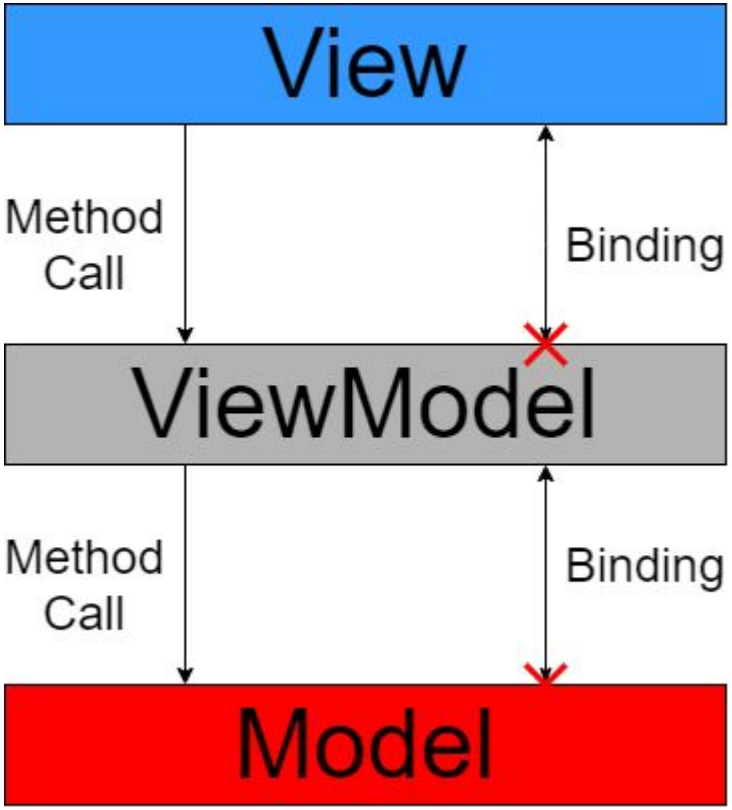
Elevator Control System

Hauptmann, Hubl, Grüneis

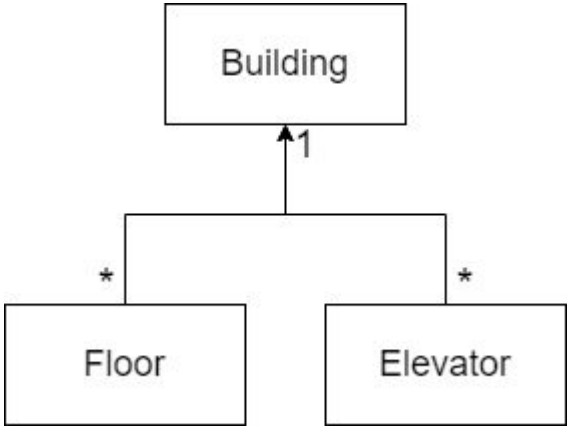
General architecture

- 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




```java
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

