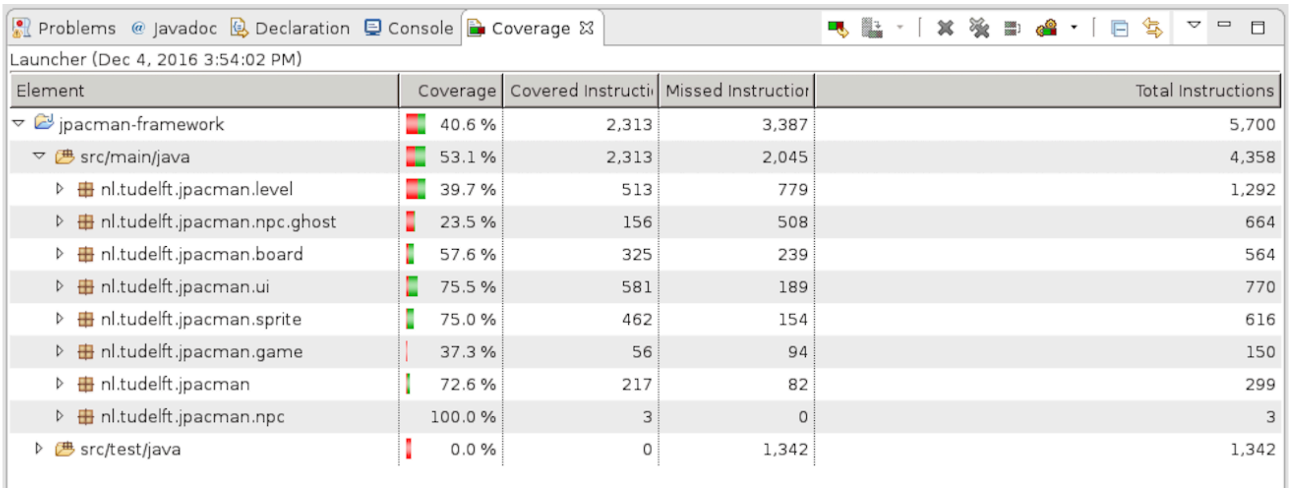
csc410 assignment2
Zhousenye Liu 1001475780

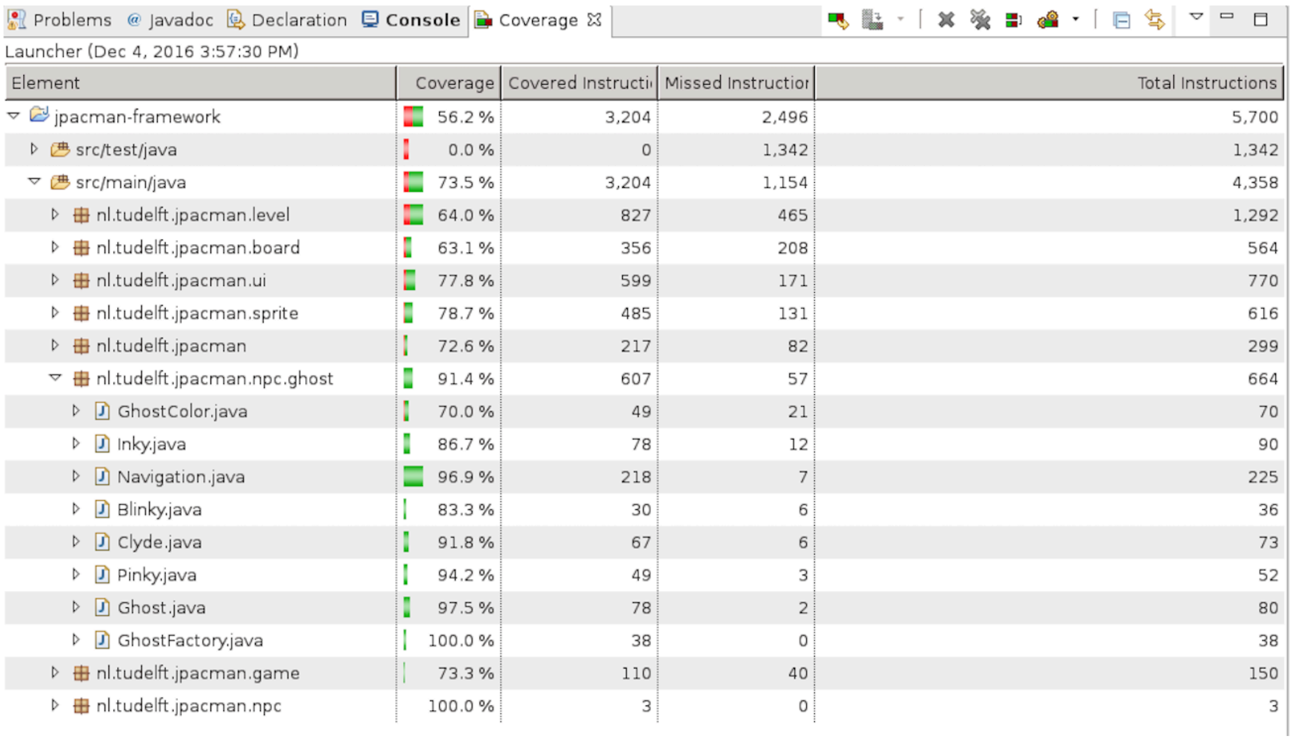
before read my report , I have to say something about my zip files .
Because of unknown system error (exceed max shortage of my cdf machine) , when I tried to do part 4 on my original project files I had many errors , which causes me cannot go forward . So I have to download another project file. So I submitted 2 project file , one for part1-part3 and one for part4 , please pay attention . (also the code I write for is working with my friend, we used the same idea)
part 1
1. Here are three scenarios I come up with and there coverage analysis separately :
a. do nothing (open the game then close the window directly)


Problems @ Javadoc Declaration Console Coverage

Launcher (Dec 4, 2016 3:54:02 PM)

