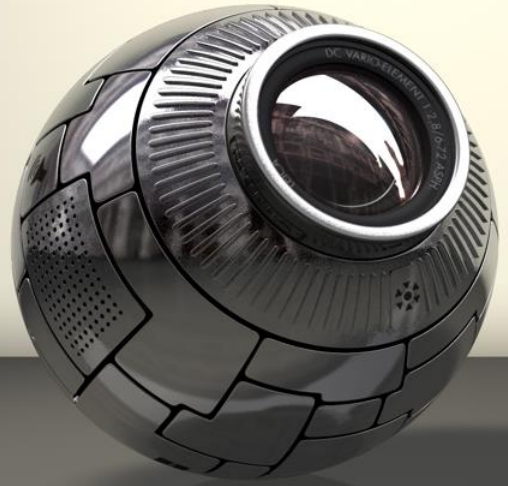
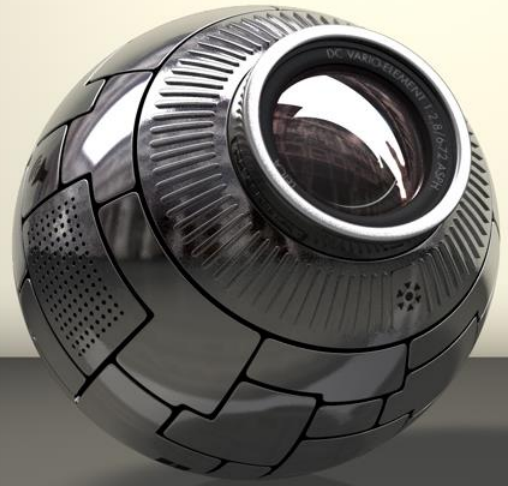
Software Verification
#5. 2nd Testing



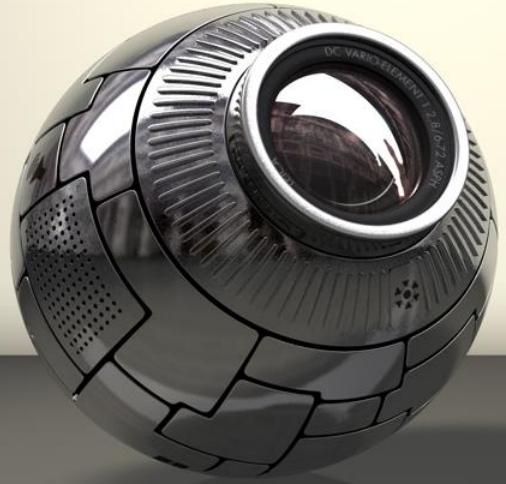
T3. SangYoon Kim
Green Kim

INDEX

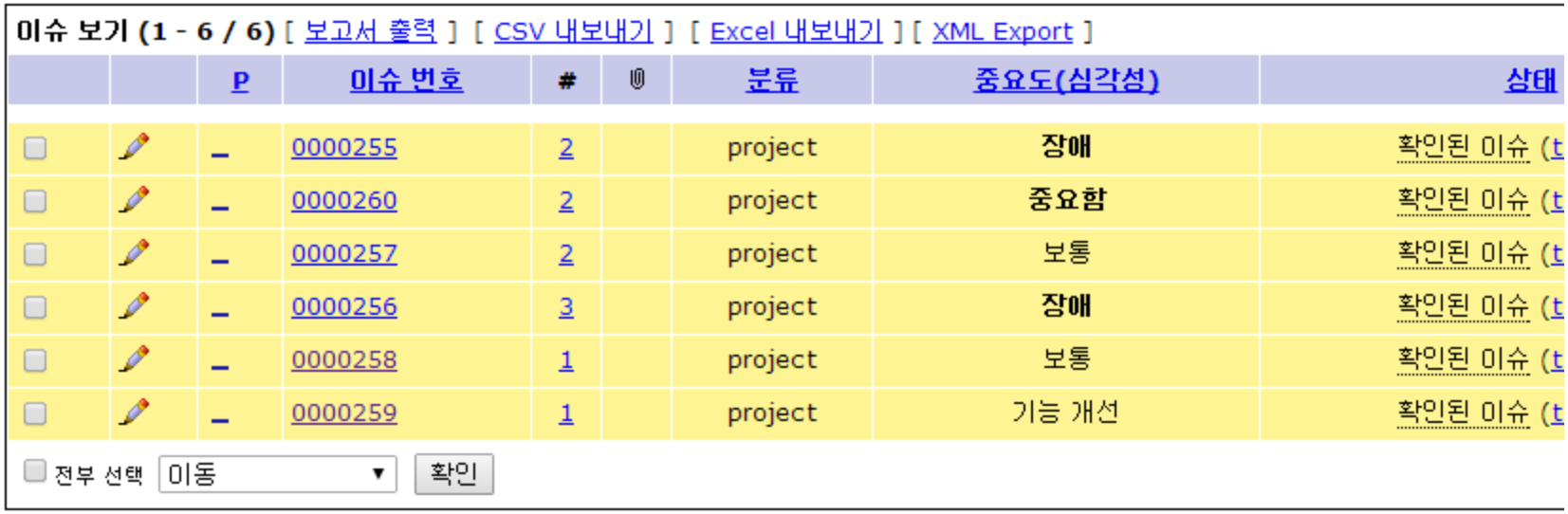
- Category Partition Testing
- Testing
  - Jfeature
- Static Analysis
  1) Clover
  2) Codepro



Category Partitioning



Before Start



이슈 보기 (1 - 6 / 6) [보고서 출력] [CSV 내보내기] [Excel 내보내기] [XML Export]

		P	이슈 번호	#		분류	중요도(심각성)	상태
☐	✎	_	0000255	2		project	**장애**	확인된 이슈 (t
☐	✎	_	0000260	2		project	**중요함**	확인된 이슈 (t
☐	✎	_	0000257	2		project	보통	확인된 이슈 (t
☐	✎	_	0000256	3		project	**장애**	확인된 이슈 (t
☐	✎	_	0000258	1		project	보통	확인된 이슈 (t
☐	✎	_	0000259	1		project	기능 개선	확인된 이슈 (t

☐ 전부 선택 이동 ▾ 확인

Category Partitioning

Step 1. Choosing Categories

분류	항목
엘리베이터 이동 요청	상승
	하강
	하차
요청 취소	요청취소
수치 조정	maxLoad
응급	Fire
	Black out
	Fix
복구	Normal

Category Partitioning

Step 2. Identify Representative Value

엘리베이터 이동 요청

1) 상승
    1. request
    2. floor ≦ 0
    3. weight ≦ 0
    4. weight > maxLoad
2) 하강
    1. request
    2. floor ≦ 0
    3. weight ≦ 0
    4. weight > maxLoad

Category Partitioning

Step 2. Identify Representative Value

엘리베이터 이동 요청

3) 하차

1.1 좌측 request
2.1 좌측 floor ≦ 0
3.1 좌측 weight ≦ 0
4.1 좌측 weight > maxLoad

1.2 우측 request
2.2 우측 floor ≦ 0
3.2 weight ≦ 0
4.2 weight > maxLoad

Category Partitioning

Step 2. Identify Representative Value

요청 취소

1) 요청취소

1.1 좌측 request

2.1 좌측 floor ≦ 0 || floor = not exist

1.2 우측 request

2.2 우측 floor ≦ 0 || floor = not exist

Category Partitioning

Step 2. Identify Representative Value

수치 조정

1) maxLoad

1.1 좌측 maxLoad ≧ nowWeight

2.1 좌측 maxLoad < nowWeight

1.2 우측 maxLoad ≧ nowWeight

2.2 우측 maxLoad < nowWeight

Category Partitioning

Step 2. Identify Representative Value

응급

1) Fire
   1.화재 발생

2) Black out
   1. 정전 발생

3) Fix
   1. 점검 상태로 변경

Category Partitioning

Step 2. Identify Representative Value

복구

1) Normal

1. 복구를 성공한다.

