<Software Modeling & Analysis>

NWES
JUnit Test

Team 2
김민우 201111339
김재엽 201111344
최하나 201211386

1. Test Environment

- Team : Team2
- Date : 2013.05.27
- OS : Mac OSX Mevericks 64bit

2.Unit Test

1. Simulator getIncetance();

testGetSimulatorInstance()
시뮬레이션의 인스턴스의 생성을 확인. 싱글턴 패턴을 확인하기위해 존재. null이 아니어야함.
assertNotNull(Simulator.getInstance());
`@Test` `public void testGetSimulatorInstance() {` `    //시뮬레이션의 인스턴스의 생성을 확인. 싱글턴 패턴을 확인하기위해 존재. null이 아니어야함.` `    assertNotNull(Simulator.getInstance());` `}`
Pass

2. Simulator getCabinList()

testGetCabinList()
cabinList의 생성을 확인.
assertNotNull(cabinList);
`@Test` `public void testGetCabinList(){` `    //cabinList의 생성을 확인.` `    assertNotNull(cabinList);` `}`
Pass

3. Simulator GetFloorList()

testGetFloorList()
FloorList의 생성 확인.
assertNotNull(floorList);
`@Test` `public void testGetFloorList(){` `    assertNotNull(floorList);` `}`
Pass

4. Simulator getCabin()

testGetCabinCount()
카빈의 개수를 확인.
assertEquals(2, simulator.getCabin().size());
`@Test` `public void testGetCabinCount(){` `    //카빈의 생성을 테스트` `    assertEquals(2, simulator.getCabin().size());` `}`
Pass

5. Simulator getFloor()

testGetFloorCount()
층의 개수를 확인.
assertEquals(10, simulator.getFloor().size());
`@Test` `public void testGetFloorCount(){` `    //층의 생성을 테스` `    assertEquals(10, simulator.getFloor().size());` `}`
Pass

6. Cabin getCurFloor(), getCabinDirection

testCabinInit()
cabin의 초기화 확인
assertEquals(0, cabin.getCurFloor()); assertEquals(0, cabin.getCainDirection());
`@Test` `public void testCabinInit(){` `    //카빈의 초기화확인.` `    for(Cabin cabin : cabinList){` `        assertEquals(0, cabin.getCurFloor());` `        assertEquals(0, cabin.getCainDirection());` `    }` `}`
Pass

7. Floor getHumanNum(), getWaitingNum()

testFloorInit()
Floor의 초기화 확인
assertEquals(0, floor.getHumanNum()); assertEquals(0, floor.getWaitingNum());
`@Test` `public void testFloorInit(){` `    //각층의 초기화 확인.` `    for(Floor floor : floorList){` `        assertEquals(0, floor.getHumanNum() );` `        assertEquals(0, floor.getWaitingNum() );` `    }` `}`
Pass

8. Cabin getDestination(), getDirection()

testSetDestination()

0 에서 5로 가는 기능 확인.

assertEquals(0, cabinList.get(0).getCurFloor());
assertEquals(5, cabinList.get(0).getDestination());
assertEquals(1, cabinList.get(0).getDirection());
assertEquals(7, cabinList.get(1).getDestination());
assertEquals(1, cabinList.get(1).getDirection());

```
@Test
public void testSetDestination(){
    //0 에서 5 로 가는 기능 확
    assertEquals(0, cabinList.get(0).getCurFloor());

    simulator.goToFloor(0, 5);

    simulator.update(); // 업데이트를 하여 한틱이 돌아가게 한다. --> 사람을 태움.

    assertEquals(5, cabinList.get(0).getDestination());

    simulator.update(); // 업데이트를 하여 한틱이 돌아가게 한다. --> cabin 이동시작.

    assertEquals(1, cabinList.get(0).getDirection()); //방향 체크.

    simulator.update(); // 업데이트를 하여 한틱이 돌아가게 한다. --> cabin 0 이동.

    simulator.goToFloor(0, 7); // 이동 한번더 호출

    simulator.update(); // 업데이트를 하여 한틱이 돌아가게 한다. --> 사람을 태움.

    assertEquals(7, cabinList.get(1).getDestination());//두번째 엘리베이터 목적지 확인.

    simulator.update(); // 업데이트를 하여 한틱이 돌아가게 한다. --> 두번째  이동.

    assertEquals(1, cabinList.get(1).getDirection()); //방향 체크.



}
```