Element	Coverage	Covered Instructi	Missed Instructior	Total Instructions
jpacman-framework	40.6 %	2,313	3,387	5,700
src/main/java	53.1 %	2,313	2,045	4,358
nl.tudelft.jpacman.level	39.7 %	513	779	1,292
nl.tudelft.jpacman.npc.ghost	23.5 %	156	508	664
nl.tudelft.jpacman.board	57.6 %	325	239	564
nl.tudelft.jpacman.ui	75.5 %	581	189	770
nl.tudelft.jpacman.sprite	75.0 %	462	154	616
nl.tudelft.jpacman.game	37.3 %	56	94	150
nl.tudelft.jpacman	72.6 %	217	82	299
nl.tudelft.jpacman.npc	100.0 %	3	0	3
src/test/java	0.0 %	0	1,342	1,342

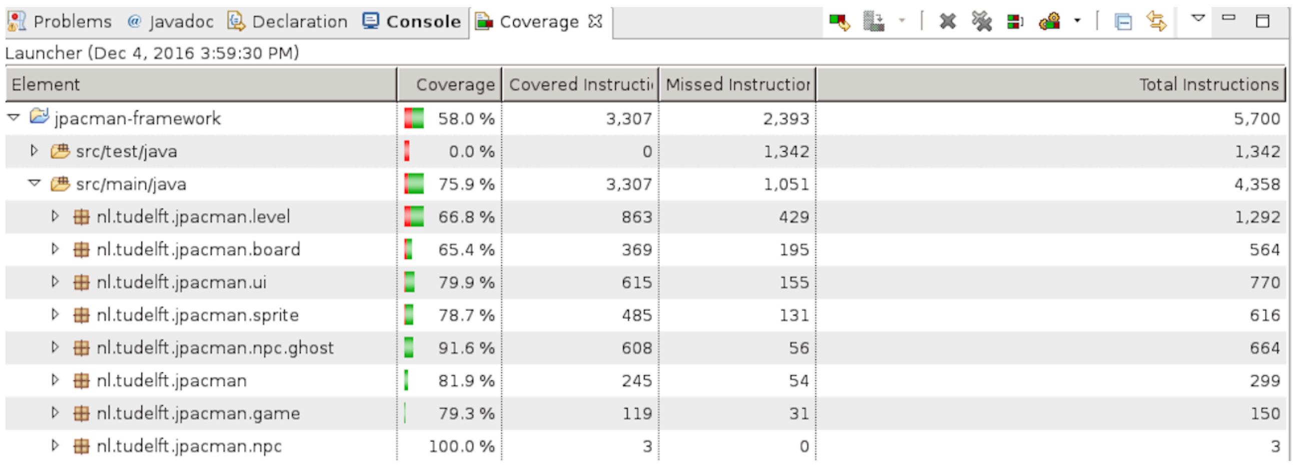
b. do not move Pacman , then let it be attacked by goats (eat nothing)


Problems @ Javadoc Declaration Console Coverage

Launcher (Dec 4, 2016 3:57:30 PM)

Element	Coverage	Covered Instructi	Missed Instructior	Total Instructions
jpacman-framework	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
GhostColor.java	70.0 %	49	21	70
Inky.java	86.7 %	78	12	90
Navigation.java	96.9 %	218	7	225
Blinky.java	83.3 %	30	6	36
Clyde.java	91.8 %	67	6	73
Pinky.java	94.2 %	49	3	52
Ghost.java	97.5 %	78	2	80
GhostFactory.java	100.0 %	38	0	38
nl.tudelft.jpacman.game	73.3 %	110	40	150
nl.tudelft.jpacman.npc	100.0 %	3	0	3