```java
@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());
}
```

```java
@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

sonarcloud

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

Quality Gate Passed

Reliability Measures

New code: last 30 days

0 A Bugs

started 3 months ago

0 A New Bugs

Security Measures

0 A Vulnerabilities

0 A Security Hotspots

0 A New Vulnerabilities

0 A New Security Hotspots

Maintainability Measures

1h A Debt

23 Code Smells

53min A New Debt

12 New Code Smells

Coverage Measures

95.7% Coverage

128 Unit Tests

93.43% Coverage on 288 New Lines to Cover

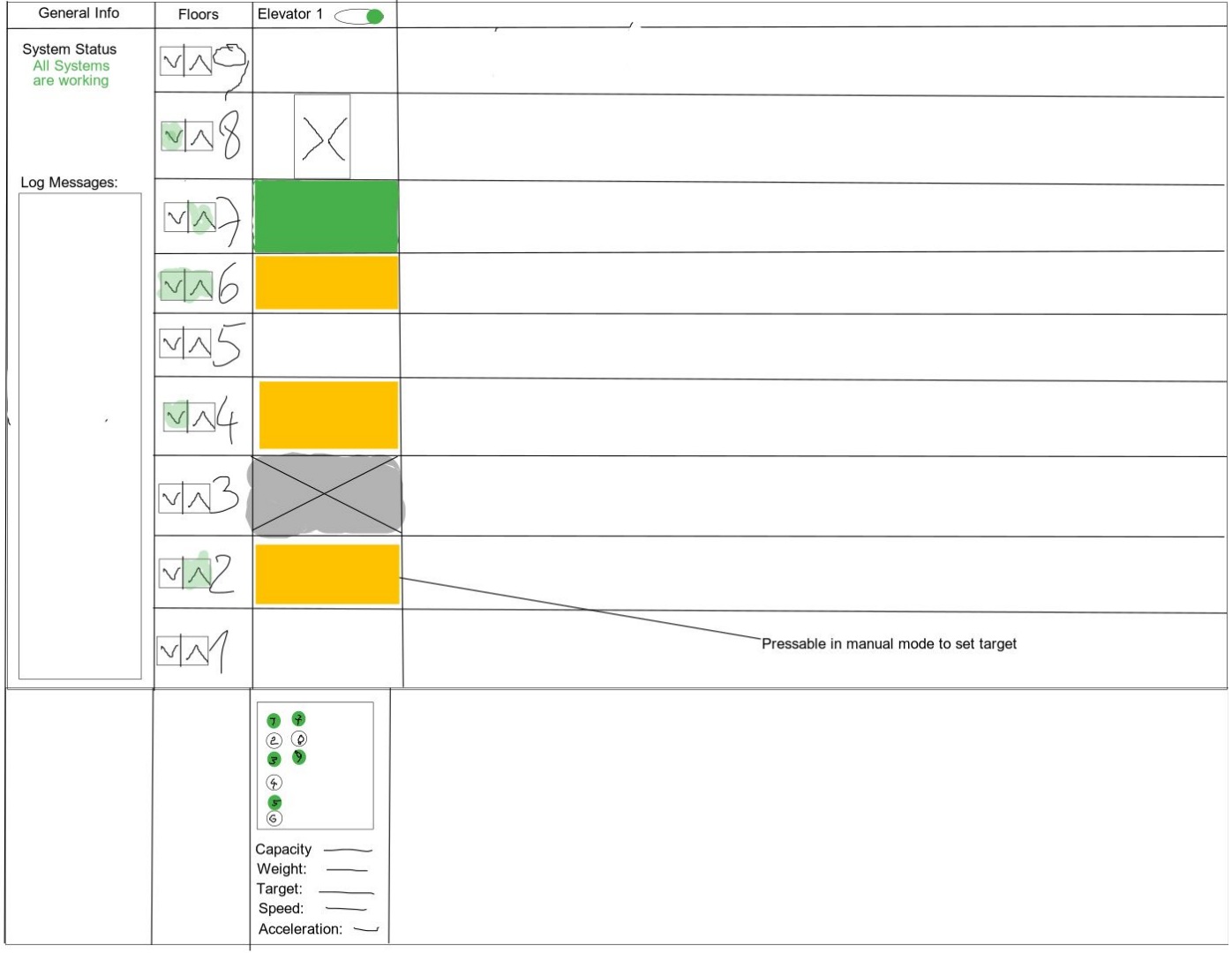
Duplications Measures

0.0% Duplications

