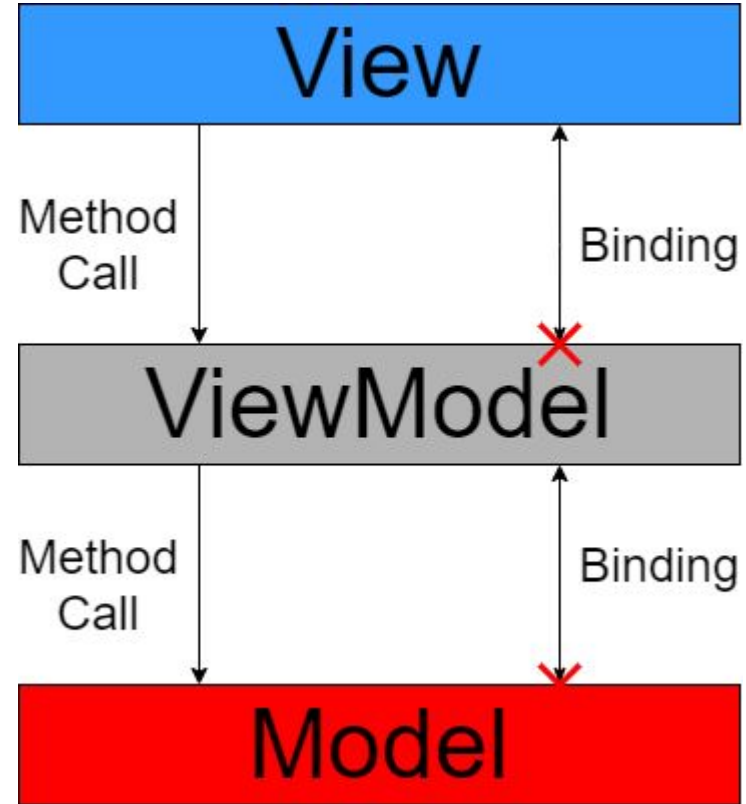


# Elevator Control System

Hauptmann, Hubl, Grüneis

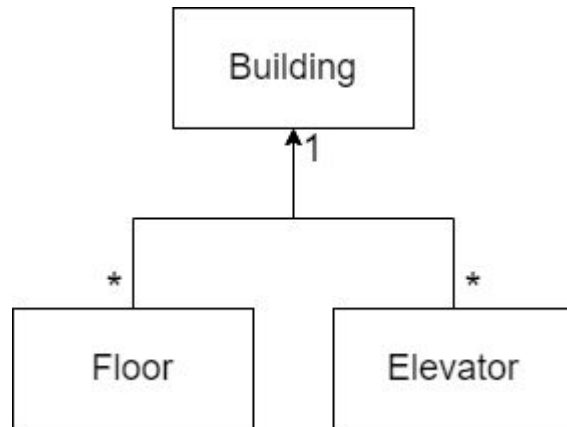
# General architecture

- Web said MVVM
- Overkill?
- We didn't fully follow the guideline
- Model = Building
- ViewModel translates Model
  - e.g. position of pixel distance of elevator
  - e.g. target to color of floor
  - e.g. double speed to speed string with " ft/s"
- Clicks are sent to view model that calls model function
  - e.g. set target of elevator
- Makes combination of ViewModel and View a bit more testable in our opinion -> show test



# Building up the data model

- Three classes
- Building contains list of floors and elevators
- Converter creates new building
- Values copied if clock ticks ok
- Else thrown away



```
public boolean convert(Building building) throws RemoteException {
    Long firstClockTick = elevatorConnection.getClockTick();
    if (!firstClockTick.equals(lastClockTick)) {
        List<Floor> floors = getFloorsFromInterface();
        if (!compareTicks(firstClockTick, elevatorConnection.getClockTick())) {
            return false;
        }
        List<Elevator> elevators = getElevatorsFromInterface(floors);
        if (!compareTicks(firstClockTick, elevatorConnection.getClockTick())) {
            return false;
        }
        int floorHeight = getFloorHeight();
        lastClockTick = elevatorConnection.getClockTick();
        if (compareTicks(firstClockTick, lastClockTick)) {
            Building newBuildingMapping = new Building(elevators, floors, floorHeight);
            building.copyValues(newBuildingMapping);
            return true;
        }
    }
    return false;
}
```