c. eat something then let Pacman be attacked by goest

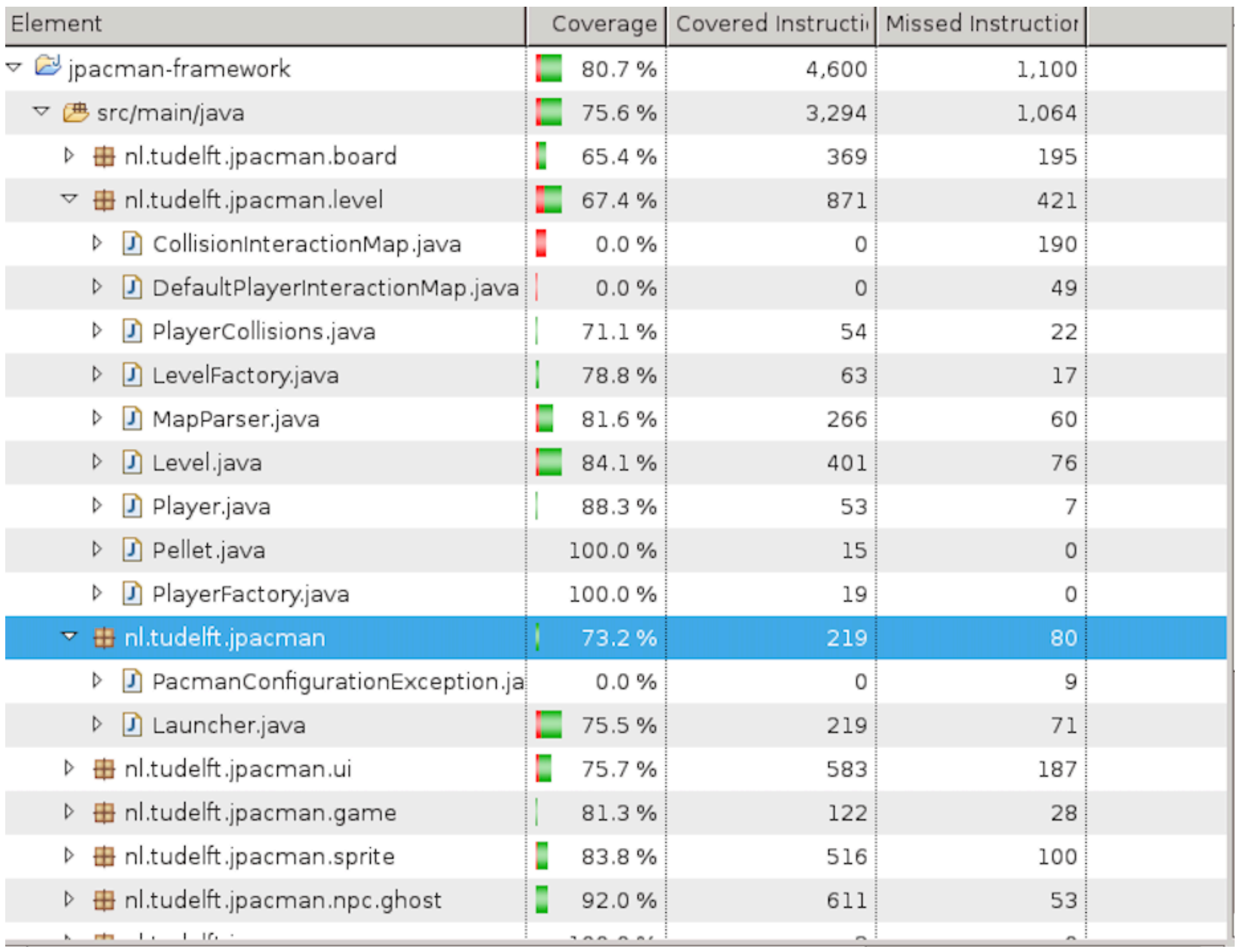


Launcher (Dec 4, 2016 3:59:30 PM)

Element	Coverage	Covered Instructi	Missed Instructior	Total Instructions
jpacman-framework	58.0 %	3,307	2,393	5,700
src/test/java	0.0 %	0	1,342	1,342
src/main/java	75.9 %	3,307	1,051	4,358
nl.tudelft.jpacman.level	66.8 %	863	429	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.npc.ghost	91.6 %	608	56	664
nl.tudelft.jpacman	81.9 %	245	54	299
nl.tudelft.jpacman.game	79.3 %	119	31	150
nl.tudelft.jpacman.npc	100.0 %	3	0	3

We can see there is a huge difference among 3 scenarios. Obviously , the scenario1 : do nothing , has least coverage . the reason is very simple and straight: we did nothing in this step: neither ghost nor Pacman move(that is why the ghost package has only 23.5% coverage) , Pacman eats nothing.
As for the scenario2 , it has better coverage since we start the game (click the button) and make ghost move.(that is why ghost coverage has increased to 91.4% hugely)
Finally , for scenario3 , it has best coverage since we also move the Pacman to eat something , which causes more method to be executed.

2.



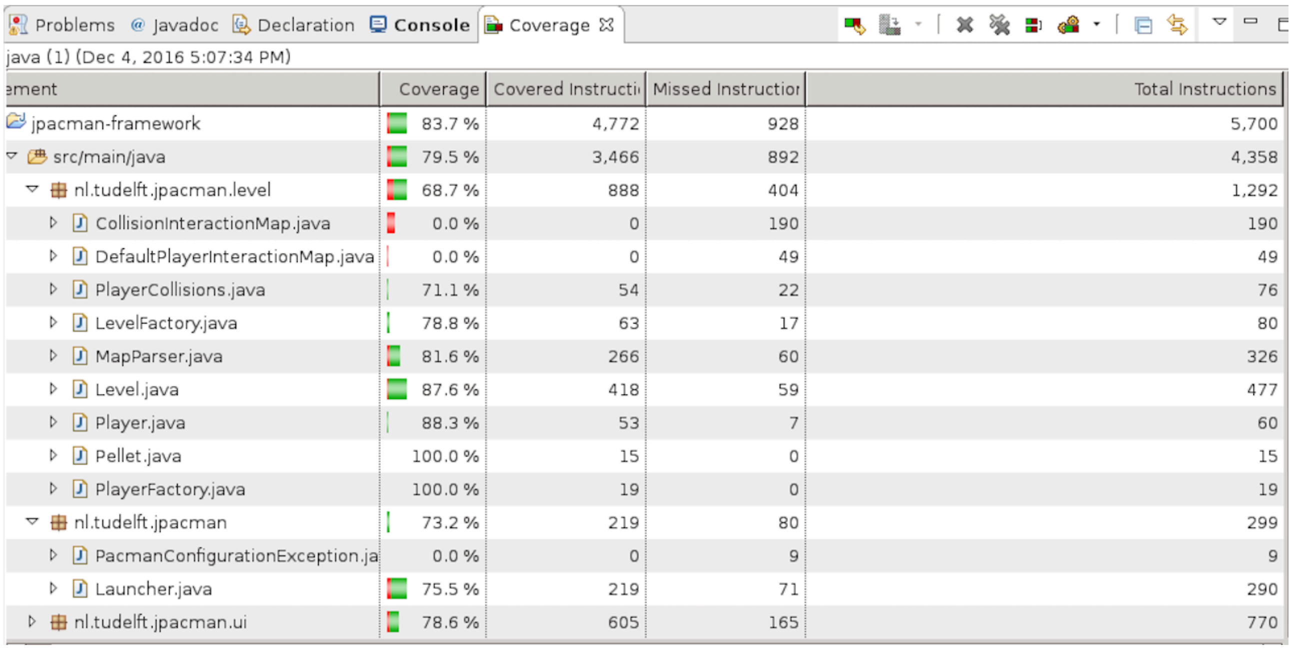
Element	Coverage	Covered Instructi	Missed Instructior
jpacman-framework	80.7 %	4,600	1,100
src/main/java	75.6 %	3,294	1,064
nl.tudelft.jpacman.board	65.4 %	369	195
nl.tudelft.jpacman.level	67.4 %	871	421
CollisionInteractionMap.java	0.0 %	0	190
DefaultPlayerInteractionMap.java	0.0 %	0	49
PlayerCollisions.java	71.1 %	54	22
LevelFactory.java	78.8 %	63	17
MapParser.java	81.6 %	266	60
Level.java	84.1 %	401	76
Player.java	88.3 %	53	7
Pellet.java	100.0 %	15	0
PlayerFactory.java	100.0 %	19	0
nl.tudelft.jpacman	73.2 %	219	80
PacmanConfigurationException.ja	0.0 %	0	9
Launcher.java	75.5 %	219	71
nl.tudelft.jpacman.ui	75.7 %	583	187
nl.tudelft.jpacman.game	81.3 %	122	28
nl.tudelft.jpacman.sprite	83.8 %	516	100
nl.tudelft.jpacman.npc.ghost	92.0 %	611	53

this is coverage analysis I get from test.
so the coverage percentage is for main is 75.6%.
the least covered application classes are :CollisionInteractionMap, DefaultPlayerInteractionMap, PacmanConfigurationException separately , they are all 0% .

