



소프트웨어 모델링

RE-ason Elevator

OOP 1st Cycle & Unit Test

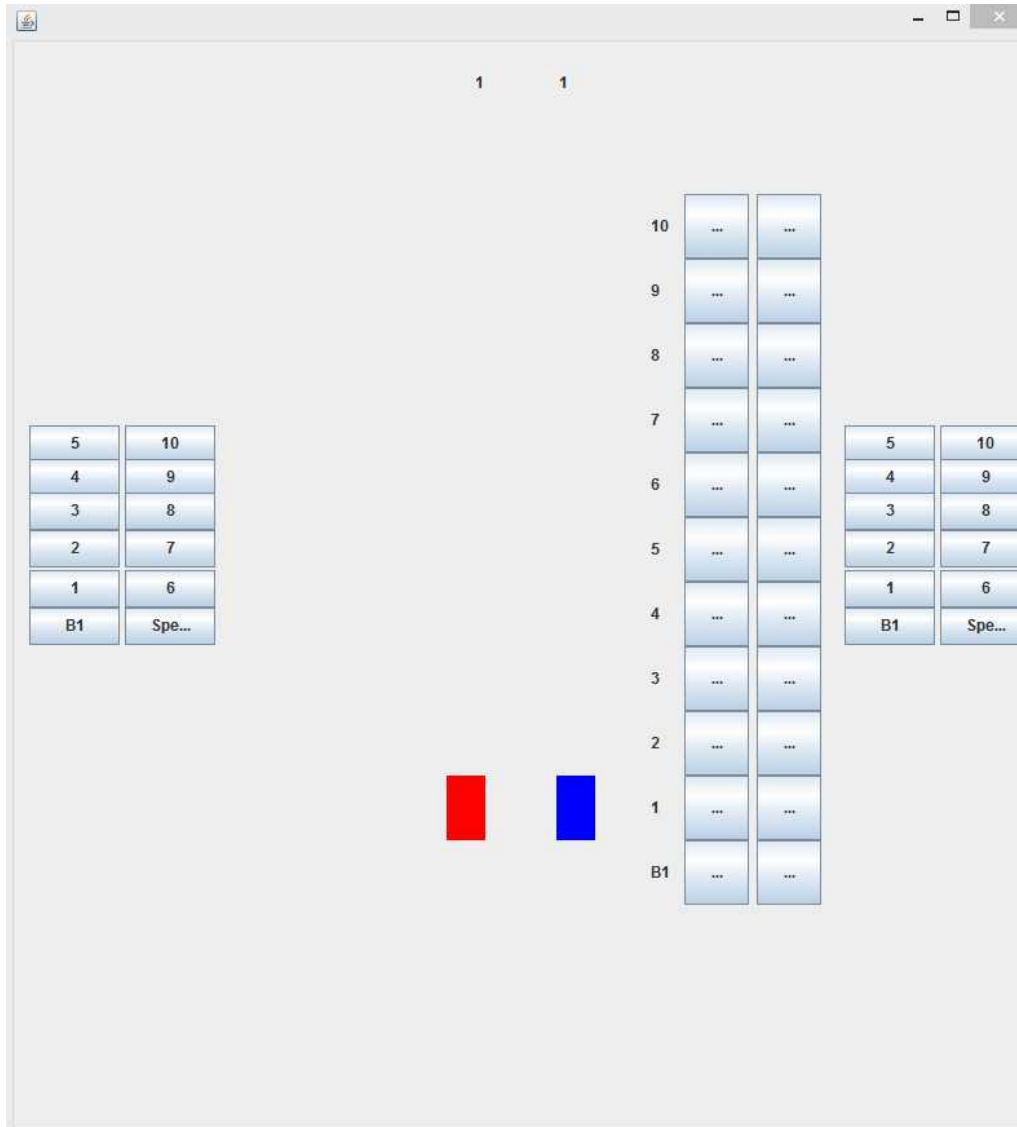
The Team 1

박현규 200911393
김도현 201211328
송해찬 201211358



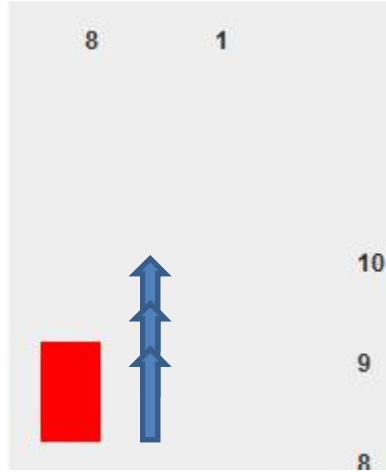
DEMO

DEMO - 메인화면

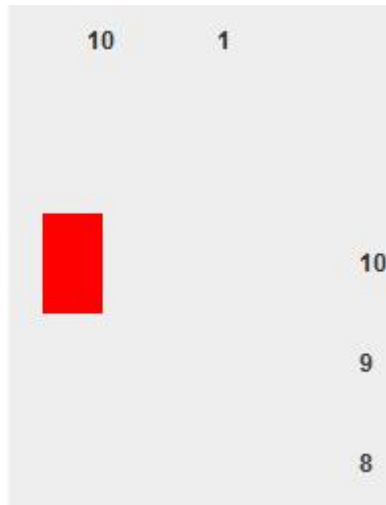


DEMO - 층 선택 버튼

5	10
4	9
3	8
2	7
1	6
B1	Spe...

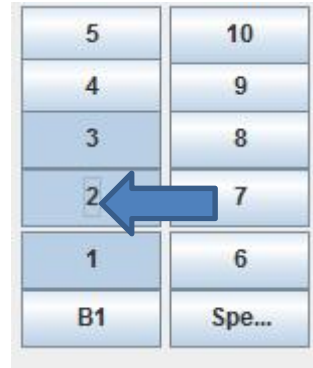
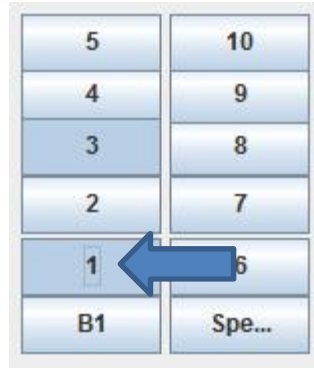
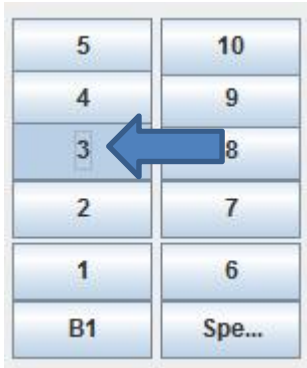


5	10
4	9
3	8
2	7
1	6
B1	Spe...



8
8
8