# Testing with mocks

```
@Test
void testConvertIfItCreatesRightAmountOfFloors() throws RemoteException {
    when(interfaceMock.getElevatorNum()).thenReturn(0);
    when(interfaceMock.getFloorNum()).thenReturn(5);

    Building building = new Building();
    interfaceToModelConverter.convert(building);

    assertEquals( expected: 5, building.getNumberOfFloors());
}
```

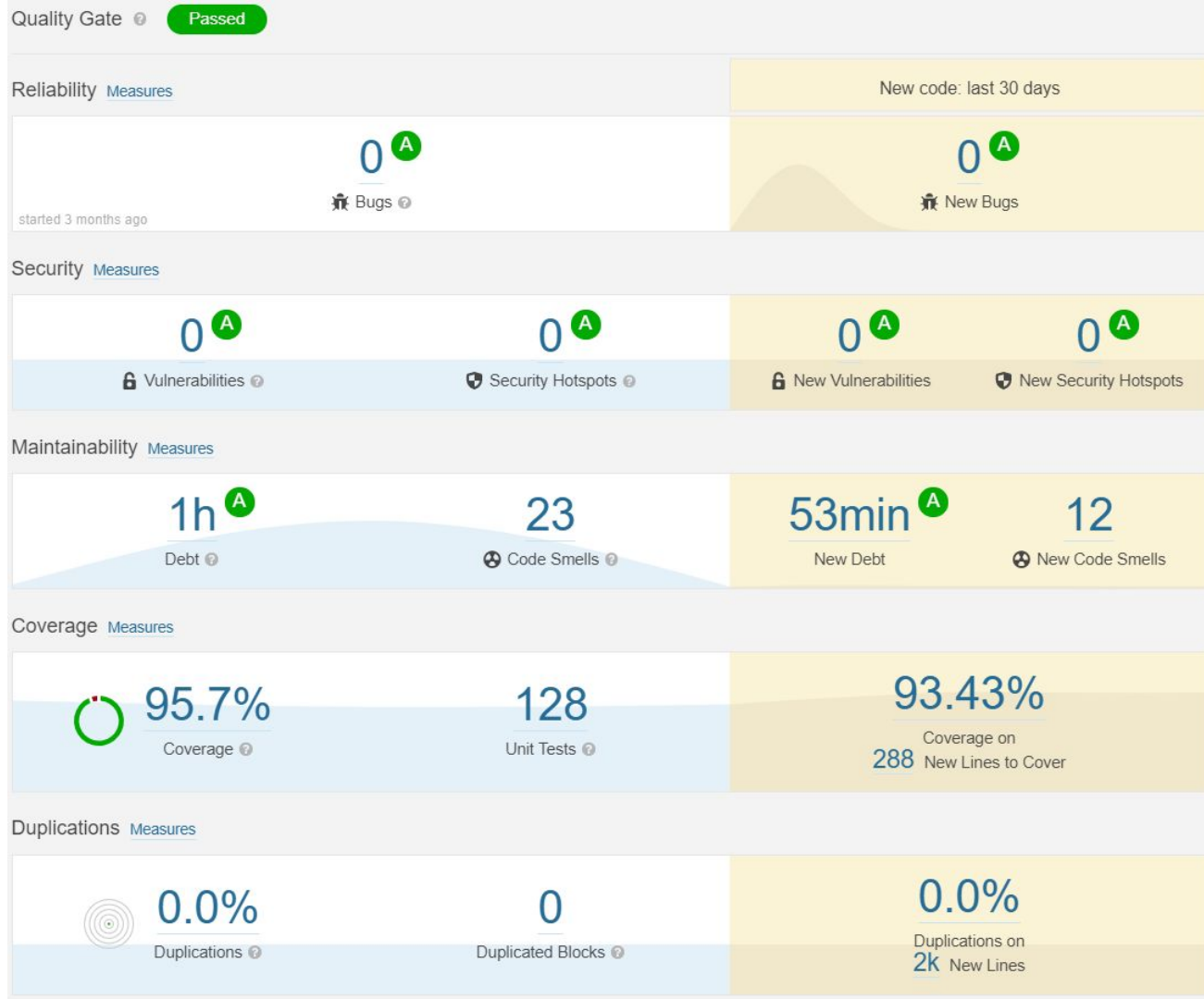
```
@Test
void testConvertIfButtonUpIsConvertedCorrectly() throws RemoteException {
    when(interfaceMock.getElevatorNum()).thenReturn(0);
    when(interfaceMock.getFloorNum()).thenReturn(2);
    when(interfaceMock.getFloorButtonUp(0)).thenReturn(true);
    when(interfaceMock.getFloorButtonUp(1)).thenReturn(false);

    Building building = new Building();
    interfaceToModelConverter.convert(building);

    assertTrue(building.getFloor( index: 0).isButtonUp());
    assertFalse(building.getFloor( index: 1).isButtonUp());
}
```