Firstly , we do not need to care about CollisionInteractionMap and DefaultPlayerInteractionMap because we do not need to use them(they are not used anywhere), we have replaced them by PlayerCollisions. (That is why methods in DefaultPlayerInteractionMap and CollisionInteractionMap are not used)

Secondly, we also do not need to worry about PacmanConfigurationException, because Pacman can always be loaded successfully. (if we want this class be covered , we can just delete the board.txt file , which can cause IOException when calling makelevel()in Launcher.java)

3.



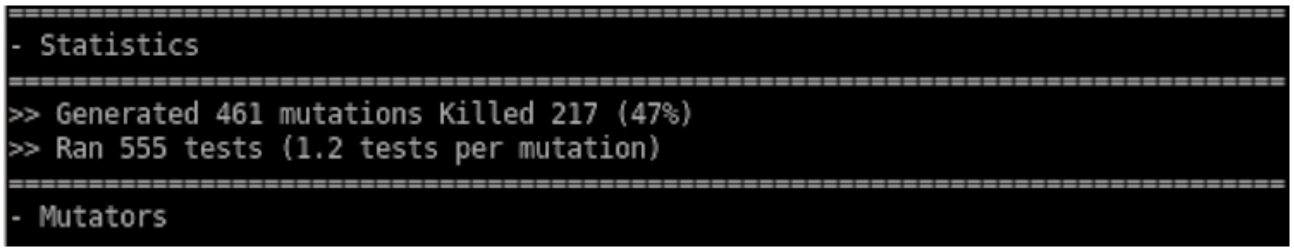
Problems | Javadoc | Declaration | Console | Coverage

java (1) (Dec 4, 2016 5:07:34 PM)

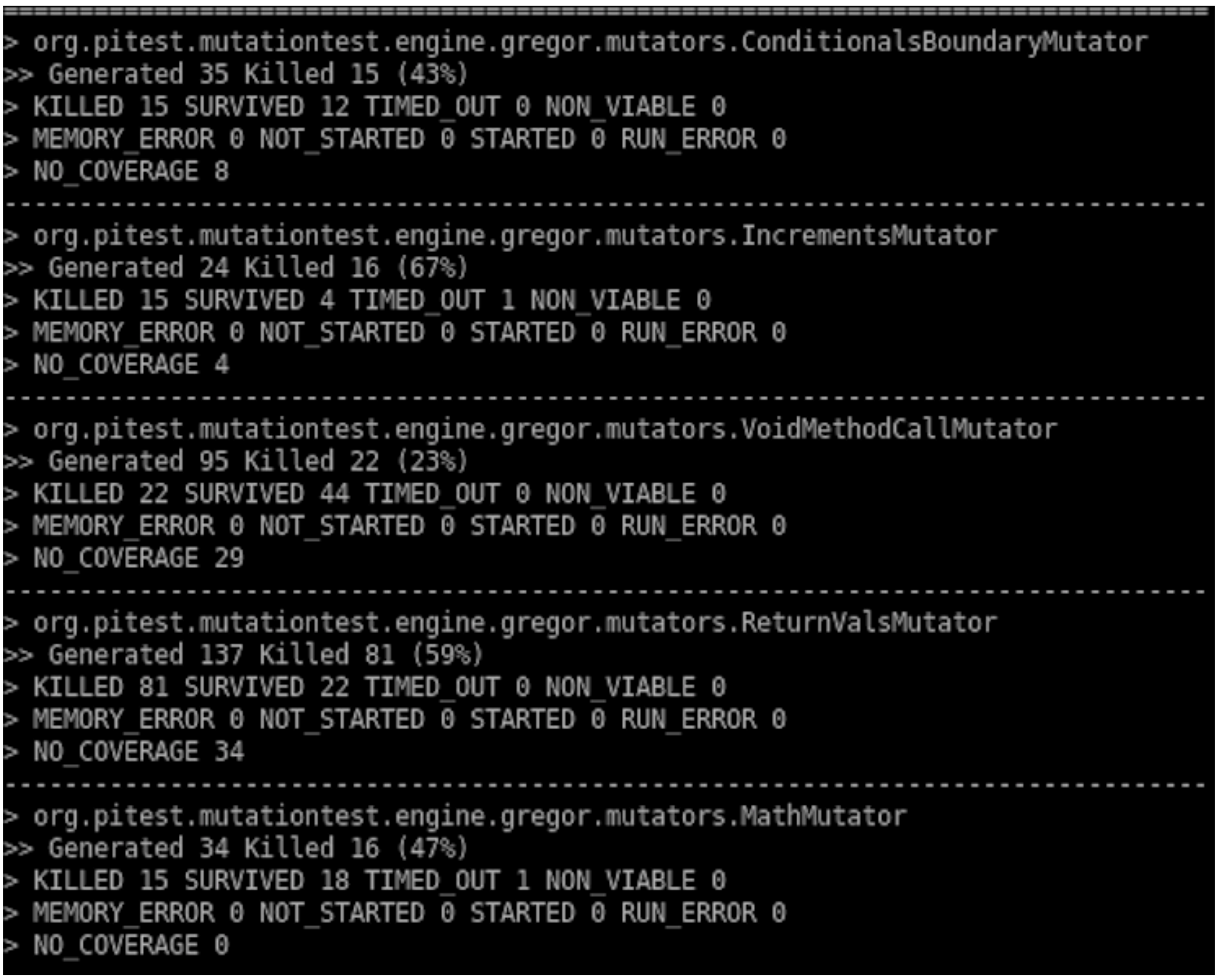
ement	Coverage	Covered Instructi	Missed Instructior	Total Instructions
jpacman-framework	83.7 %	4,772	928	5,700
src/main/java	79.5 %	3,466	892	4,358
nl.tudelft.jpacman.level	68.7 %	888	404	1,292
CollisionInteractionMap.java	0.0 %	0	190	190
DefaultPlayerInteractionMap.java	0.0 %	0	49	49
PlayerCollisions.java	71.1 %	54	22	76
LevelFactory.java	78.8 %	63	17	80
MapParser.java	81.6 %	266	60	326
Level.java	87.6 %	418	59	477
Player.java	88.3 %	53	7	60
Pellet.java	100.0 %	15	0	15
PlayerFactory.java	100.0 %	19	0	19
nl.tudelft.jpacman	73.2 %	219	80	299
PacmanConfigurationException.ja	0.0 %	0	9	9
Launcher.java	75.5 %	219	71	290
nl.tudelft.jpacman.ui	78.6 %	605	165	770

after we add -ea to VM arguments , we can see the overall coverage percentage increases slightly ,   that is because some asserts are executed . (for example , the asserts on line 95-97 in Level.java)

part 2
1.

