighted area now,
since previously the startNPCs() and stopNPCs() are red(not executed at all) but now fully executed. It indicates a good efficiency of using the code in this java file(Level.java) and our implementation of the (Un)Freeze button improves the code/line coverage percentage.
(3)We last looked at the file called PacManUiBuilder.java.html, the highlighted pattern almost remains unchanged. But his file is actually where we added the button and handler the onclick event of the (Un)Freeze button. It is because within the test suit we created(Part4Test.java), we did not call any function through the PacManUiBuilder package(But Game.java instead). I think this is one of the defect of making such a test suit.

Thus,the improvement coule be done by calling function through PacManUiBuilder, in this way, we can potentially detect more bugs due to implementation as well as 'increase' code coverages to make further analysis.

3, Mutation Testing
Command:
'Modification of pom.xml'
mvn compile
mvn test -Dtest=Part4Test
mvn org.pitest:pitest-maven:mutationCoverage
Documentation:
We modified the pom.xml which is located at as2-project by modifying the revelant plugin element. We modified the targetTest and targetClass's parameter to be nl.tudelft.jpacman.PacManUiBuilder* and modified the mutators of that plug in to only include conditionals. The results we 35 conditional boundary mutators generated and 15 killed(43%).