Unit Testing Report for Fluxvator: an Elevator Control Simulator ver. 2

May 28th 2014

Team 3

Team Organization

200913215 기세파

201013275 강태호

201013760 이인구

Context

2

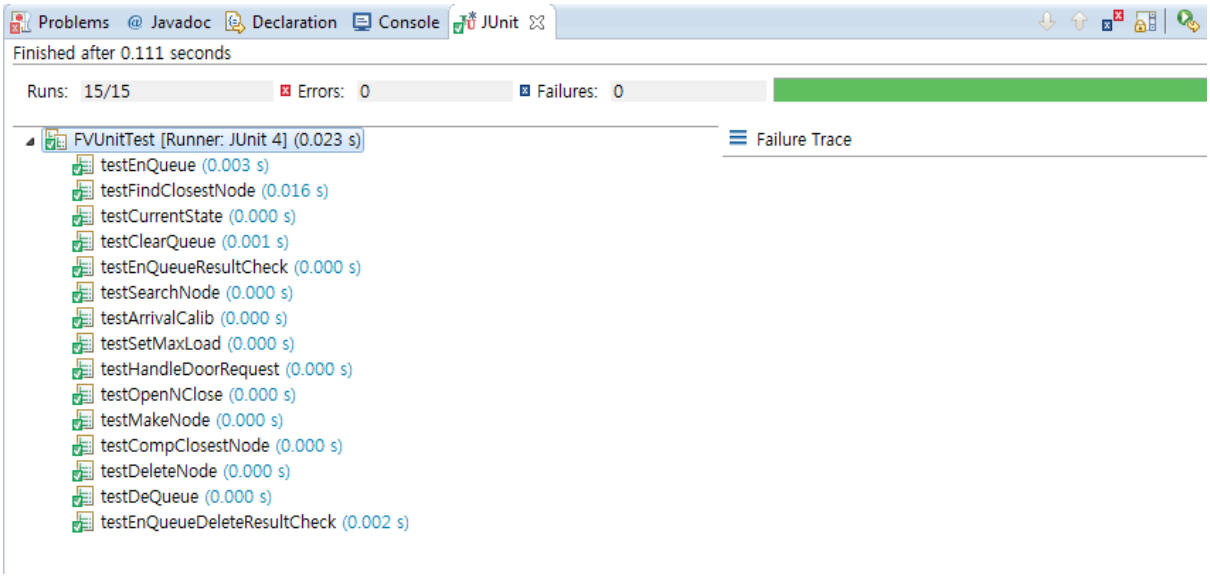
1. Disclaimer & Changes From 2040

본래 implementation시 2040단계에서 도출된 class diagram을 기반으로 구현을 하는 것이 정설이며, 구현당시 이대로 진행하기 위하여 노력하였다. 이는 클래스 다이어그램이 유스케이스와 Sequence Diagram을 기반으로 제작되었기 때문인데, 실제 구현 당시에 분명 유스케이스와 Sequence Diagram의 flow를 그대로 따랐음에도 불구하고 기술적인 이유(e.g. Threading, GUI와의 연동)혹은 구현하고자 한 일부 구조(ex : list나 iterator)가 java 기본 제공 라이브러리에서 이미 제공되어 구현의 편의를 위해 이들을 대신 사용함 등으로 인한 변경점이 있었다. 이처럼 유스케이스나 시퀀스 다이어그램의 **큰 그림에 변화를 주지 않는 수준의 클래스 내 구조 변화**를 적용하고 이전 레포트 클래스 다이어그램에서 메소드들의 return type을 적어놓지 않은 것을 수정하여 클래스 다이어그램을 다시 작성하였다. GUI 관련 객체는 애초에 클래스 다이어그램에 넣지 않기로 하여 변경된 클래스 다이어그램에서도 GUI에 구현에 관련된 클래스나 메소드들은 포함되지 않았다.

주의사항: 프로젝트 내 코드에서 GUI 클래스 참조가 자주일어나(이유 : logging) Unit test 시 해당 참조가 일어나는 코드들은 모두 Unit test가 자동적으로 실패가 일어나는 상황이 발생하였다. 때문에 Unit test 동안은 해당 코드들에서 MainGUI 참조하는 코드들에 한해 주석처리를 하고 진행하게 되었다.