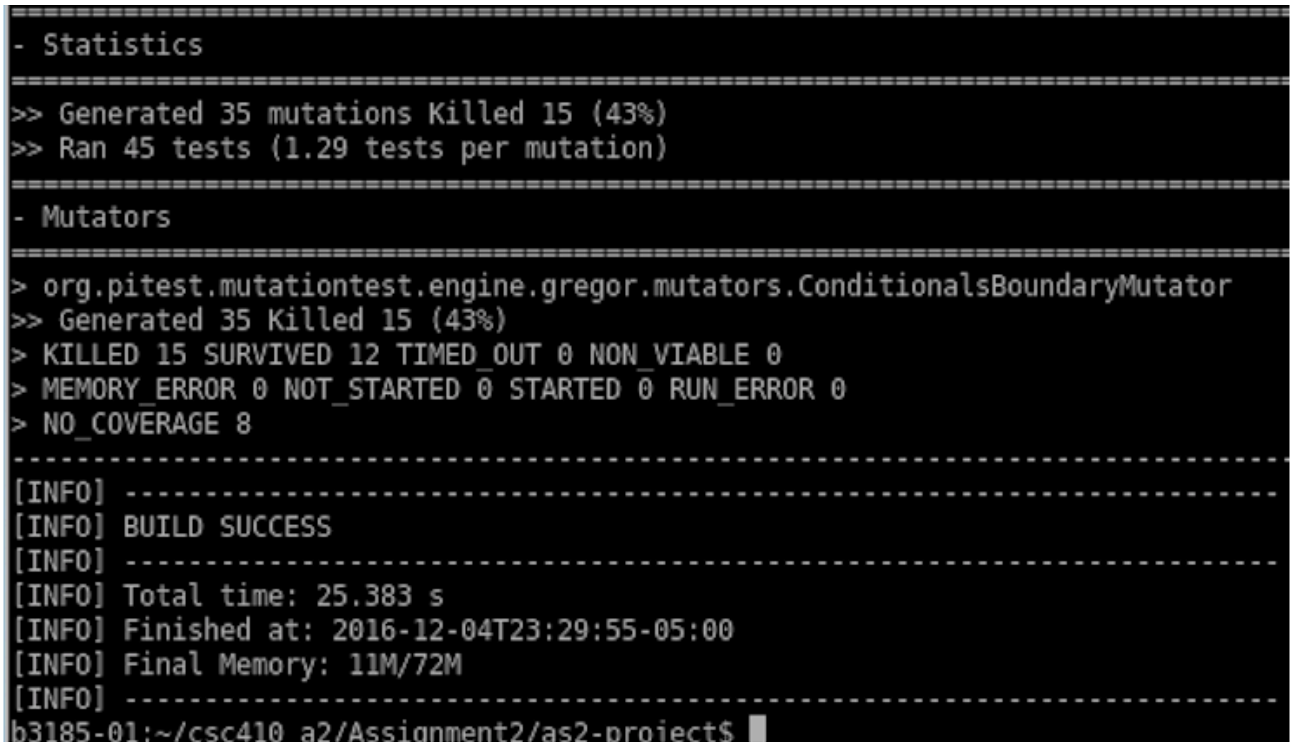


```
================================================================================
- Statistics
================================================================================
>> Generated 461 mutations Killed 217 (47%)
>> Ran 555 tests (1.2 tests per mutation)
================================================================================
- Mutators
```



```
================================================================================
> 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
```

this is the result I got when running pit to do mutation test. the result is 47%.
this is percentage is smaller than the result we got when using line coverage.
the reason is simple , mutation coverage is much more advanced than line coverage .
more advanced a test is , more potential problems we can find, a lower percentage always represents that more potential problems exist.
(the picture below is the second half of result which includes every mutator's coverage)

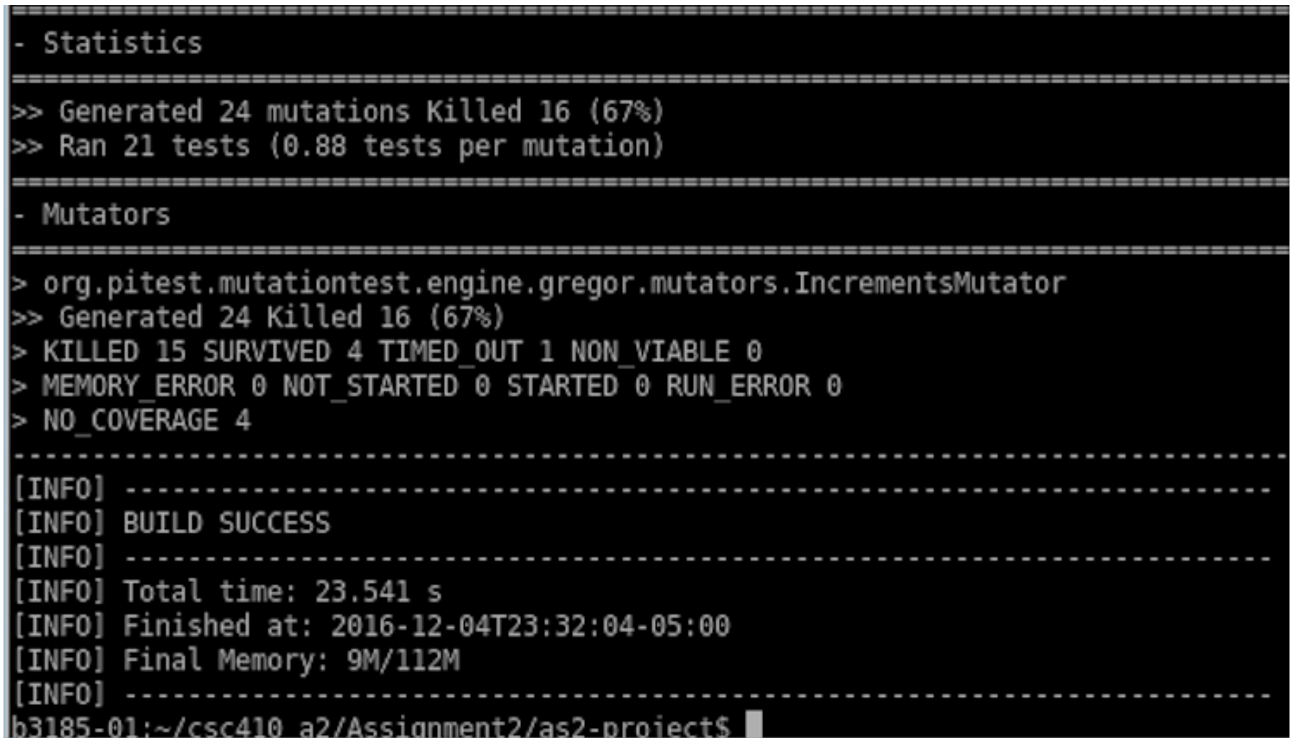


```
================================================================================
- Statistics
================================================================================
>> Generated 35 mutations Killed 15 (43%)
>> Ran 45 tests (1.29 tests per mutation)
================================================================================
- Mutators
================================================================================
> 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
--------------------------------------------------------------------------------
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 25.383 s
[INFO] Finished at: 2016-12-04T23:29:55-05:00
[INFO] Final Memory: 11M/72M
[INFO] ------------------------------------------------------------------------
b3185-01:~/csc410_a2/Assignment2/as2-project$
```