0 Duplicated Blocks

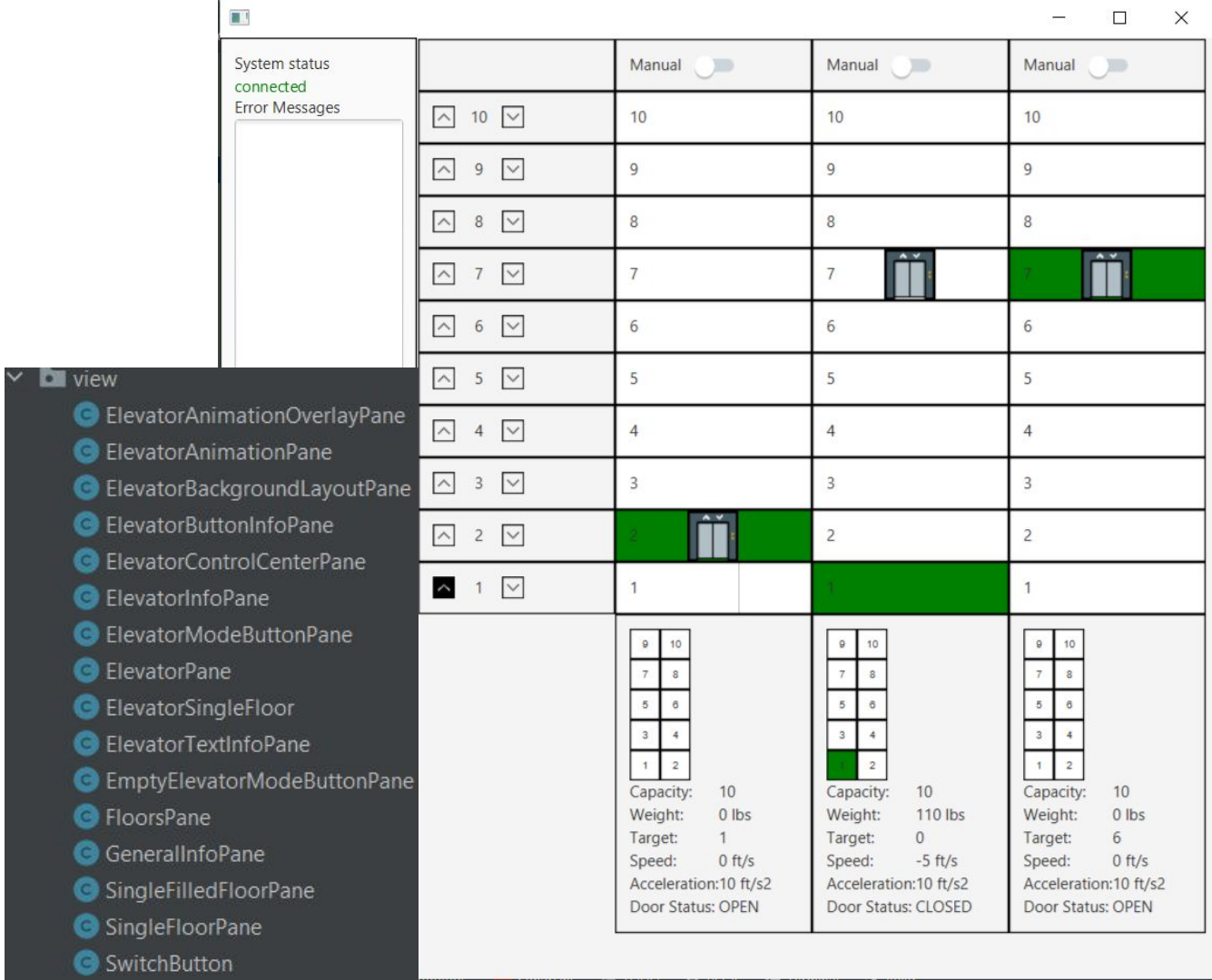
0.0% Duplications on 2k New Lines

GUI Mockup

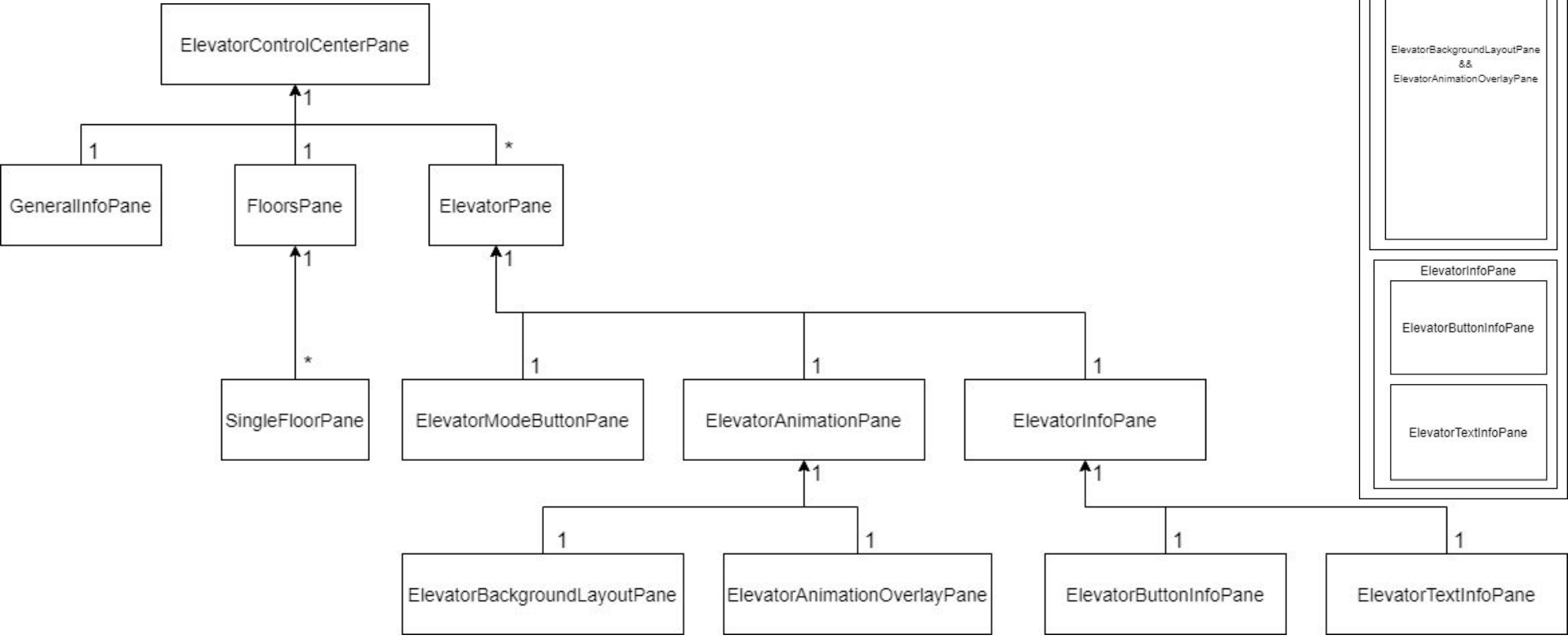

General Info
Floors
Elevator 1
System Status
All Systems are working
Log Messages:
9
8
7
6
5
4
3
2
1
Pressable in manual mode to set target
Capacity
Weight:
Target:
Speed:
Acceleration:

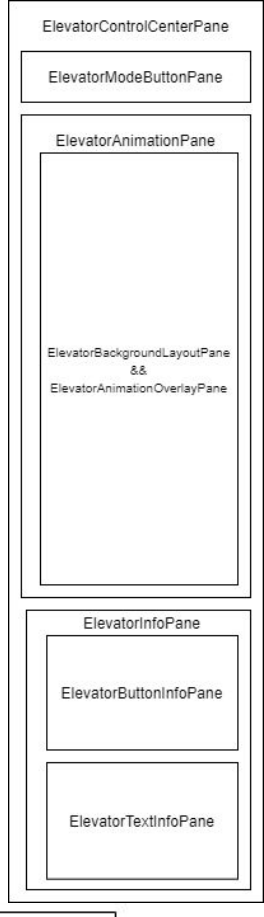
GUI Realization

- Modular
- No SceneBuilder
- Tested classes
- Show tests



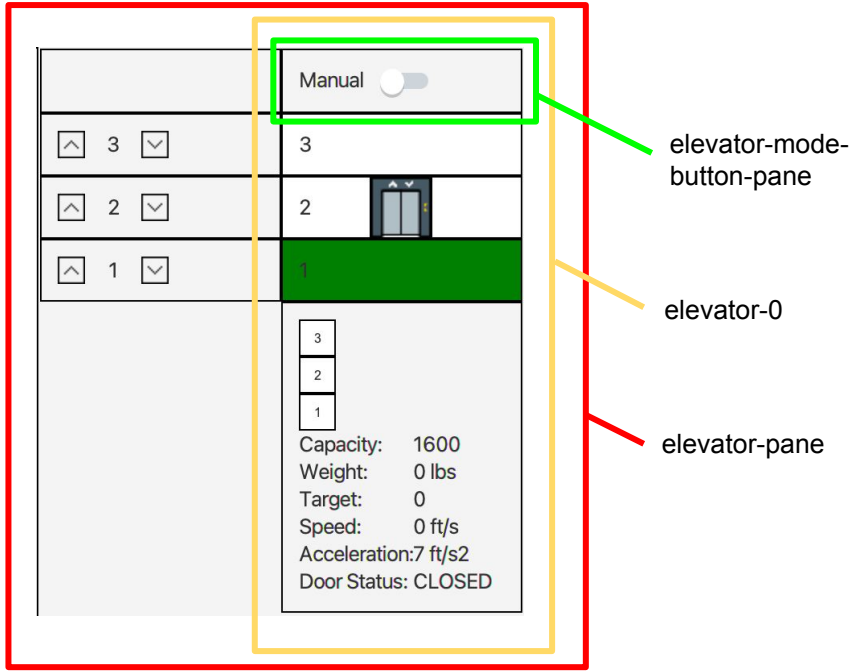
GUI Structure





GUI Tests

Testing API calls - enable manual mode



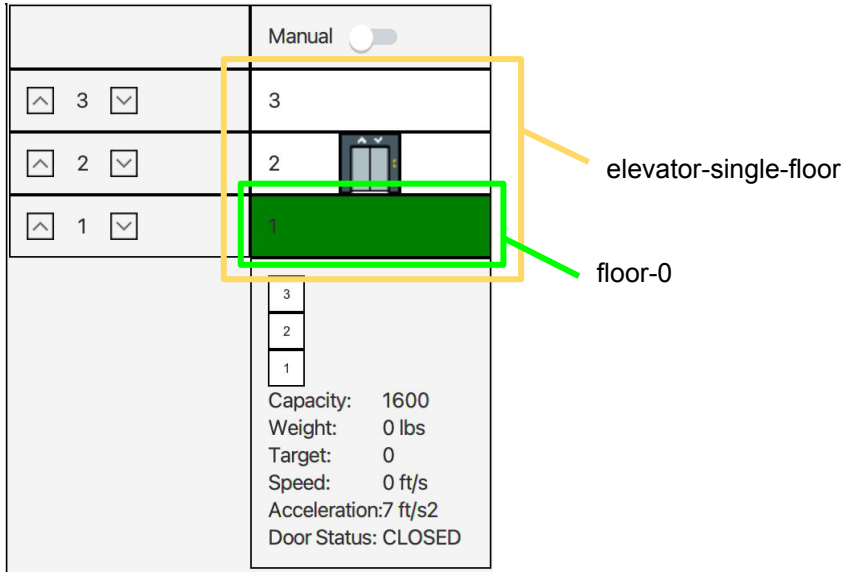

```
robot = new FxRobot();

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

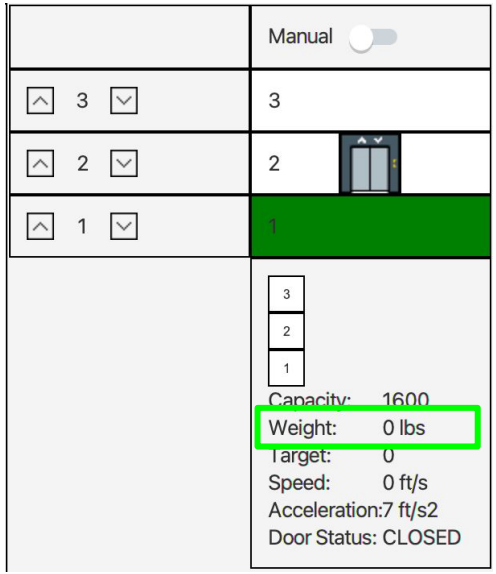



```
Node floor = robot.lookup(".elevator-single-floor")
  .lookup("#floor-0").query();