Category Partitioning

Step 3. Generate Test Case Specifications

1. 각 카테고리 값을 곱하여 테스트 케이스 생성
   → 6144개의 테스트 케이스 생성

2. Constratints 적용
   1. Error constraints
   2. Property constraints
   3. Single constraints

Category Partitioning

Summary of categories

엘리베이터 이동 요청

1) 상승

1. request
2. floor ≦ 0 [error]
3. weight ≦ 0 [error]
4. weight > maxLoad [error]

2) 하강

1. request
2. floor ≦ 0 [error]
3. weight ≦ 0 [error]
4. weight > maxLoad [error]

3) 하차

1.1 좌측 request
2.1 좌측 floor ≦ 0 [error]
3.1 좌측 weight ≦ 0 [error]
4.1 좌측 weight > maxLoad [error]

1.2 우측 request
2.2 우측 floor ≦ 0 [error]
3.2 우측 weight ≦ 0 [error]
4.2 우측weight > maxLoad [error]

요청 취소

1) 요청취소

1.1 좌측 request
2.1 좌측 floor ≦ 0 || floor = not exist [error]

1.2 우측 request
2.2 우측 floor ≦ 0 || floor = not exist [error]

수치 조정

1) maxLoad

1.1 좌측 maxLoad ≧ nowWeight [error]
2.1 좌측 maxLoad < nowWeight [error]

1.2 우측 maxLoad ≧ nowWeight [error]
2.2 우측 maxLoad < nowWeight [error]

응급

1) Fire

1.화재 발생

2) Black out

1. 정전 발생

3) Fix

1. 점검 상태로 변경

복구

1) Normal

1. 복구를 성공한다.

Category Partitioning

Brute force tests Plan – 독립성 확인

Category partitioning test에서 검증할 수 없는 내용들

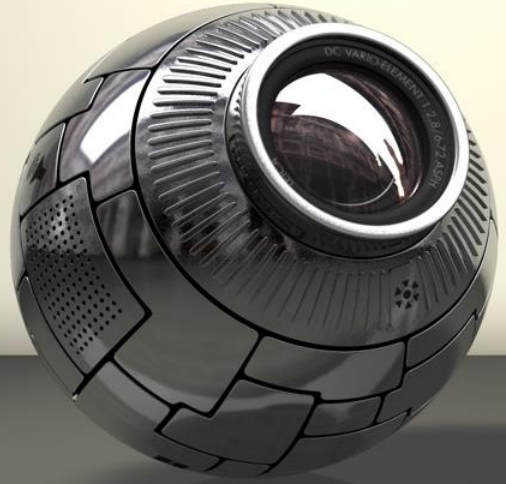
Num	제어	첫번째	두번째	결과	Num	제어	첫번째	두번째	결과
1	문 제어	좌측 Open	우측 Open		17	큐할당	좌측 Up	우측 Up	
2	문 제어	좌측 Open	우측 Close		18	큐할당	좌측 Up	우측 Down	
3	문 제어	좌측 Open	우측 Open		19	큐할당	좌측 Up	우측 Up	
4	문 제어	좌측 Open	우측 Close		20	큐할당	좌측 Up	우측 Down	
5	문 제어	좌측 Close	우측 Open		21	큐할당	좌측 Down	우측 Up	
6	문 제어	좌측 Close	우측 Close		22	큐할당	좌측 Down	우측 Down	
7	문 제어	좌측 Close	우측 Open		23	큐할당	좌측 Down	우측 Up	
8	문 제어	좌측 Close	우측 Close		24	큐할당	좌측 Down	우측 Down	
9	문 제어	우측 Open	좌측 Open		25	큐할당	우측 Up	좌측 Up	
10	문 제어	우측 Open	우측 Close		26	큐할당	우측 Up	좌측 Down	
11	문 제어	우측 Open	좌측 Open		27	큐할당	우측 Up	좌측 Up	
12	문 제어	우측 Open	우측 Close		28	큐할당	우측 Up	좌측 Down	
13	문 제어	우측 Close	좌측 Open		29	큐할당	우측 Down	좌측 Up	
14	문 제어	우측 Close	우측 Close		30	큐할당	우측 Down	좌측 Down	
15	문 제어	우측 Close	좌측 Open		31	큐할당	우측 Down	좌측 Up	
16	문 제어	우측 Close	우측 Close		32	큐할당	우측 Down	좌측 Down	

Category Partitioning

Brute force tests Plan – Scenario Testing

번호	내용	결과
1	같은 요청을 3개 연속 입력	
2	비상상황(화재, 정전, 점검) 중에 요청 입력	
3	4층 8층 5층 순서로 입력	
4	요청- 화재 – 복구 – 요청 – 화재 – 복구	

Testing



Pairwise Testing

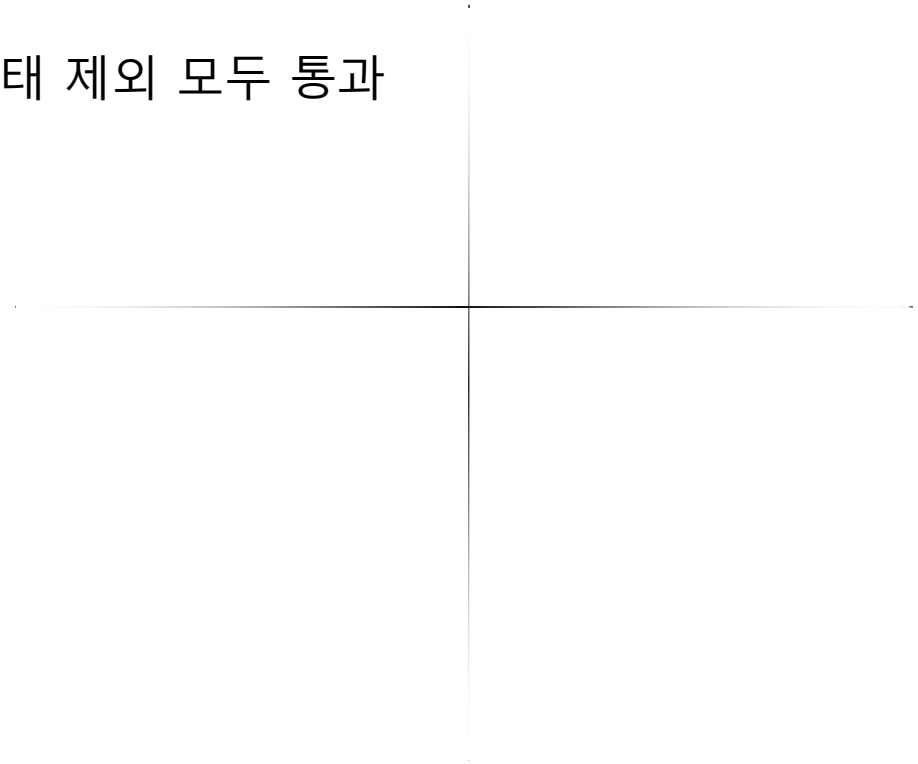
Hexawise

HEXAWISE

MORE COVERAGE. FEWER TESTS.