the coverage of “Conditionals Boundary Mutator”, “Increments Mutator”, and “Math Mutator” are 43%, 67% and 47%, respectively.

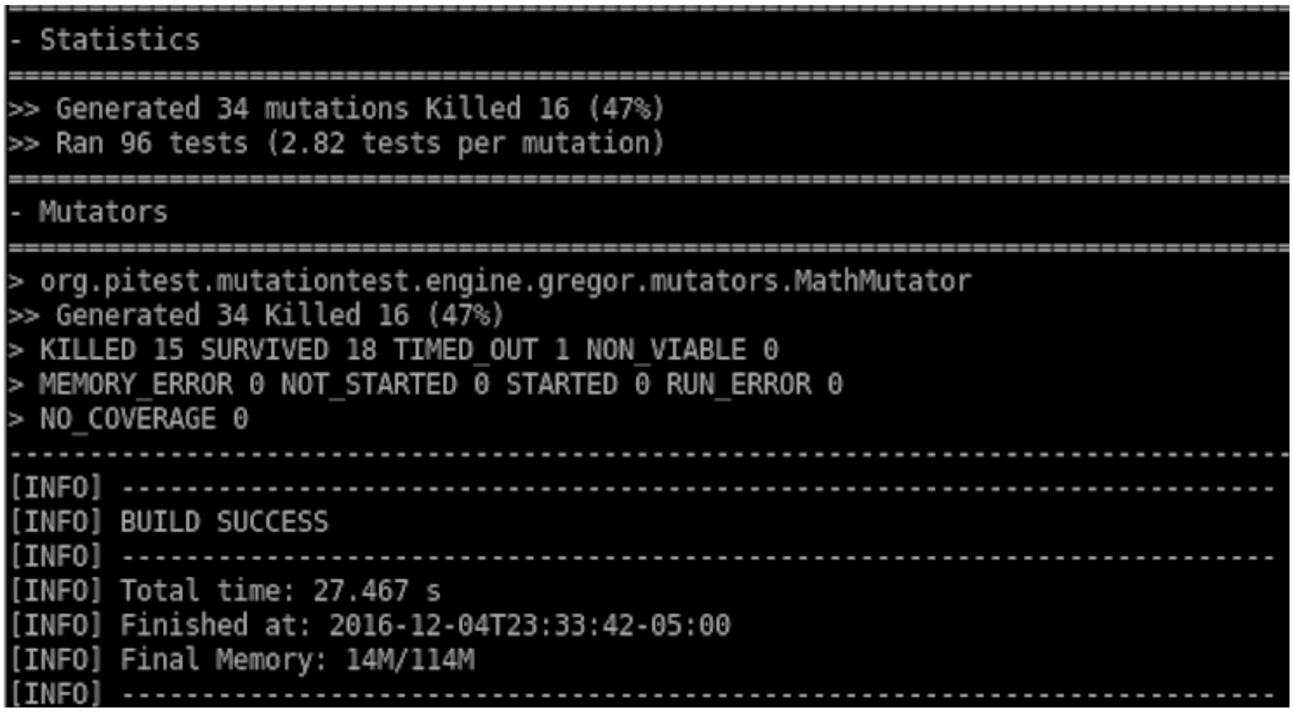
2.
if we run PIT with only “Conditionals Boundary Mutator”, “Increments Mutator”, and “Math Mutator” , we can get result below:

(for Conditionals Boundary Mutator)



```
================================================================================
- Statistics
================================================================================
>> Generated 24 mutations Killed 16 (67%)
>> Ran 21 tests (0.88 tests per mutation)
================================================================================
- Mutators
================================================================================
> 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
--------------------------------------------------------------------------------
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 23.541 s
[INFO] Finished at: 2016-12-04T23:32:04-05:00
[INFO] Final Memory: 9M/112M
[INFO] ------------------------------------------------------------------------
b3185-01:~/csc410_a2/Assignment2/as2-project$
```

(for increment mutator)