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

GUI Tests

Testing API calls - backend changes

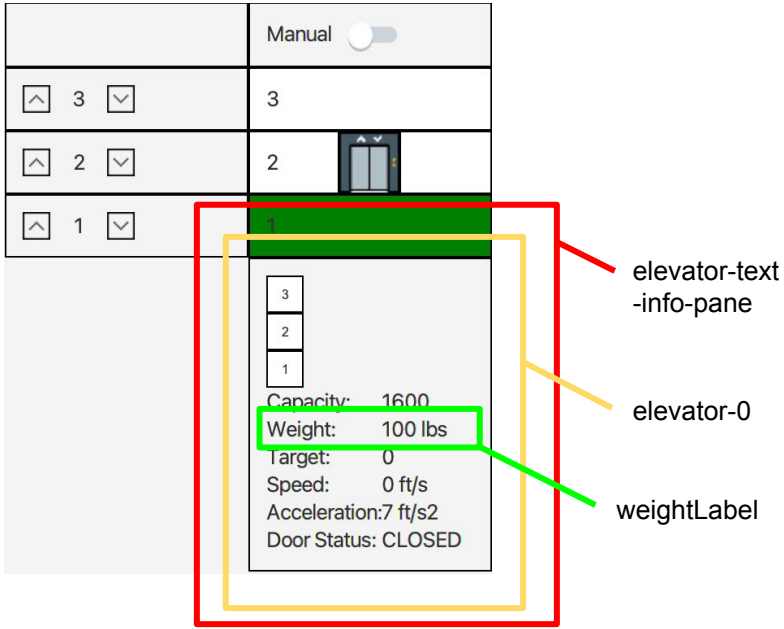



```
// initial mock return value
when(interfaceMock.getElevatorWeight(0)).thenReturn(0);

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

GUI Tests

Testing API calls - backend changes