Bruteforce Testing
기본 입력 점검

화재 정전 점검 상태 제외 모두 통과



Pairwise Testing

Testing result

request button (5)	pass	floor less t...	weight less ...	weight over	N/A
right landing button (5)	pass	floor less t...	weight less ...	weight more ...	N/A
right cancel request (4)	cancel pass	not exist floor	id != 1,2,3	N/A	
left landing button (5)	pass	floor less t...	weight less ...	weight more ...	N/A
left cancel request (4)	cancel pass	not exist floor	id != 1,2,3	N/A	
set Maxload (3)	more than no...	less than no...	N/A		
emergency (4)	fire	black out	fix	N/A	
restore (3)	restored	restore fail	N/A		

Pairwise Testing

Testing result

Test Number	request button	right landing button	right cancel request	left landing button	left cancel request	set Maxload	emergency	restore
1	pass	pass	cancel pass - 1, 2, 3	pass	cancel pass - 1, 2, 3	more than now weight	fire	restored
2	floor less than zero	pass	not exist floor	floor less than zero	not exist floor	N/A	black out	restore fail
3	weight over	pass	N/A	weight more than now weight	N/A	less than now weight	N/A	N/A
4	weight less than zero	pass	id != 1,2,3	weight less than zero	id != 1,2,3	more than now weight	fix	restore fail
5	pass	weight less than zero	cancel pass - 1, 2, 3	weight less than zero	not exist floor	less than now weight	N/A	N/A
6	pass	floor less than zero	N/A	floor less than zero	id != 1,2,3	N/A	fire	restored
7	N/A	pass	not exist floor	N/A	cancel pass - 1, 2, 3	N/A	fix	restored
8	pass	weight more than now weight	id != 1,2,3	weight more than now weight	cancel pass - 1, 2, 3	more than now weight	black out	restored
9	pass	N/A	cancel pass - 1, 2, 3	N/A	N/A	more than now weight	black out	restore fail
10	floor less than zero	floor less than zero	id != 1,2,3	pass	N/A	N/A	fix	restored
11	floor less than zero	weight more than now weight	N/A	weight less than zero	cancel pass - 1, 2, 3	more than now weight	N/A	N/A
12	weight less than zero	N/A	id != 1,2,3	floor less than zero	cancel pass - 1, 2, 3	less than now weight	N/A	N/A
13	weight over	weight more than now weight	not exist floor	pass	id != 1,2,3	less than now weight	N/A	N/A
14	floor less than zero	weight less than zero	cancel pass - 1, 2, 3	weight more than now weight	id != 1,2,3	N/A	fire	restore fail
15	weight less than zero	floor less than zero	not exist floor	weight more than now weight	not exist floor	more than now weight	fire	restored
16	weight less than zero	weight more than now weight	cancel pass - 1, 2, 3	floor less than zero	N/A	N/A	fix	restore fail
17	weight over	floor less than zero	cancel pass - 1, 2, 3	weight less than zero	cancel pass - 1, 2, 3	N/A	black out	restored
18	floor less than zero	N/A	N/A	pass	not exist floor	more than now weight	fix	restored
19	weight less than zero	weight less than zero	not exist floor	pass	cancel pass - 1, 2, 3	more than now weight	black out	restore fail
20	weight over	weight less than zero	id != 1,2,3	floor less than zero	not exist floor	more than now weight	fire	restored
21	N/A	weight less than zero	N/A	floor less than zero	N/A	more than now weight	fire	restore fail
22	N/A	floor less than zero	cancel pass - 1, 2, 3	pass	not exist floor	less than now weight	N/A	N/A
23	weight over	N/A	not exist floor	weight less than zero	id != 1,2,3	N/A	fire	restore fail
24	N/A	N/A	N/A	weight more than now weight	id != 1,2,3	*more than now weight*	black out	*restored*
25	floor less than zero	floor less than zero	id != 1,2,3	N/A	not exist floor	*N/A*	fire	restore fail
26	weight less than zero	weight less than zero	N/A	N/A	id != 1,2,3	*N/A*	fix	*restore fail*
27	N/A	weight more than now weight	id != 1,2,3	weight less than zero	not exist floor	*more than now weight*	fire	*restored*
28	weight over	weight more than now weight	not exist floor	N/A	N/A	*more than now weight*	fix	*restored*
29	floor less than zero	*pass*	*N/A*	N/A	*cancel pass - 1, 2, 3*	less than now weight	N/A	N/A
30	pass	*pass*	not exist floor	weight less than zero	N/A	*more than now weight*	fix	*restore fail*
31	*pass*	*pass*	*not exist floor*	*pass*	*cancel pass - 1, 2, 3*	N/A	N/A	N/A
32	*pass*	*pass*	*cancel pass - 1, 2, 3*	weight more than now weight	*N/A*	*more than now weight*	fix	*restored*
33	*pass*	*floor less than zero*	*cancel pass - 1, 2, 3*	*weight less than zero*	*cancel pass - 1, 2, 3*	less than now weight	*no possible value*	restored
34	*weight less than zero*	*pass*	*cancel pass - 1, 2, 3*	*weight less than zero*	*cancel pass - 1, 2, 3*	less than now weight	*no possible value*	restore fail

Pairwise Testing

Testing result

번호	결과	번호	결과	번호	결과	번호	결과
1	pass	9	pass	17	pass	25	pass
2	pass	10	pass	18	pass	26	pass
3	pass	11	pass	19	pass	27	pass
4	pass	12	pass	20	pass	28	pass
5	pass	13	pass	21	pass	29	pass
6	pass	14	pass	22	pass	30	pass
7	pass	15	pass	23	pass	31	pass
8	pass	16	pass	24	pass	32	pass