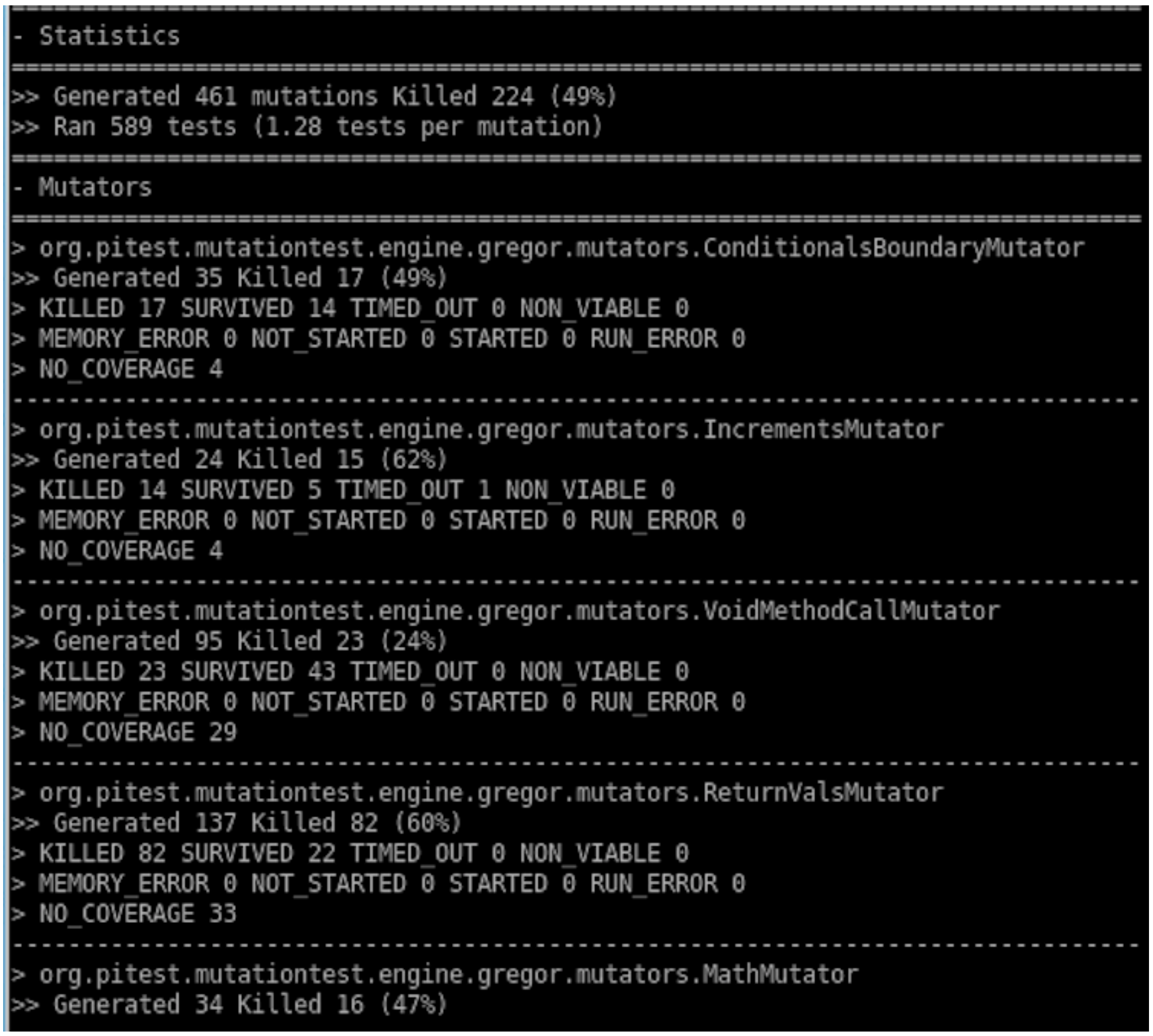


```
================================================================================
- Statistics
================================================================================
>> Generated 34 mutations Killed 16 (47%)
>> Ran 96 tests (2.82 tests per mutation)
================================================================================
- Mutators
================================================================================
> 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
--------------------------------------------------------------------------------
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 27.467 s
[INFO] Finished at: 2016-12-04T23:33:42-05:00
[INFO] Final Memory: 14M/114M
[INFO] ------------------------------------------------------------------------
```

(for math mutator)
I think the main reason why mutator 's coverage is higher than other two is that there are many places where assignment increments/decrements are used. And it has bigger influences on the project (if there are something wrong with them , the result will easily become different , which causes more mutations be killed)

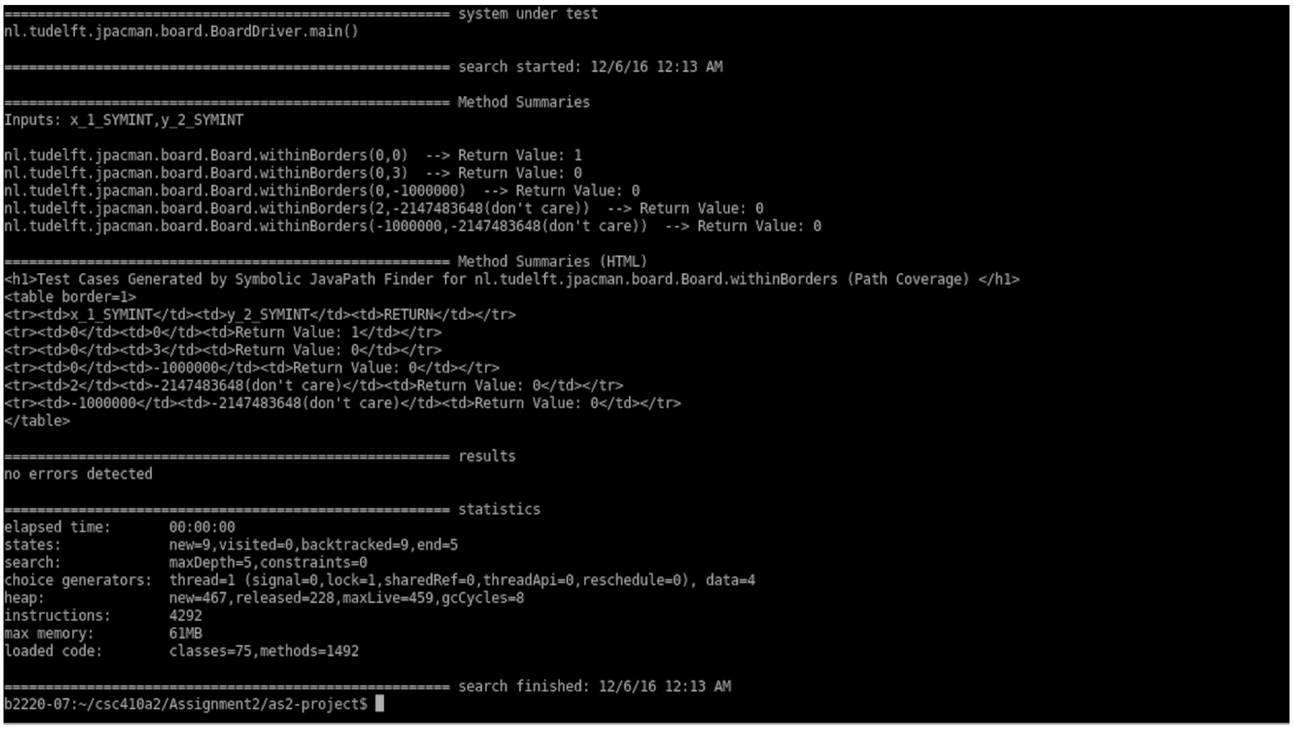
2.3
 I added a method testWithinBorders() to the BoardTest.java file, this method tests the method withinBorders(int x, int y) in the Board.java file.
 Since there are some conditional boundary operators in that file and testWithinBorders() test an edge example withinBorders(0,0), so this could actually change the return result entirely (ex:
 he conditional boundary mutator could change:
 return x >= 0 && x < getWidth() && y >= 0 && y < getHeight();   change to
 return x > 0 && x < getWidth() && y > 0 && y < getHeight();)
so the overall coverage has been improved from 47->49% (pay attention to change on conditional boundary mutation coverage : 43% 49%)