Pass

9 Cabin getCurFloor()

testDestination()
2개의 cabin이 각각 목표층에 도착하는지 확인.
assertEquals(0, cabinList.get(0).getCurFloor()); assertEquals(5, cabinList.get(0).getCurFloor()); assertEquals(1, floorList.get(5).getHumanNum());
`` @Test public void testDestination(){   assertEquals(0, cabinList.get(0).getCurFloor());// 첫번째 cabin 1층에 위치함을 확인.   assertEquals(0, cabinList.get(1).getCurFloor());// 두번째 cabin 1층에 위치함을 확인.   simulator.goToFloor(0, 5);// 첫번째 cabin 5 로 이동.   simulator.update(); // 업데이트를 하여 한틱이 돌아가게 한다. 첫번째 cabin 보냄.   simulator.goToFloor(0, 7);// 두번째 cabin 7 로 이동.   for(int i =0; i<10; i++){   simulator.update(); // 업데이트를 하여 한틱이 돌아가게 한다   } // 10번 반복하여 충분한 틱이 돌아가게 한다.   assertEquals(5, cabinList.get(0).getCurFloor());// 목표층에 도착함을 확인.   assertEquals(7, cabinList.get(1).getCurFloor());// 목표층에 도착함을 확인. } ```
Pass

10. Floor getHumanNum()

testHumanNumInFloor()
사람이 목표층에 도착함을 확인.
assertEquals(0, cabinList.get(0).getCurFloor()); assertEquals(5, cabinList.get(0).getCurFloor()); assertEquals(1, floorList.get(5).getHumanNum());
`@Test` `public void testHumanNumInFloor(){` `    assertEquals(0, cabinList.get(0).getCurFloor());// 첫번째 cabin 1층에 위치함을 확인.` `    simulator.goToFloor(0, 5);// 첫번째 cabin 5 로 이동.` `    for(int i =0; i<10; i++){` `        simulator.update(); // 업데이트를 하여 한틱이 돌아가게 한다` `    } // 10번 반복하여 충분한 틱이 돌아가게 한다.` `    assertEquals(5, cabinList.get(0).getCurFloor());// 목표층에 도착함을 확인.` `    assertEquals(1, floorList.get(5).getHumanNum());` `}`
Pass

11. Cabin getHumanNum()

testHumanNumInCabin()
사람이 cabin에 탑승함을 확인.
assertEquals(0, cabinList.get(0).getCurFloor()); assertEquals(1, cabinList.get(0).getHumanNum());
`@Test` `public void testHumanNumInCabin(){` `    assertEquals(0, cabinList.get(0).getCurFloor());// 첫번째 cabin 1층에 위치함을 확인.` `    simulator.goToFloor(0, 5);// 첫번째 cabin 5 로 이동.` `    simulator.update(); // 업데이트를 하여 한틱이 돌아가게 한다. 사람을 태움.` `    assertEquals(1, cabinList.get(0).getHumanNum()); // 사람이 탐을 확인.` `}`
Pass