8
8
8
8
10
10
10
10
10
10
10

DEMO - 층 선택 & 이동

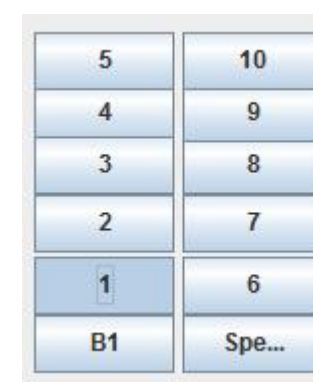
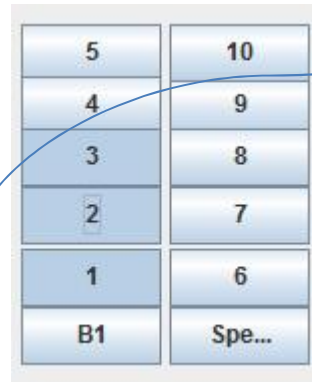


우선순위큐

층 선택
버튼 입력



엘리베이터 이동



DEMO – 회원 버튼

A 2x6 grid of buttons. The buttons are arranged in two columns and six rows. The values are: Row 1: 5, 10; Row 2: 4, 9; Row 3: 3, 8; Row 4: 2, 7; Row 5: 1, 6; Row 6: B1, Spe... The button with '3' is highlighted with a blue border and a small white square, indicating it is the active button.