Bruteforce Testing

Brute force testing result – 독립성 확인

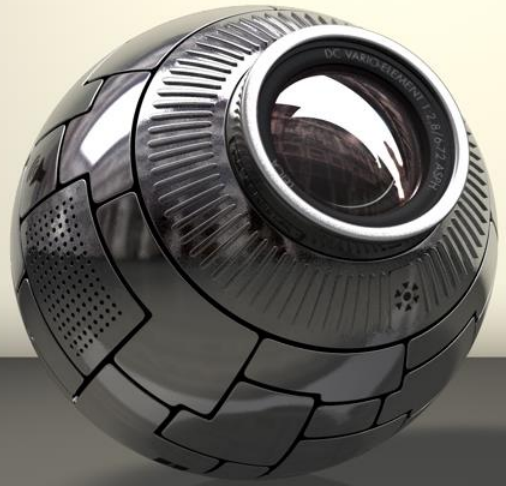
Num	제어	첫번째	두번째	결과	Num	제어	첫번째	두번째	결과
1	문 제어	좌측 Open	우측 Open	Pass	17	큐할당	좌측 Up	우측 Up	Pass
2	문 제어	좌측 Open	우측 Close	Pass	18	큐할당	좌측 Up	우측 Down	Pass
3	문 제어	좌측 Open	우측 Open	Pass	19	큐할당	좌측 Up	우측 Up	Pass
4	문 제어	좌측 Open	우측 Close	Pass	20	큐할당	좌측 Up	우측 Down	Pass
5	문 제어	좌측 Close	우측 Open	Pass	21	큐할당	좌측 Down	우측 Up	Pass
6	문 제어	좌측 Close	우측 Close	Pass	22	큐할당	좌측 Down	우측 Down	Pass
7	문 제어	좌측 Close	우측 Open	Pass	23	큐할당	좌측 Down	우측 Up	Pass
8	문 제어	좌측 Close	우측 Close	Pass	24	큐할당	좌측 Down	우측 Down	Pass
9	문 제어	우측 Open	좌측 Open	Pass	25	큐할당	우측 Up	좌측 Up	Pass
10	문 제어	우측 Open	우측 Close	Pass	26	큐할당	우측 Up	좌측 Down	Pass
11	문 제어	우측 Open	좌측 Open	Pass	27	큐할당	우측 Up	좌측 Up	Pass
12	문 제어	우측 Open	우측 Close	Pass	28	큐할당	우측 Up	좌측 Down	Pass
13	문 제어	우측 Close	좌측 Open	Pass	29	큐할당	우측 Down	좌측 Up	Pass
14	문 제어	우측 Close	우측 Close	Pass	30	큐할당	우측 Down	좌측 Down	Pass
15	문 제어	우측 Close	좌측 Open	Pass	31	큐할당	우측 Down	좌측 Up	Pass
16	문 제어	우측 Close	우측 Close	Pass	32	큐할당	우측 Down	좌측 Down	Pass

Bruteforce Testing

Brute force testing result – Scenario testing

번호	내용	결과
1	같은 요청을 3개 연속 입력	Pass
2	비상상황(화재, 정전, 점검) 중에 요청 입력	Pass
3	4층 8층 5층 순서로 입력	Pass
4	요청- 화재 – 복구 – 요청 – 화재 – 복구	Pass

Jfeature



Unit Test 분석

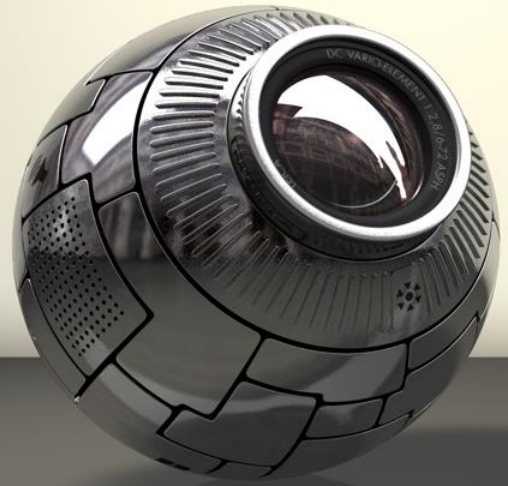
```
public void testEnQueue(){
QueueAlgorithm testA = new QueueAlgorithm();
testA.enQueue(1, 3, 1, 5); //add to queue 1: request 3 level 1 load 5
testA.enQueue(2, 3, 6, 7); //add to queue 1 : request 3 lev 6 load 7
testA.enQueue(3, 1, 4, 6); //add to queue 3 : request 1 lev 4 load 6
testA.enQueue(3, 2, 5, 60);
assertEquals(1, testA.selectQueue1.queuestore.size());
assertEquals(2, testA.aboardQueue.queuestore.size());
assertEquals(1, testA.selectQueue1.queuestore.size());

}
```

Unit Test 분석

- 테스트케이스가 UseCase 의 일부분만을 테스트.
- Jfeature를 사용하기 위해서는 각 요구사항의 주체가 되는 함수를 테스트 해야 함.
- 자세한 내용은 Clover 에서 설명