Shown later again

- Reduced smells
- Increased coverage
- Fixed bugs
- Rather cool tool
- Sonarlint IntelliJ



# GUI Mockup

General Info	Floors	Elevator 1	
<div>System Status All Systems are working</div> <div>Log Messages:</div>	<div><div><div>✓</div><div>^</div></div><div>8</div></div>	<div><div><div>X</div><div>X</div></div></div>	
	<div><div><div>✓</div><div>^</div></div><div>7</div></div>	<div></div>	
	<div><div><div>✓</div><div>^</div></div><div>6</div></div>	<div></div>	
	<div><div><div>✓</div><div>^</div></div><div>5</div></div>	<div></div>	
	<div><div><div>✓</div><div>^</div></div><div>4</div></div>	<div></div>	
	<div><div><div>✓</div><div>^</div></div><div>3</div></div>	<div></div>	
	<div><div><div>✓</div><div>^</div></div><div>2</div></div>	<div></div>	
	<div><div><div>✓</div><div>^</div></div><div>1</div></div>	<div></div>	<div>Pressable in manual mode to set target</div>
		<div><div><div>7</div><div>8</div><div>2</div><div>0</div><div>3</div><div>9</div><div>4</div><div>5</div><div>6</div></div><div>Capacity: _____</div><div>Weight: _____</div><div>Target: _____</div><div>Speed: _____</div><div>Acceleration: _____</div></div>	

# GUI Realization

- Modular
- No SceneBuilder
- Tested classes
- Show tests

The screenshot displays a Java Swing window titled "Elevator Control GUI". It features a sidebar with a "view" menu listing 18 components, each preceded by a blue circular icon. The main area is divided into four panels, each representing an elevator. The top-left panel shows the "System status" as "connected" in green and an empty "Error Messages" list. The top-right panel has three "Manual" toggle switches, all currently turned off. The bottom-right panel contains three small 3x2 floor indicator grids, each with a "Capacity" of 10. The first grid shows floor 1 selected, the second shows floor 2 selected, and the third shows floor 1 selected. Below each grid are labels for "Weight", "Target", "Speed", "Acceleration", and "Door Status".

**view**

- ElevatorAnimationOverlayPane
- ElevatorAnimationPane
- ElevatorBackgroundLayoutPane
- ElevatorButtonInfoPane
- ElevatorControlCenterPane
- ElevatorInfoPane
- ElevatorModeButtonPane
- ElevatorPane
- ElevatorSingleFloor
- ElevatorTextInfoPane
- EmptyElevatorModeButtonPane
- FloorsPane
- GeneralInfoPane
- SingleFilledFloorPane
- SingleFloorPane
- SwitchButton