비활성화 된 버튼들

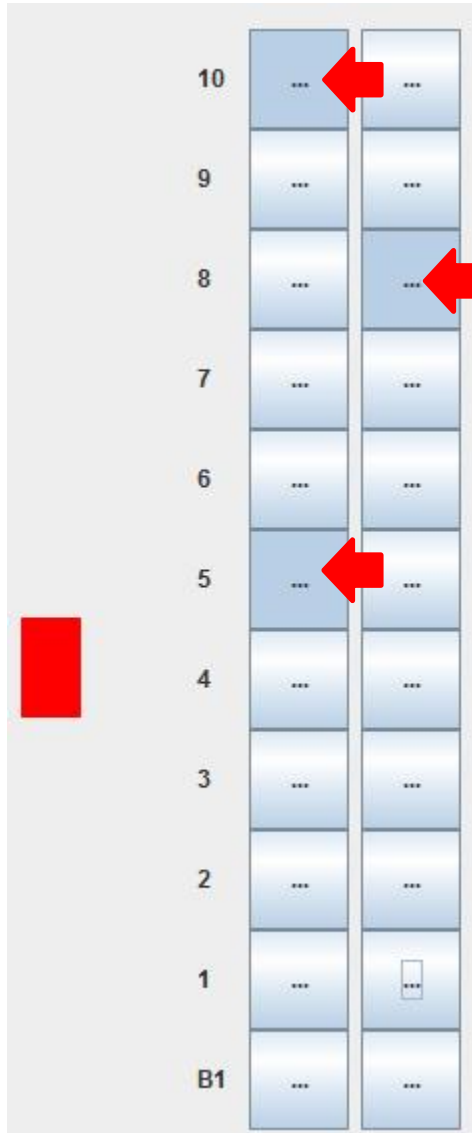
A 2x6 grid of buttons. The buttons are arranged in two columns and six rows. The values are: Row 1: 5, 10; Row 2: 4, 9; Row 3: 3, 8; Row 4: 2, 7; Row 5: 1, 6; Row 6: B1, Spe... All buttons in this grid are highlighted with a blue gradient, indicating they are all active.

회원 기능으로
버튼을 활성화 시킴

A 2x6 grid of buttons. The buttons are arranged in two columns and six rows. The values are: Row 1: 5, 10; Row 2: 4, 9; Row 3: 3, 8; Row 4: 2, 7; Row 5: 1, 6; Row 6: B1, Spe... The buttons with '3', '8', and '1' are highlighted with a blue gradient, indicating they are active.