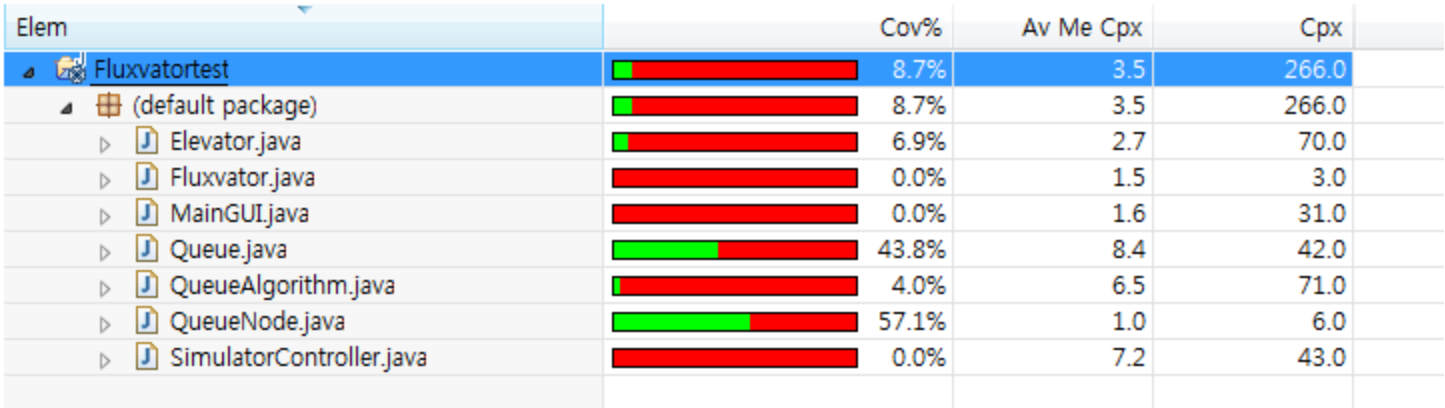
Static Analysis



1) Clover
2) Codepro

Static Analysis : Clover

Testing result



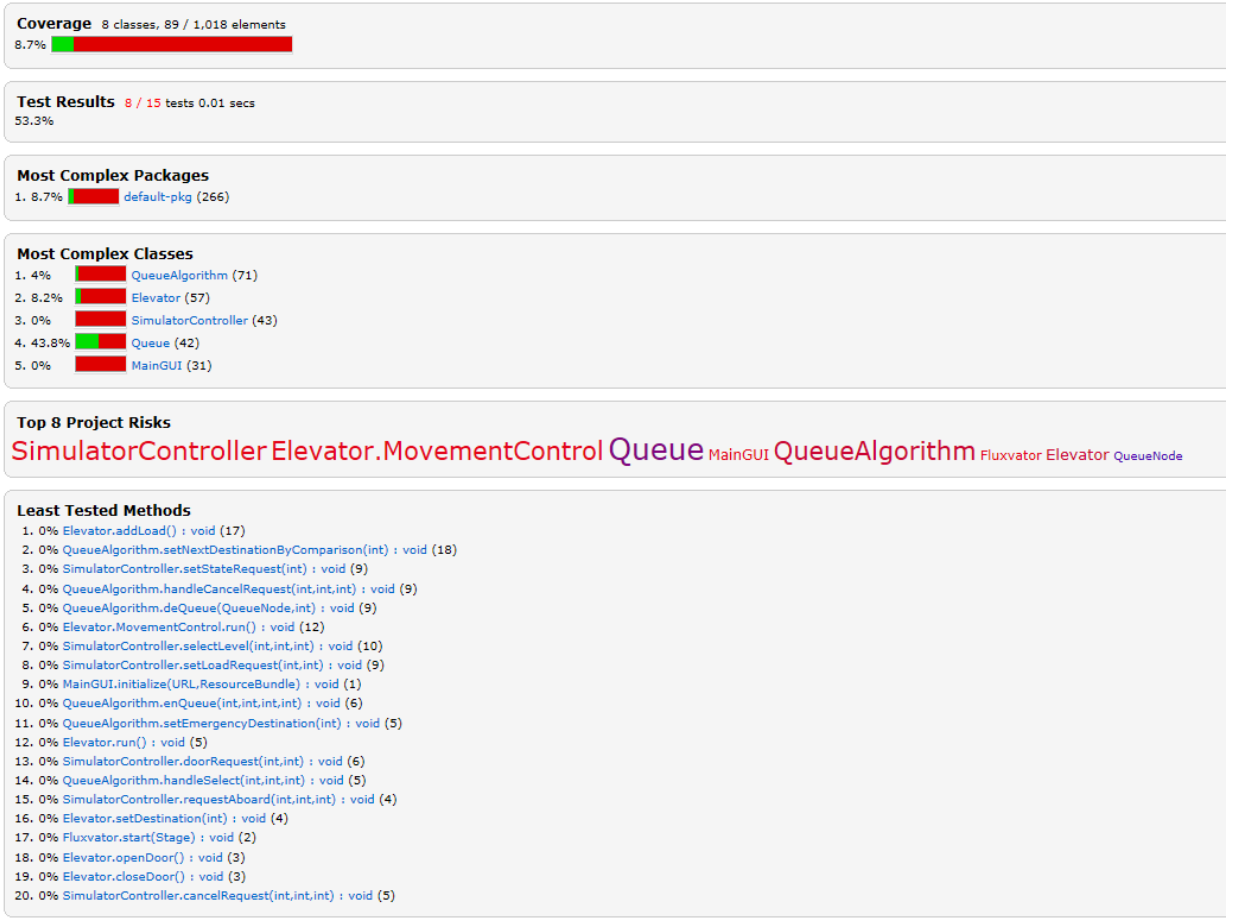
Elem	Cov%	Av Me Cpx	Cpx
Fluxvatortest	8.7%	3.5	266.0
(default package)	8.7%	3.5	266.0
Elevator.java	6.9%	2.7	70.0
Fluxvator.java	0.0%	1.5	3.0
MainGUI.java	0.0%	1.6	31.0
Queue.java	43.8%	8.4	42.0
QueueAlgorithm.java	4.0%	6.5	71.0
QueueNode.java	57.1%	1.0	6.0
SimulatorController.java	0.0%	7.2	43.0

Testing result
Coverage Contribution

Coverage Contribution:

Class	Contrib%	Uniq%
Elevator	23.2%	23.2%
Elevator	100.0%	100.0%
arrivalCalibratio	50.0%	50.0%
changeDirectior	55.6%	55.6%
closeDoor	30.0%	30.0%
getCurrentState	100.0%	100.0%
getDoorState	100.0%	100.0%
getMaxLoad	100.0%	100.0%
handleDoorReq	42.9%	42.9%
movingOrNot	100.0%	100.0%
openDoor	30.0%	30.0%
setCurrentState	100.0%	100.0%
setMaxLoad	100.0%	100.0%
stopOrMove	100.0%	100.0%
stopping	100.0%	100.0%
Queue	43.8%	43.8%
clearNode	100.0%	100.0%
deleteNode	100.0%	100.0%
findClosestNode	31.5%	31.5%
makeNode	100.0%	100.0%
searchForNode	100.0%	100.0%
QueueAlgorithm	8.0%	8.0%
QueueAlgorithm	100.0%	100.0%
compareClosest	25.0%	25.0%
enQueue	13.3%	13.3%
QueueNode	57.1%	57.1%
QueueNode	100.0%	100.0%
getDestinationL	100.0%	100.0%
getRequestID	100.0%	100.0%