**System status**  
connected  
Error Messages

**Manual** ☐ **Manual** ☐ **Manual** ☐

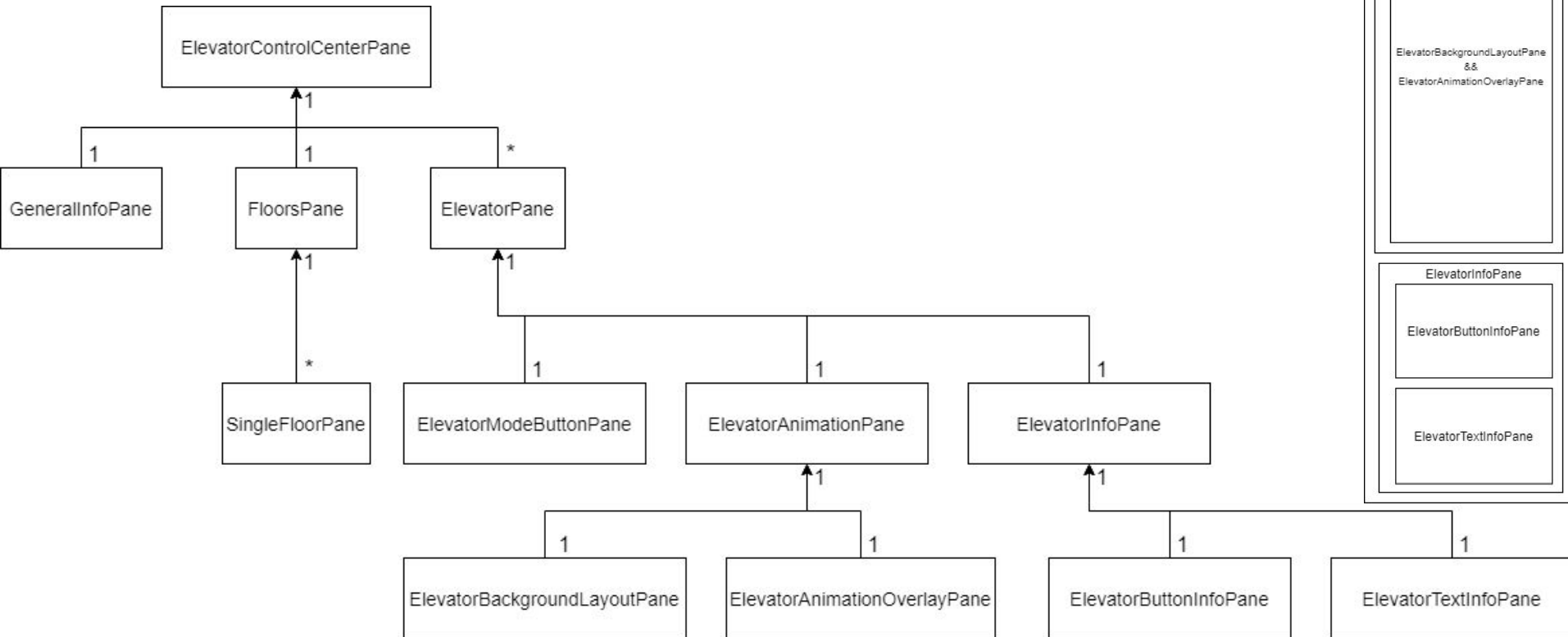
Floor	Up	Down	Capacity	Weight	Target	Speed	Acceleration	Door Status
10	^	v	10	0 lbs	1	0 ft/s	10 ft/s <sup>2</sup>	OPEN
9	^	v	9	0 lbs	0	-5 ft/s	10 ft/s <sup>2</sup>	CLOSED
8	^	v	8	0 lbs	6	0 ft/s	10 ft/s <sup>2</sup>	OPEN
7	^	v	7	0 lbs	0	-5 ft/s	10 ft/s <sup>2</sup>	CLOSED
6	^	v	6	0 lbs	1	0 ft/s	10 ft/s <sup>2</sup>	OPEN
5	^	v	5	0 lbs	0	-5 ft/s	10 ft/s <sup>2</sup>	CLOSED
4	^	v	4	0 lbs	0	-5 ft/s	10 ft/s <sup>2</sup>	CLOSED
3	^	v	3	0 lbs	0	-5 ft/s	10 ft/s <sup>2</sup>	CLOSED
2	^	v	2	110 lbs	0	-5 ft/s	10 ft/s <sup>2</sup>	CLOSED
1	^	v	1	0 lbs	6	0 ft/s	10 ft/s <sup>2</sup>	OPEN

**Capacity: 10**  
**Weight: 0 lbs**  
**Target: 1**  
**Speed: 0 ft/s**  
**Acceleration: 10 ft/s<sup>2</sup>**  
**Door Status: OPEN**

**Capacity: 10**  
**Weight: 110 lbs**  
**Target: 0**  
**Speed: -5 ft/s**  
**Acceleration: 10 ft/s<sup>2</sup>**  
**Door Status: CLOSED**

**Capacity: 10**  
**Weight: 0 lbs**  
**Target: 6**  
**Speed: 0 ft/s**  
**Acceleration: 10 ft/s<sup>2</sup>**  
**Door Status: OPEN**