the result is :



```
================================================================================
- Statistics
================================================================================
>> Generated 461 mutations Killed 224 (49%)
>> Ran 589 tests (1.28 tests per mutation)
================================================================================
- Mutators
================================================================================
> org.pitest.mutationtest.engine.gregor.mutators.ConditionalsBoundaryMutator
>> Generated 35 Killed 17 (49%)
> KILLED 17 SURVIVED 14 TIMED_OUT 0 NON_VIABLE 0
> MEMORY_ERROR 0 NOT_STARTED 0 STARTED 0 RUN_ERROR 0
> NO_COVERAGE 4
--------------------------------------------------------------------------------
> org.pitest.mutationtest.engine.gregor.mutators.IncrementsMutator
>> Generated 24 Killed 15 (62%)
> KILLED 14 SURVIVED 5 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 23 (24%)
> KILLED 23 SURVIVED 43 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 82 (60%)
> KILLED 82 SURVIVED 22 TIMED_OUT 0 NON_VIABLE 0
> MEMORY_ERROR 0 NOT_STARTED 0 STARTED 0 RUN_ERROR 0
> NO_COVERAGE 33
--------------------------------------------------------------------------------
> org.pitest.mutationtest.engine.gregor.mutators.MathMutator
>> Generated 34 Killed 16 (47%)
```

part 3
1 (I only give the JPFBoardTest.java with complete test suite , so see my code under of 3.2)



```
====================================================== system under test
nl.tudelft.jpacman.board.BoardDriver.main()

====================================================== search started: 12/6/16 12:13 AM

====================================================== Method Summaries
Inputs: x_1_SYMINT,y_2_SYMINT

nl.tudelft.jpacman.board.Board.withinBorders(0,0)  --> Return Value: 1
nl.tudelft.jpacman.board.Board.withinBorders(0,3)  --> Return Value: 0
nl.tudelft.jpacman.board.Board.withinBorders(0,-1000000)  --> Return Value: 0
nl.tudelft.jpacman.board.Board.withinBorders(2,-2147483648(don't care))  --> Return Value: 0
nl.tudelft.jpacman.board.Board.withinBorders(-1000000,-2147483648(don't care))  --> Return Value: 0

====================================================== Method Summaries (HTML)
<h1>Test Cases Generated by Symbolic JavaPath Finder for nl.tudelft.jpacman.board.Board.withinBorders (Path Coverage) </h1>
<table border=1>
<tr><td>x_1_SYMINT</td><td>y_2_SYMINT</td><td>RETURN</td></tr>
<tr><td>0</td><td>0</td><td>Return Value: 1</td></tr>
<tr><td>0</td><td>3</td><td>Return Value: 0</td></tr>
<tr><td>0</td><td>-1000000</td><td>Return Value: 0</td></tr>
<tr><td>2</td><td>-2147483648(don't care)</td><td>Return Value: 0</td></tr>
<tr><td>-1000000</td><td>-2147483648(don't care)</td><td>Return Value: 0</td></tr>
</table>

====================================================== results
no errors detected

====================================================== statistics
elapsed time:       00:00:00
states:             new=9,visited=0,backtracked=9,end=5
search:             maxDepth=5,constraints=0
choice generators:  thread=1 (signal=0,lock=1,sharedRef=0,threadApi=0,reschedule=0), data=4
heap:               new=467,released=228,maxLive=459,gcCycles=8
instructions:       4292
max memory:         61MB
loaded code:        classes=75,methods=1492

====================================================== search finished: 12/6/16 12:13 AM
b2220-07:~/csc410a2/Assignment2/as2-project$ █
```

this is the test cases I got .
for JPFBoardTest.java file , please see below)

2. the generated test cases are not complete, it has test case on boundary (the first not ,the (0,0) one, also pay attention my MAX_WIDTH is 2, MAX_HEIGHT is 3) , it also has test case off boundary (for example , the third one , the(0,-1000000) one).
So we need to add a test case in boundary (I added (1,1) here)
here is content of my JPFBoardTest.java file:

```
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 = 4;
private static final int MAX_HEIGHT = 6;

/**
 * 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);
}

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


/**
 * test withinBorders() with return value of false
 */
@Test
public void testWithinBorders2(){
        assertEquals(false, board.withinBorders(0, 3));
}

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

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

/**
 * test withinBorders() with return value of false
 */
@Test
public void testWithinBorders5(){
        assertEquals(false, board.withinBorders(-1000000,-2147483648));
}

////////////////////////the generated test suite is not complete////////////
// since we already have inputs which are off boundary(0,3) and on boundary (0,0)
```

```
	//we need to add an input which is in boundary
	public void testWithinBorders6(){
		assertEquals(false, board.withinBorders(1,1));
	}
}
```

part 4
1. see my code