Testing result
Total Result

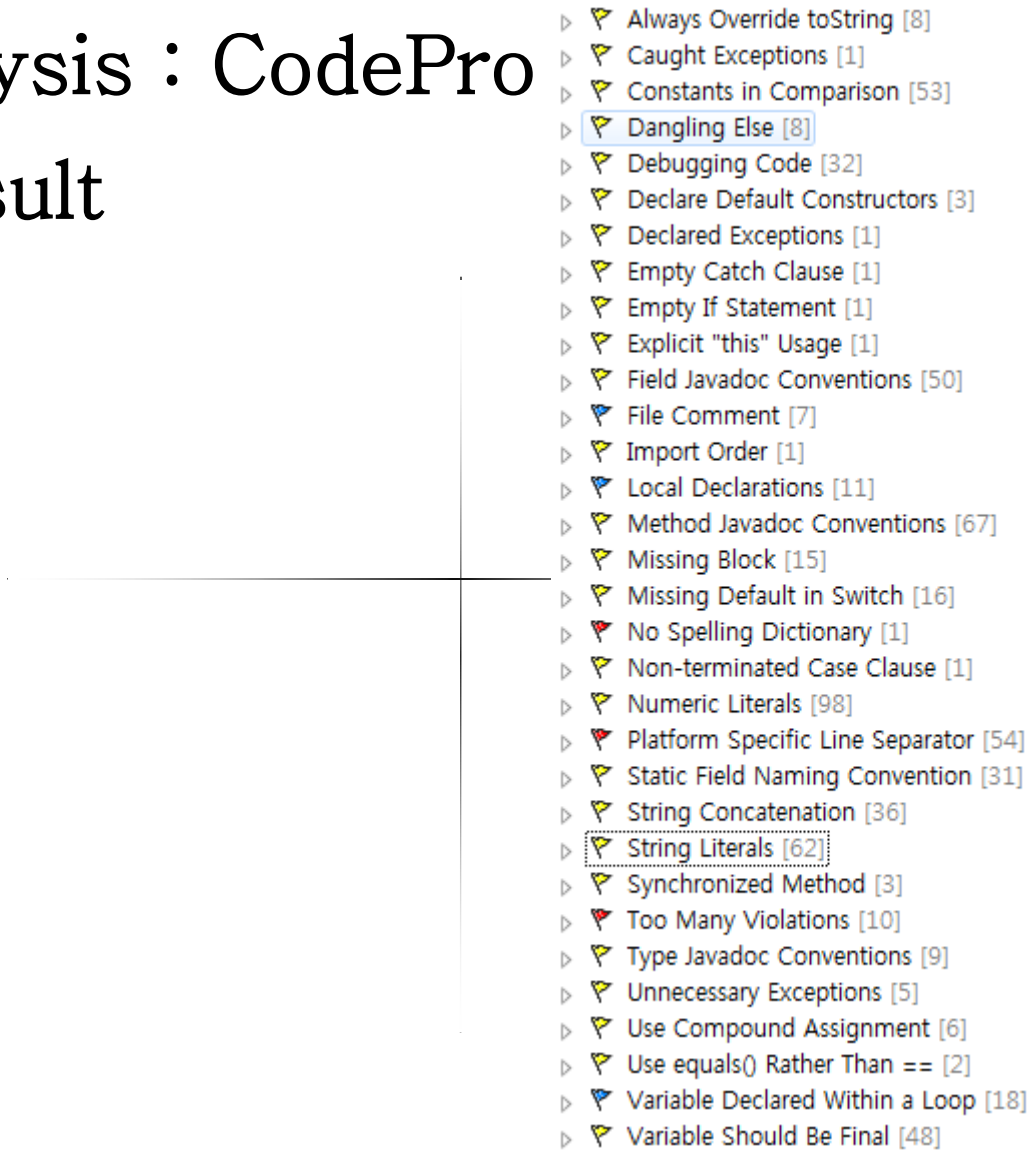


Static Analysis : Clover

Testing result에 대한 해석

- 전체적인 테스트 비중이 낮음
  - 커버리지가 50% 이상 되어야 함
  - 특정 클래스에만 테스트가 편중됨.

Static Analysis : CodePro Testing result



Static Analysis : Codepro

Testing result에 대한 해석

- 코드 작성 규칙에 대한 내용

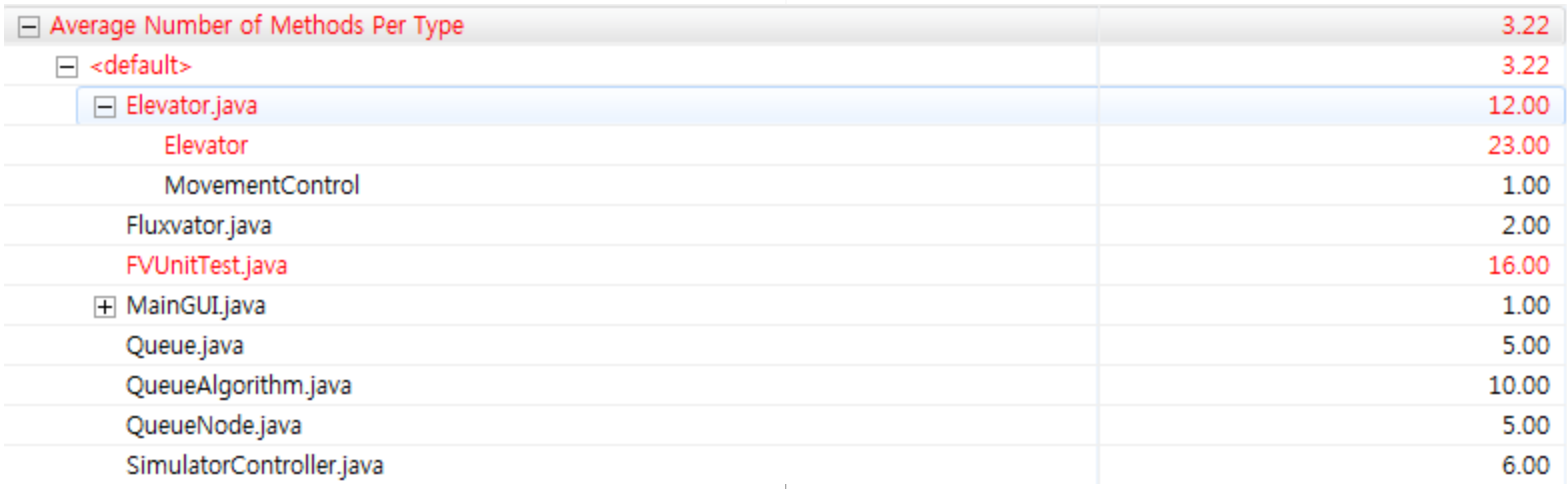
Static Analysis : CodePro

Testing result

Abstractness	0%
Average Block Depth	1.01
Average Cyclomatic Complexity	3.09
Average Lines Of Code Per Method	16.39
Average Number of Constructors Per Type	0.14
Average Number of Fields Per Type	3.25
Average Number of Methods Per Type	3.22
Average Number of Parameters	0.81
Comments Ratio	3.3%
Efferent Couplings	8
Lines of Code	1,449
Number of Characters	47,690
Number of Comments	48
Number of Constructors	4
Number of Fields	88
Number of Lines	1,556
Number of Methods	87
Number of Packages	1
Number of Semicolons	696
Number of Types	27
Weighted Methods	282

Static Analysis : CodePro

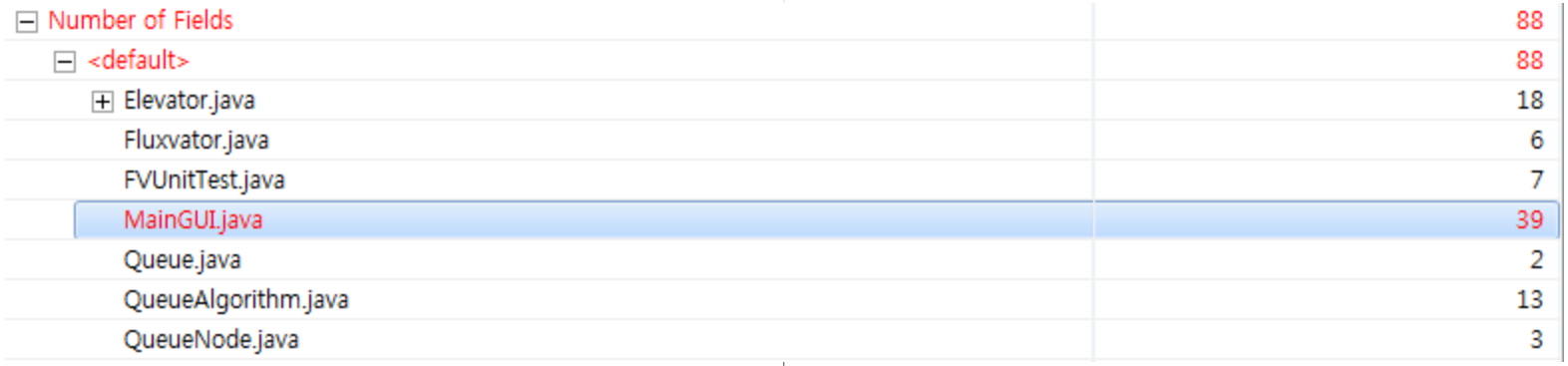
Testing result



Average Number of Methods Per Type	3.22
<default>	3.22
Elevator.java	12.00
Elevator	23.00
MovementControl	1.00
Fluxvator.java	2.00
FVUnitTest.java	16.00
MainGUI.java	1.00
Queue.java	5.00
QueueAlgorithm.java	10.00
QueueNode.java	5.00
SimulatorController.java	6.00