2. Unit Testing Environment

2차 Unit Testing 시행은 May 27~28 2014에 진행하였다.
JUnit 4를 사용하였고, 클래스 내의 가장 기본적인 메소드 (getter, setter) 등에 대한 Unit Test는 제외하였다.



3. Unit Testing Specification

Name of Test, Procedure, Projected Result, Actual Result(Pass/Non-pass)

Test Case Name	**1. testEnQueue**
Objective(Procedure)	-enQueue(목적큐, 요청ID, 목적층, 하중변화량) 기능 확인 -큐를담은 큐알고리즘 생성 -Enqueue(1,3,1,5) (2,3,6,7) (3,1,4,6) (3,2,5,60) 시행 -큐 1,2,3의 크기 각각 확인
Estimated Result	selectQueue1의 크기 1, aboardQueue의 크기 2 예상
Actual Result(Test Pass/Non-Pass)	PASS

Test Case Name	**2. testMakeNode**
Objective(Procedure)	-Queue에서의 MakeNode 작동 확인 -테스트용 큐 생성 -큐에 makeNode(5,4,3) (4,3,1) (3,2,2) 호출 -큐의 크기 확인 -큐에 makenode(10,22,2) 한번 더 호출 -큐의 크기 확인
Estimated Result	Queue size 3, 4
Actual Result(Test Pass/Non-Pass)	PASS

Test Case Name	**3. testSearchNode**
Objective(Procedure)	- searchForNode 메소드 정상 작동 확인 -테스트용 큐 생성 -빈 큐에 무작위 큐 검색 시행 후 결과값 null과 대조 -makeNode(15,6,1) (13,2,2) 시행 -각 요청에 대해 검색하여 하나는 requestID, 다른 하나는 목적층 비교
Estimated Result	빈큐 검색결과 null, 검색결과 각각 1,13
Actual Result(Test Pass/Non-Pass)	PASS

Test Case Name	**4. testfindClosestNodeFromPos**
Objective(Procedure)	-Queue에서 현재엘레베이터의 위치와 방향을 넣었을 때 가

	장 가까운 요청 검색하는 기능 -테스트용 큐 생성 -makeNode(5,4,3) (4,3,1) 시행 -엘리베이터가 3층에서 상승중이라고 가정하고 그에 가장 가까운 노드 검색, 그 노드의 층 값을 4와 비교
Estimated Result	가장 가까운 목적지 층 결과값 4
Actual Result(Test Pass/Non-Pass)	PASS

Test Case Name	**5. testOpenNClose**
Objective(Procedure)	-Elevator 클래스 내 문을 여는 메소드 OpenDoor와 CloseDoor 두 메소드 작동 확인 -문을 열고 문의 현재상태 비교 -문을 닫고 문의 현재상태 비교 -문을 다시 닫고 현재상태 비교
Estimated Result	문을 열었을 때 문의상태 1, 문을 닫았을 때 0
Actual Result(Test Pass/Non-Pass)	PASS

Test Case Name	**6. testClearQueue**
Objective(Procedure)	-Queue 클래스 내에 Queue 내용을 모두 비우는 메소드 ClearNode 메소드 작동 확인 -ClearNode 호출 후 Queue가 비었는지 확인
Estimated Result	Clear 후 큐의 크기 0
Actual Result(Test Pass/Non-Pass)	PASS

Test Case Name	**7. testCompClosestNode**
Objective(Procedure)	-큐알고리즘에서 수행하는 엘리베이터의 현재위치에 대해 가장 가까운 요청 검색기능 되는지 확인 -두 노드 생성(5,4,1) (4,3,3) -엘리베이터 위치가 1이라 가정하고 둘중 더가까운 노드 계산
Estimated Result	결과로 되돌려진 요청의 층은 4가 되야 할 것이다.
Actual Result(Test Pass/Non-Pass)	PASS