# GUI Structure





# GUI Tests

## Testing API calls - enable manual mode



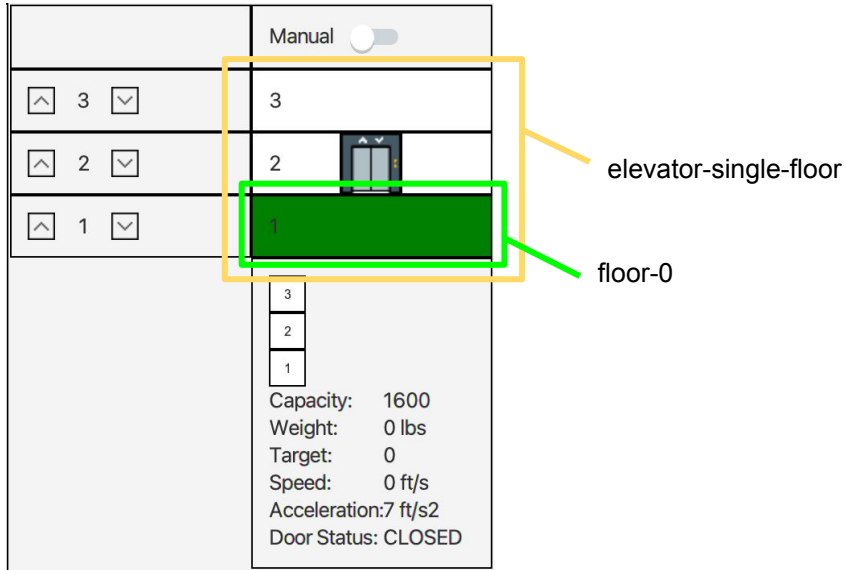
```
robot = new FxRobot();

Node elevatorModeSwitch = robot.lookup(".elevator-pane")
    .lookup("#elevator-0")
    .lookup(".elevator-mode-button-pane")
    .query();

robot.clickOn(elevatorModeSwitch);
```

# GUI Tests

## Testing API calls - select target floor




```
Node floor = robot.lookup(".elevator-single-floor")
    .lookup("#floor-0").query();

robot.clickOn(floor);
verify(interfaceMock, times(1)).setTarget(0, 0);
```

# GUI Tests

## Testing API calls - backend changes

	Manual <input type="checkbox"/>
⬆ 3 ⬇	3
⬆ 2 ⬇	2 
⬆ 1 ⬇	1
<div><div>3</div><div>2</div><div>1</div><div>Capacity: 1600</div><div>Weight: 0 lbs</div><div>Target: 0</div><div>Speed: 0 ft/s</div><div>Acceleration: 7 ft/s<sup>2</sup></div><div>Door Status: CLOSED</div></div>	

```
// initial mock return value
when(interfaceMock.getElevatorWeight(0)).thenReturn(0);

// modified mock return value
when(interfaceMock.getClockTick()).thenReturn((long) 1);
when(interfaceMock.getElevatorWeight(0)).thenReturn(100);
```

# GUI Tests

## Testing API calls - backend changes

The screenshot shows a GUI for an elevator system. At the top, there is a 'Manual' toggle switch. Below it are three buttons labeled 1, 2, and 3, each with an up arrow and a down arrow. A small elevator icon is positioned between the buttons. Below the buttons is a large panel containing a list of buttons (1, 2, 3) and a text area displaying the following information: Capacity: 1600, Weight: 100 lbs, Target: 0, Speed: 0 ft/s, Acceleration: 7 ft/s<sup>2</sup>, and Door Status: CLOSED. Annotations with colored boxes and lines identify specific elements: a red box around the entire panel is labeled 'elevator-text-info-pane'; a yellow box around the text area is labeled 'elevator-0'; and a green box around the 'Weight: 100 lbs' text is labeled 'weightLabel'.