Static Analysis : CodePro

Testing result

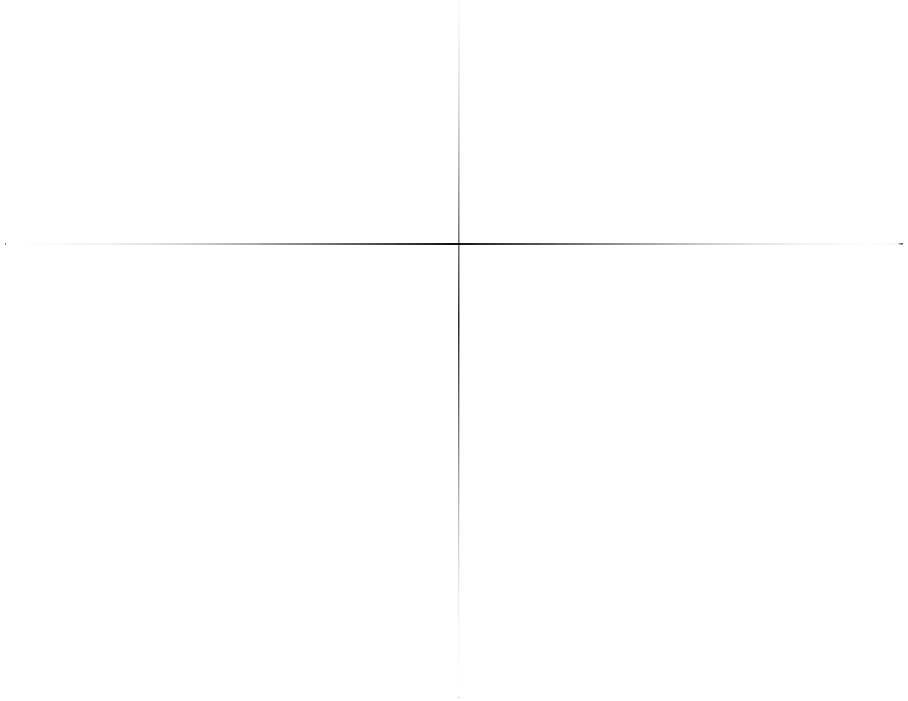


Number of Fields	88
<default>	88
Elevator.java	18
Fluxvator.java	6
FVUnitTest.java	7
MainGUI.java	39
Queue.java	2
QueueAlgorithm.java	13
QueueNode.java	3