Test Case Name	**8. testDeQueue**

Objective(Procedure)	-QueueAlgorithm에서의 dequeue 기능 작동 확인 -테스트용 큐알고리즘인스턴스 생성 -enQueue(1,3,7,5)(3,2,5,60) 시행 -큐1에 한 개 큐3에 한 개 넣은 상태에서 큐1의 크기 확인 -큐1에서 방금 넣었던 요청 dequeue -큐1의 크기 0 확인
Estimated Result	큐 1의 크기 1 -> 0
Actual Result(Test Pass/Non-Pass)	PASS

Test Case Name	**9. testEnqueueResultCheck**
Objective(Procedure)	-위에서 테스트한 기능들 enqueue, makenode, findClosestNodesFromposition 엮인 작동 확인 -테스트 알고리즘 생성 -enQueue(1,3,7,5) (2,3,6,7) (3,1,4,6) (3,2,5,60) -선택큐1에 대해 makeNode(11, 22, 3) -엘리베이터 위치를 1이라 가정하고 선택큐1에대해 findclosestNodesfromPosition 시행, 결과 비교
Estimated Result	closestNodesFromPosition 7 리턴
Actual Result(Test Pass/Non-Pass)	PASS

Test Case Name	**10. testDeleteNode**
Objective(Procedure)	-큐 내 deleteNode 작동 확인 -테스트용 큐 생성 -큐 노드 (5,4,3) (4,3,1) 생성 -(5.4.3)에 대한 삭제 요청 -큐의 크기 확인
Estimated Result	큐의 크기 1
Actual Result(Test Pass/Non-Pass)	PASS

Test Case Name	**11. testEnQueueDeleteResultCheck**
Objective(Procedure)	-enqueue했던 요청에 대한 정보 보존 여부 판별 -테스트용 큐알고리즘 생성 -enqueue(1,3,4,5)(2,3,6,7)(3,1,4,6)(3,2,5,60) -선택큐1에대해 makenode(11,22,3) -엘리베이터 위치 1이라 가정하고 선택큐 내 가장 가까운

	요청 검사, -위에서 판별한 가장 가까운 노드 dequeue -다시 1층에서 가장 가까운 노드 검색후 목적지 층수 확인 -searchforNode로 선택큐1에 마지막으로 남아있을 노드 삭제 -선택큐1 크기 다시 확인
Estimated Result	첫번째 가까운층 4, 두번째 11. 삭제후 큐 크기 0
Actual Result(Test Pass/Non-Pass)	PASS

Test Case Name	**12. testCurrentState**
Objective(Procedure)	-현재상태 확인 및 상태변경 후 상태 반영 확인 -테스트용 엘리베이터 생성 -현재 상태 1인지 확인(정상) -현재 상태 2로 변경 -변경여부 확인
Estimated Result	초기 상태1, 다음 상태 2
Actual Result(Test Pass/Non-Pass)	PASS

Test Case Name	**13. testArrivalCalib**
Objective(Procedure)	-엘리베이터 정지상태에서 움직이기 명령 후 이동/정지 상태 변경 확인 -테스트용 엘리베이터 생성 -현재 이동/정지 확인 -이동상태변경 -다시 확인 -arrivalCalibration 적용(정지했다가 일정시간 후 다시 움직임으로 바뀜) -시행 후 이동/정지 확인
Estimated Result	이동/정지 상태 각가 0, 1, ,1
Actual Result(Test Pass/Non-Pass)	PASS

Test Case Name	**14. testHandleDoorRequest**
Objective(Procedure)	- 요청이 들어왔을 때 수행되는 문열기/닫기요청 프로세스 메소드 확인 -테스트용 엘리베이터 생성

	-현재 문상태 확인 -handleDoorRequest로 문열기 요청 전달 -문상태 확인
Estimated Result	문상태 0 - >1
Actual Result(Test Pass/Non-Pass)	PASS

Test Case Name	**15. testSetMaxLoad**
Objective(Procedure)	- 최대하중 설정 기능 확인 - 테스트 엘리베이터 생성 - 현재 한도하중(생성시 하중 1300) 확인 - setMaxLoad로 하중 2500으로 변경 - 최대 하중 확인
Estimated Result	1300 -> 2500
Actual Result(Test Pass/Non-Pass)	PASS