```
// 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

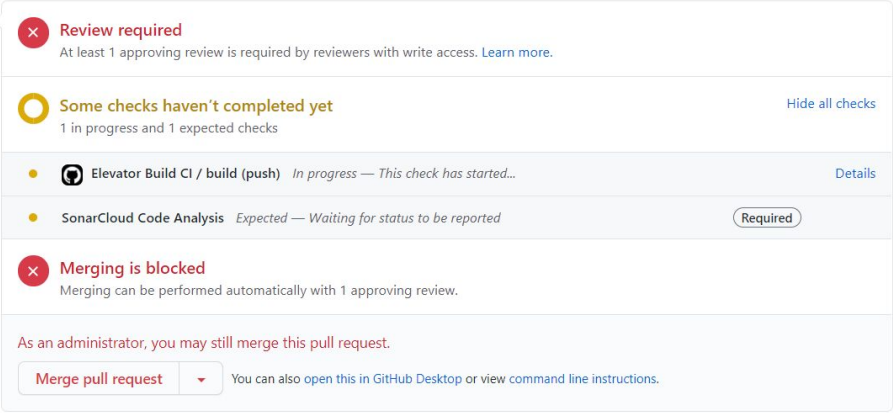
```
@Test
void testUpdateStateInitialElevatorButtonsOnly() {
    RMIElevatorAdapter rmiAdapterMock = mock(RMIElevatorAdapter.class);
    Floor floor01 = new Floor( number: 0,  buttonUp: false,  buttonDown: false);
    Floor floor02 = new Floor( number: 1,  buttonUp: false,  buttonDown: true);
    Floor 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,  commitedDirection: 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

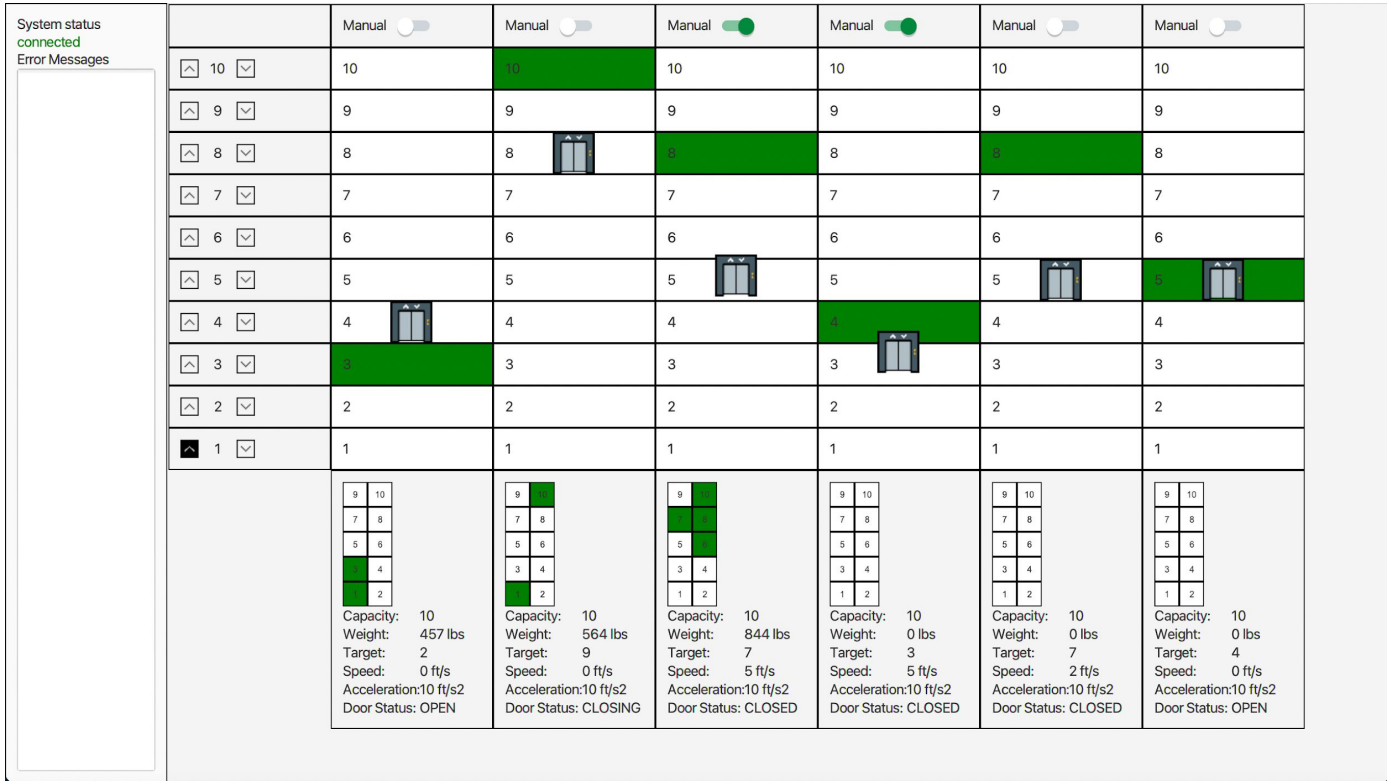
- Build Automation (GitHub Actions)
- Sonarcloud
  (code coverage, static analysis)
- Pull Requests
- Issue management &
  Communication
- Generally fix bugs when seeing them



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


System status
connected
Error Messages
Manual
Manual
Manual
Manual
Manual
Manual
10
9
8
7
6
5
4
3
2
1
Capacity: 10
Weight: 457 lbs
Target: 2
Speed: 0 ft/s
Acceleration:10 ft/s2
Door Status: OPEN
Capacity: 10
Weight: 564 lbs
Target: 9
Speed: 0 ft/s
Acceleration:10 ft/s2
Door Status: CLOSING
Capacity: 10
Weight: 844 lbs
Target: 7
Speed: 5 ft/s
Acceleration:10 ft/s2
Door Status: CLOSED
Capacity: 10
Weight: 0 lbs
Target: 3
Speed: 5 ft/s
Acceleration:10 ft/s2
Door Status: CLOSED
Capacity: 10
Weight: 0 lbs
Target: 7
Speed: 2 ft/s
Acceleration:10 ft/s2
Door Status: CLOSED
Capacity: 10
Weight: 0 lbs
Target: 4
Speed: 0 ft/s
Acceleration:10 ft/s2
Door Status: OPEN