Static Analysis : Codepro

Testing result에 대한 해석

- 전반적인 코드의 분산도를 분석



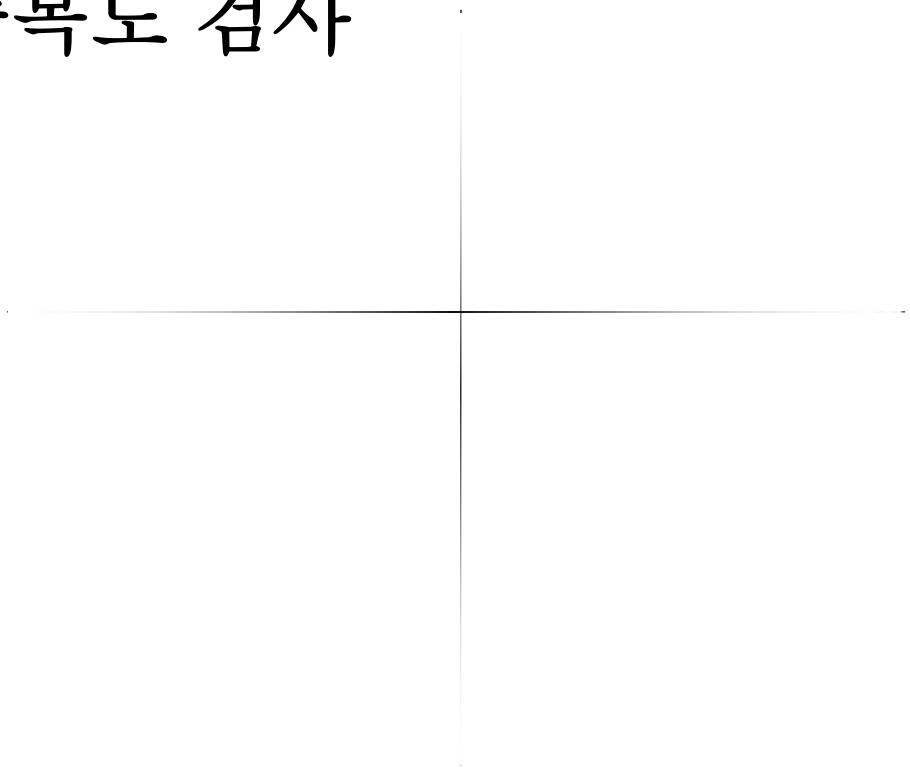
Static Analysis : CodePro

Testing result

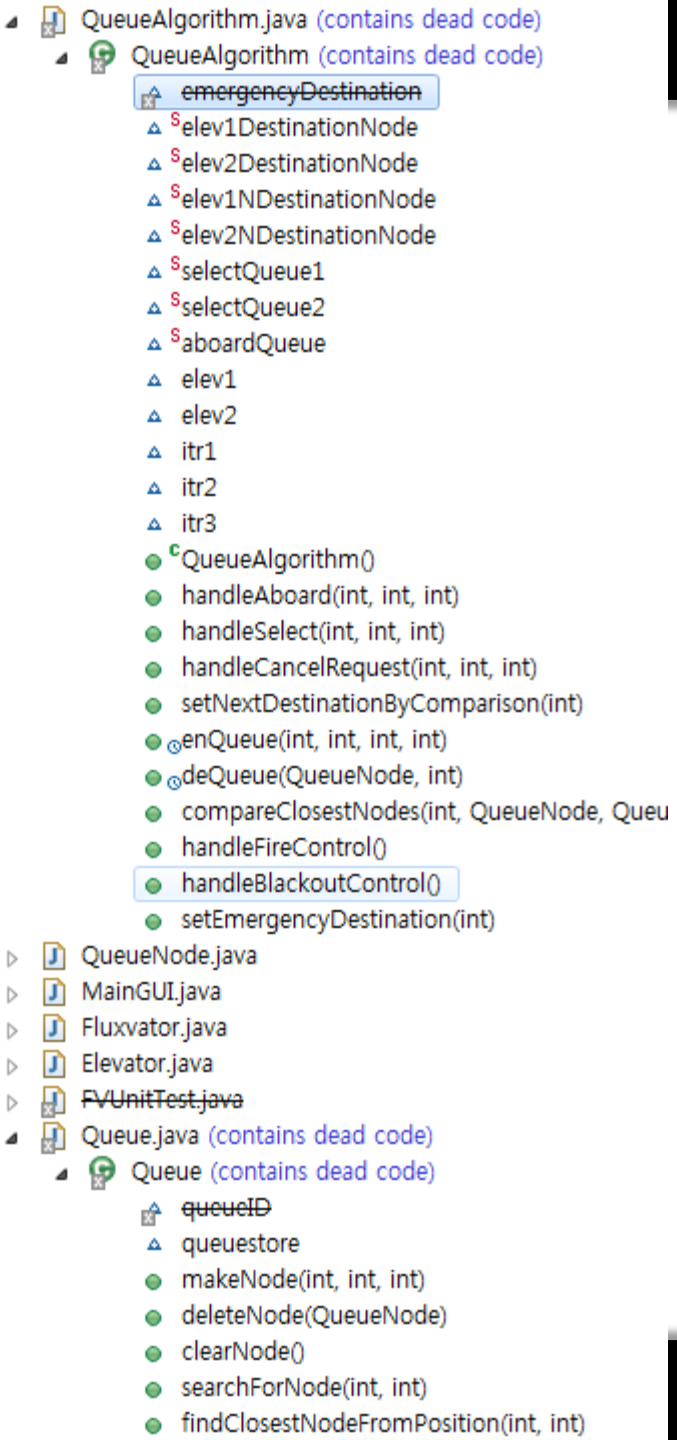
Static Analysis : Codepro

Testing result에 대한 해석

- 코드의 중복도 검사



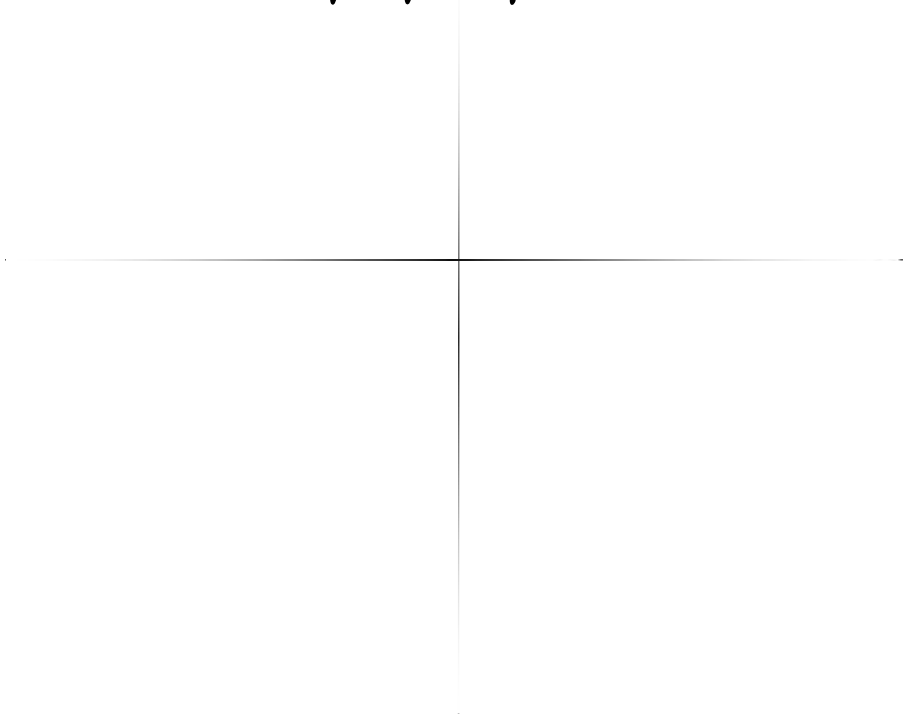
Static Analysis : CodePro Testing result



Static Analysis : Codepro

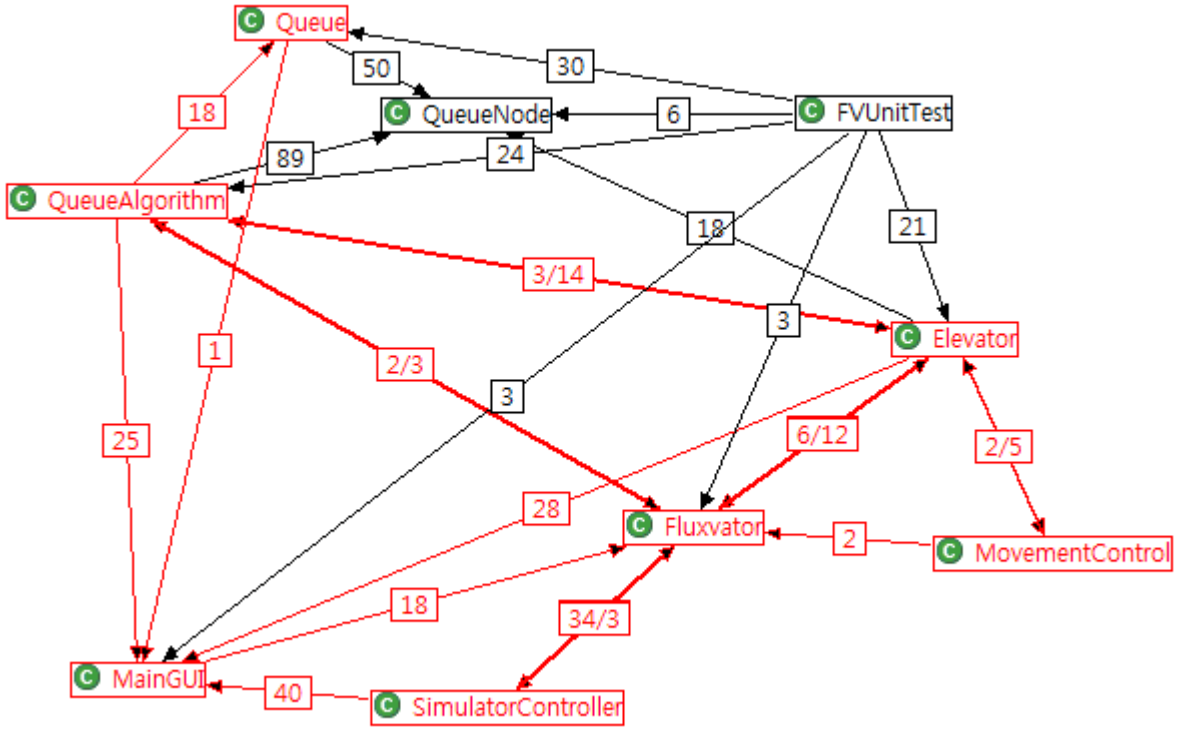
Testing result에 대한 해석

- 쓰이지 않는 코드의 검사



Static Analysis : CodePro

Testing result



Static Analysis : Codepro

Testing result에 대한 해석

- 순환참조 검사
- 단 방향인데 빨간색인것은 A->B->C->A 의 구조로 참조가 이뤄지고 있다는 표시.

추가사항

심각한 Memory Leak 발생.

프로세스를 종료시켰음에도 간혹 javaw.exe 가 살아있는 경우 확인 – 스레드가 종료되지 않음.

References

https://**hexawise**.com/

https://developers.google.com/java-dev-tools/codepro/doc/

Q&A

Thank You