Manual ☐

3 2 1

3 2 1

Capacity: 1600  
Weight: 100 lbs  
Target: 0  
Speed: 0 ft/s  
Acceleration: 7 ft/s<sup>2</sup>  
Door Status: CLOSED

elevator-text-info-pane

elevator-0

weightLabel



// Reading out weight value

```
NodeQuery query = robot.lookup(".elevator-text-info-pane")  
    .lookup("#elevator-0").lookup("#weightLabel");
```

```
FxAssert.verifyThat(query, LabeledMatchers  
    .hasText("100 lbs"));
```

# Bind elevator to simulator

- `RMIElevatorAdapter`  
connects to the simulator via RMI and `Naming.lookup`
- uses `InterfaceToModelConverter` to convert the data to our model
- handles `RemoteExceptions` and automatically tries to reconnect
- App calls `updateBuilding(Building building)` to update the model

# Testing Connection

- RMIElevatorAdapterTest
  - mocks the interface: `mock(IElevator.class, Mockito.withSettings().serializable())`
  - creates RMI registry and binds the interface
  - test reconnecting after unbinding and rebinding the mock
- AppTest
  - create new RMI registry at different port
  - mock and bind interface
  - use FxRobot to find label with ***connecting*** (after error) or ***connected*** text

# Automatic Mode

- One list of targets for each elevator in automatic mode
- Next target is set to the next entry in the list
- Fill lists as following:
  1. Add all pressed individual elevator floor buttons
  2. Distribute general floor buttons between elevators
- When possible we add a new floor between targets, else we add it to the end

# Automatic Mode Test

- Have to create the correct setup for each test case ➡ huge tests
- Show more details in code later


```
@Test
void testUpdateStateInitialElevatorButtonsOnly() {
    RMIElevatorAdapter rmiAdapterMock = mock(RMIElevatorAdapter.class);
    Floor floor01 = new Floor( number: 0, buttonUp: false, buttonDown: false);
    Floor floor02 = new Floor( number: 1, buttonUp: false, buttonDown: true);
    Floor floor03 = new Floor( number: 2, buttonUp: false, buttonDown: true);
    List<Boolean> elevatorFloorButtons = new ArrayList<>();
    elevatorFloorButtons.add(false);
    elevatorFloorButtons.add(true);
    elevatorFloorButtons.add(true);
    List<Integer> listOfServicedFloors = new ArrayList<>();
    listOfServicedFloors.add(0);
    listOfServicedFloors.add(1);
    listOfServicedFloors.add(2);
    Elevator elevator = new Elevator( number: 0, committedDirection: 1, acceleration: 1, doorSta
    List<Elevator> elevators = new ArrayList<>();
    List<Floor> floors = new ArrayList<>();
    elevators.add(elevator);
    floors.add(floor01);
    floors.add(floor02);
    floors.add(floor03);
    Building building = new Building(elevators, floors, floorHeight: 50);

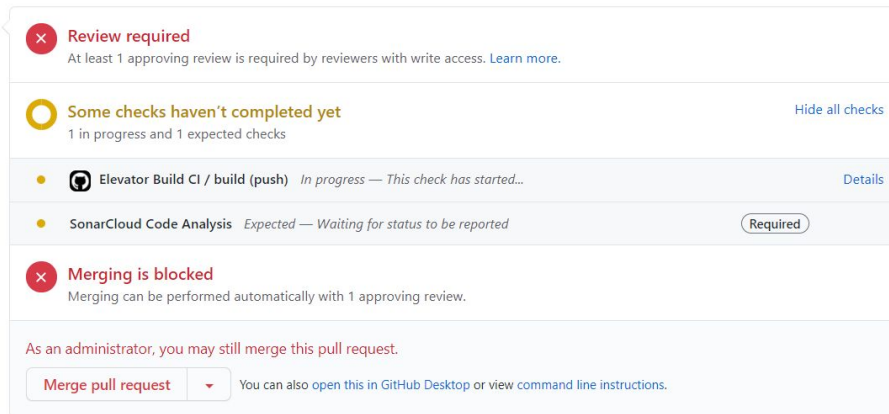
    ArgumentCaptor<Integer> elevatorCaptor = ArgumentCaptor.forClass(Integer.class);
    ArgumentCaptor<Integer> floorCaptor = ArgumentCaptor.forClass(Integer.class);
    new AutomaticStateController(rmiAdapterMock, building);

    verify(rmiAdapterMock, times( wantedNumberOfInvocations: 1)).setTarget(elevatorCaptor.c
    assertEquals( expected: 0, elevatorCaptor.getValue());
    assertEquals( expected: 1, floorCaptor.getValue());
}
```



# How did we assure product quality

- Build Automation (GitHub Actions)
- Sonarcloud  
(code coverage, static analysis)
- Pull Requests
- Issue management & Communication 
- Generally fix bugs when seeing them



**Review required**  
At least 1 approving review is required by reviewers with write access. [Learn more.](#)

**Some checks haven't completed yet** [Hide all checks](#)  
1 in progress and 1 expected checks

- Elevator Build CI / build (push)** In progress — This check has started... [Details](#)
- SonarCloud Code Analysis** Expected — Waiting for status to be reported **Required**

**Merging is blocked**  
Merging can be performed automatically with 1 approving review.

As an administrator, you may still merge this pull request.