12 Simulator callSos()

testSOScall()
sos 요청을 하면 해당층에 cabin에 정지하고 사람이 내림을 확인.
assertEquals(0, cabinList.get(0).getCurFloor()); assertEquals(1, cabinList.get(0).getHumanNum()); assertEquals(3, cabinList.get(0).getCurFloor()); assertEquals(0, floorList.get(4).getHumanNum()); assertEquals(4, cabinList.get(0).getCurFloor()); assertEquals(0, cabinList.get(0).getHumanNum()); assertEquals(1, floorList.get(4).getHumanNum());

```
@Test
public void testSOScall(){
    assertEquals(0, cabinList.get(0).getCurFloor());// 첫번째 cabin 1층에 위치함을 확인.

    simulator.goToFloor(0, 5);// 첫번째 cabin 5 로 이동.

    simulator.update(); // 업데이트를 하여 한틱이 돌아가게 한다. 사람을 태움.

    assertEquals(1, cabinList.get(0).getHumanNum());// 1명 승객 확인.

    simulator.update(); // 업데이트를 하여 한틱이 돌아가게 한다. 한층 이동.
    simulator.update(); // 업데이트를 하여 한틱이 돌아가게 한다. 한층 이동.
    simulator.update(); // 업데이트를 하여 한틱이 돌아가게 한다. 한층 이동.

    assertEquals(3, cabinList.get(0).getCurFloor()); // 현재 층 확인.

    simulator.update(); // 업데이트를 하여 한틱이 돌아가게 한다. 한층 이동.

    assertEquals(0, floorList.get(4).getHumanNum()); // 비상정지할 4 에서 사람이 없음을 확인.

    simulator.callSOS( 0 ); // 첫번째 cabin sos 요청.
    assertEquals(4, cabinList.get(0).getCurFloor()); // 4 에서 비상정지 확인.
    assertEquals(0, cabinList.get(0).getHumanNum()); // 사람이 다내림을 확인.
    assertEquals(1, floorList.get(4).getHumanNum()); // 사람이 층에 존재함을 확인.

}
```

Pass

13 Cabin getCurFloor(), getHumanNum()

testCabinSecondCall()
assertEquals(0, cabinList.get(0).getCurFloor()); assertEquals(5, cabinList.get(0).getCurFloor()); assertEquals(0, cabinList.get(0).getHumanNum()); assertEquals(1, floorList.get(5).getHumanNum()); assertEquals(0, floorList.get(5).getHumanNum()); assertEquals(1, cabinList.get(0).getHumanNum()); assertEquals(1, floorList.get(8).getHumanNum()); assertEquals(0, cabinList.get(0).getHumanNum());
@Test public void testCabinSecondCall(){   assertEquals(0, cabinList.get(0).getCurFloor());// 첫번째 cabin 1층에 위치함을 확인.   simulator.goToFloor(0, 5);// 첫번째 cabin 5 로 이동.   for(int i =0; i<10; i++){   simulator.update(); // 업데이트를 하여 한틱이 돌아가게 한다   } // 10번 반복하여 충분한 틱이 돌아가게 한다.   assertEquals(5, cabinList.get(0).getCurFloor());// 목표층에 도착함을 확인.   assertEquals(0, cabinList.get(0).getHumanNum());// cabin에 사람이 없음을 확인.   assertEquals(1, floorList.get(5).getHumanNum());   simulator.goToFloor(5, 8); // 첫번째 cabin이 가장가까운 cabin으로 판정되어 5에서 8로 이동함을 확인.   simulator.update();   assertEquals(0, floorList.get(5).getHumanNum());// 사람이 없음을 확인.   assertEquals(1, cabinList.get(0).getHumanNum());// 사람이 탑승으 확인.   for(int i =0; i<10; i++){   simulator.update(); // 업데이트를 하여 한틱이 돌아가게 한다   } // 10번 반복하여 충분한 틱이 돌아가게 한다.   assertEquals(1, floorList.get(8).getHumanNum()); // floor 8 에 사람이 도착함을 확인.   assertEquals(0, cabinList.get(0).getHumanNum()); // 사람이 내림을 확인. }
Pass

3. 결과

총 13가지의 Test case중 13개 성공
객체 유닛 기반의 테스트이기보다는 스토리 기반의 테스트이기 때문에 유닛간 객체성 및 독립성이 약함을 알 수 있다.
의존성이 강함을 알 수 있다.
객체성을 지향하는 방향으로 개선해야 한다.