비활성화 된 층
이동가능

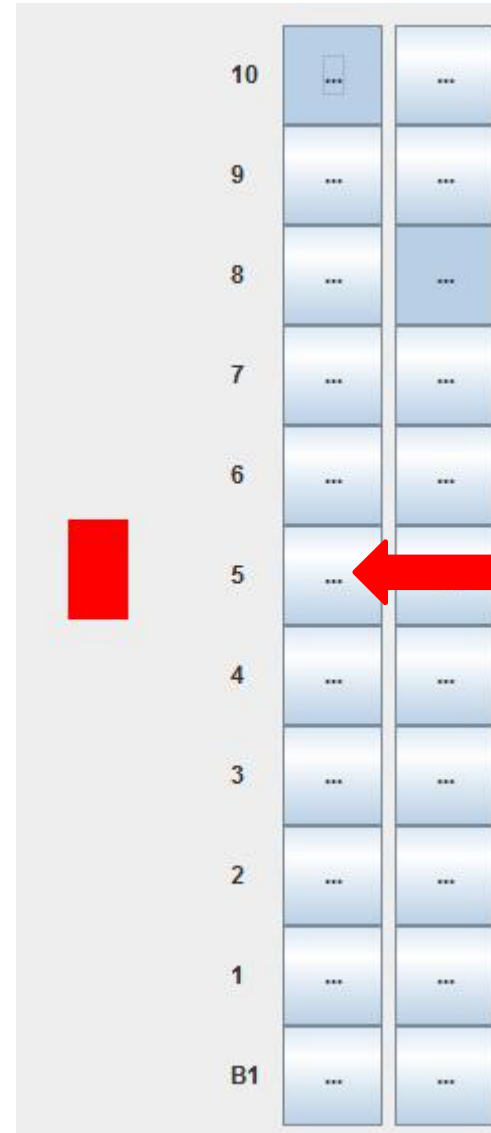
DEMO – 각 층 위 아래 버튼



위 아래 버튼으로

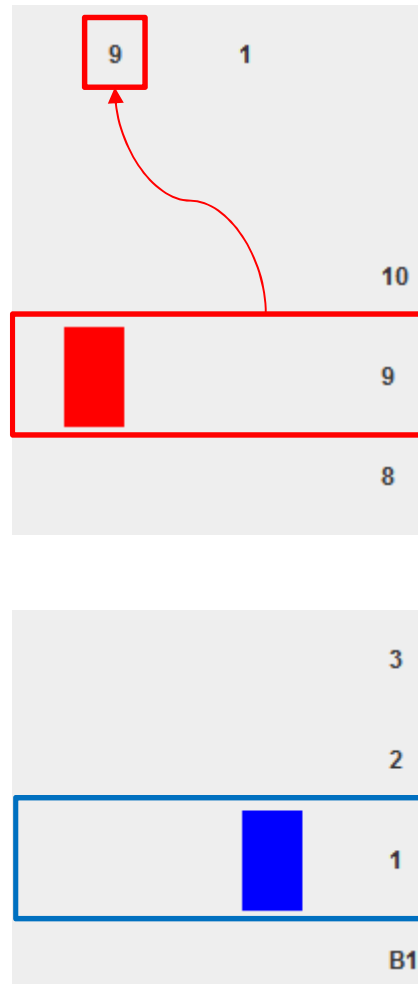
각 층에서

엘리베이터를
부른다



층에 도착하면
버튼이 비활성화

DEMO – 층 현황 디스플레이





UNIT TEST

UNIT TEST Order

제외항목

Listener

About GUI...

Class 변수

검사항목

Queue

Movement Algorithm

State

Direction

Queue

Direction

```
Finished after 0.007 seconds
Runs: 3/3      Errors: 0      Failures: 0

testQueue [Runner: JUnit 4] (0.000 s)
  testRemoveMin (0.000 s)
  testSize (0.000 s)
  testGetDirection (0.000 s)

1
queue left = new queue();

@Test
public void testSize() {
    assertEquals(left.size(), 1);
}

@Test
public void testRemoveMin() {
    left.insert(5);
    assertNotNull(left.removeMin());
}

@Test
public void testGetDirection() {
    assertNotNull(left.getDirection());
}
```

size()

- a. 초기 queue 사이즈 == 1
- b. Pressed button == queue.insert
- c. queue 사이즈 증가

removeMin()

- a. 엘리베이터 목표 층에 도착
- b. queue에서 빠져나감
- c. 도착한 층의 index 반환

getDirection()

- a. 엘리베이터 방향 == 큐의 방향
- b. 위/아래 방향 반환

Movement Algorithm

```
Finished after 0.009 seconds
Runs: 1/1      Errors: 0      Failures: 0
testMovement [Runner: JUnit 4] (0.000 s)
@Test
public void test() {
    movementAlgorithm ma = new movementAlgorithm();
    ma.init();
    assertNotNull(ma.getLeftQueue());
}
```

getLeftQueue()

- a. movementAlgorithm 안에서 queue 생성
- b. queue 반환

State

```
Finished after 0.042 seconds
Runs: 1/1     Errors: 0     Failures: 0
-----
testLeftObject [Runner: JUnit 4] (0.000 s)
  testGetState (0.000 s)
    leftObjectPanel l = new leftObjectPanel();

@Test
public void testGetState() {
    assertNotNull(l.getState());
}
```

getState()

- a. 엘리베이터 object의 현재 위치 계산
- b. 현재 위치 반환