**Merge pull request** You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

# Dev workflow

- Check assignment
- Create feature branch
- Try to write tests before (really hard)
- Implement
- Add further tests
- Static analysis (sonarcloud)
- Polish code
- Reduce commits -> interactive rebase
- Pull request
- Build, static analysis and review to merge pull request

# Demo

<div>System status</div> <div>connected</div> <div>Error Messages</div>		Manual <input type="checkbox"/>	Manual <input type="checkbox"/>	Manual <input checked="" type="checkbox"/>	Manual <input checked="" type="checkbox"/>	Manual <input type="checkbox"/>	Manual <input type="checkbox"/>
	<div><div>⬆</div>10<div>⬇</div></div>	10	10	10	10	10	10
	<div><div>⬆</div>9<div>⬇</div></div>	9	9	9	9	9	9
	<div><div>⬆</div>8<div>⬇</div></div>	8	8	8	8	8	8
	<div><div>⬆</div>7<div>⬇</div></div>	7	7	7	7	7	7
	<div><div>⬆</div>6<div>⬇</div></div>	6	6	6	6	6	6
	<div><div>⬆</div>5<div>⬇</div></div>	5	5	5	5	5	5
	<div><div>⬆</div>4<div>⬇</div></div>	4	4	4	4	4	4
	<div><div>⬆</div>3<div>⬇</div></div>	3	3	3	3	3	3
	<div><div>⬆</div>2<div>⬇</div></div>	2	2	2	2	2	2
	<div><div>⬆</div>1<div>⬇</div></div>	1	1	1	1	1	1
		<div><div><div><div>9</div><div>10</div><div>7</div><div>8</div><div>5</div><div>6</div><div>3</div><div>4</div><div>1</div><div>2</div></div><div>Capacity: 10 Weight: 457 lbs Target: 2 Speed: 0 ft/s Acceleration:10 ft/s<sup>2</sup> Door Status: OPEN</div></div></div> <div><div><div><div>9</div><div>10</div><div>7</div><div>8</div><div>5</div><div>6</div><div>3</div><div>4</div><div>1</div><div>2</div></div><div>Capacity: 10 Weight: 564 lbs Target: 9 Speed: 0 ft/s Acceleration:10 ft/s<sup>2</sup> Door Status: CLOSING</div></div></div> <div><div><div><div>9</div><div>10</div><div>7</div><div>8</div><div>5</div><div>6</div><div>3</div><div>4</div><div>1</div><div>2</div></div><div>Capacity: 10 Weight: 844 lbs Target: 7 Speed: 5 ft/s Acceleration:10 ft/s<sup>2</sup> Door Status: CLOSED</div></div></div> <div><div><div><div>9</div><div>10</div><div>7</div><div>8</div><div>5</div><div>6</div><div>3</div><div>4</div><div>1</div><div>2</div></div><div>Capacity: 10 Weight: 0 lbs Target: 3 Speed: 5 ft/s Acceleration:10 ft/s<sup>2</sup> Door Status: CLOSED</div></div></div> <div><div><div><div>9</div><div>10</div><div>7</div><div>8</div><div>5</div><div>6</div><div>3</div><div>4</div><div>1</div><div>2</div></div><div>Capacity: 10 Weight: 0 lbs Target: 7 Speed: 2 ft/s Acceleration:10 ft/s<sup>2</sup> Door Status: CLOSED</div></div></div> <div><div><div><div>9</div><div>10</div><div>7</div><div>8</div><div>5</div><div>6</div><div>3</div><div>4</div><div>1</div><div>2</div></div><div>Capacity: 10 Weight: 0 lbs Target: 4 Speed: 0 ft/s Acceleration:10 ft/s<sup>2</sup> Door Status: OPEN</div></div></div>					