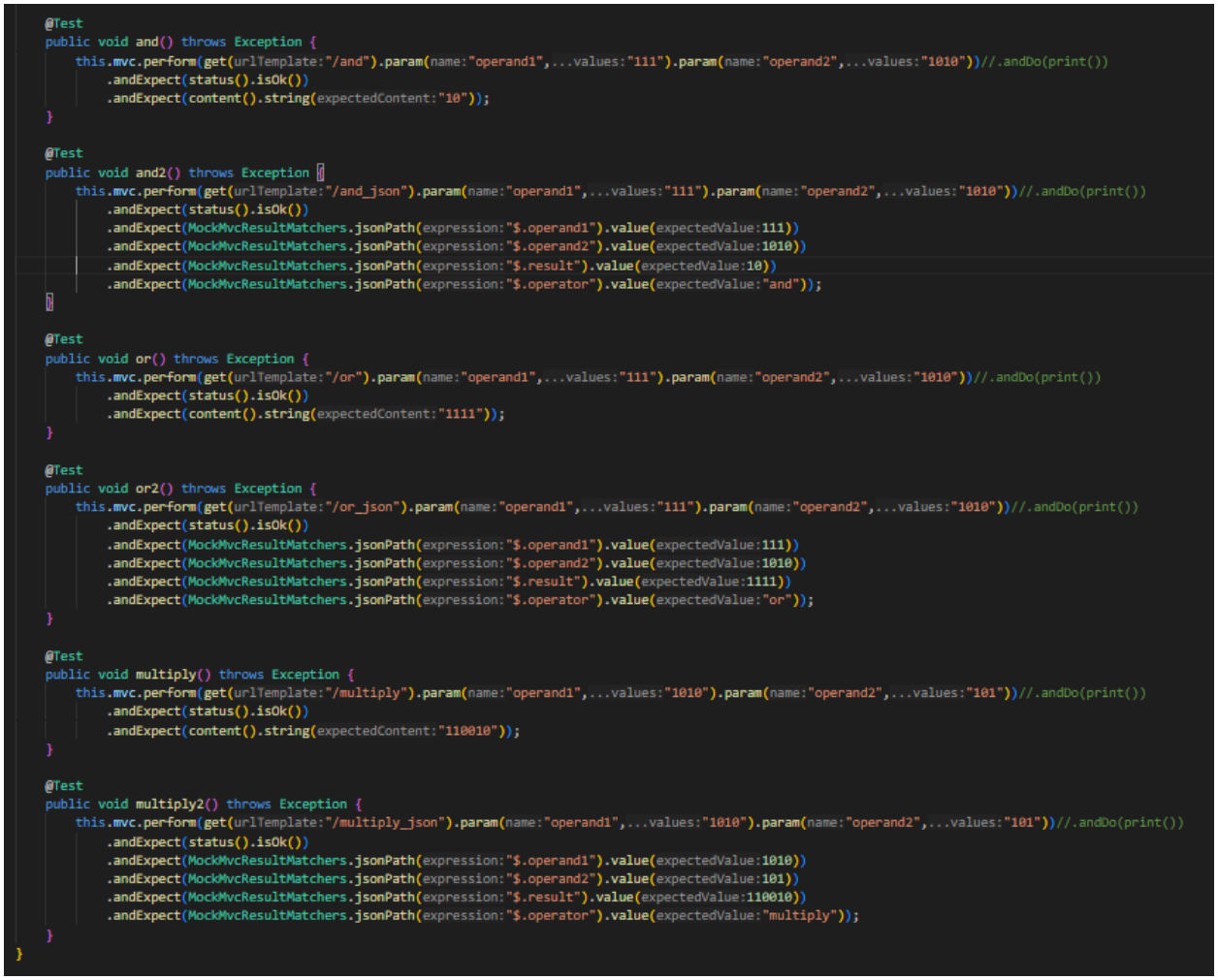
Lab 2 - Delivery
Software Quality

Okikioluwa Ojo (100790236)

Date: Feb 15, 2024

GitHub & Video Link: https://github.com/okikio-school/Soft-Quality-Lab2

BinaryAPIController, I set up the tests to send requests to the API Controller with the valid operands and an operator, and then assert the return from both the JSON result and the String result.



```
    @Test
    public void and() throws Exception {
        this.mvc.perform(get(urlTemplate:"/and").param(name:"operand1",...values:"111").param(name:"operand2",...values:"1010"))//.andDo(print())
            .andExpect(status().isOk())
            .andExpect(content().string(expectedContent:"10"));
    }

    @Test
    public void and2() throws Exception {
        this.mvc.perform(get(urlTemplate:"/and_json").param(name:"operand1",...values:"111").param(name:"operand2",...values:"1010"))//.andDo(print())
            .andExpect(status().isOk())
            .andExpect(MockMvcResultMatchers.jsonPath(expression:"$.operand1").value(expectedValue:111))
            .andExpect(MockMvcResultMatchers.jsonPath(expression:"$.operand2").value(expectedValue:1010))
            .andExpect(MockMvcResultMatchers.jsonPath(expression:"$.result").value(expectedValue:10))
            .andExpect(MockMvcResultMatchers.jsonPath(expression:"$.operator").value(expectedValue:"and"));
    }

    @Test
    public void or() throws Exception {
        this.mvc.perform(get(urlTemplate:"/or").param(name:"operand1",...values:"111").param(name:"operand2",...values:"1010"))//.andDo(print())
            .andExpect(status().isOk())
            .andExpect(content().string(expectedContent:"1111"));
    }

    @Test
    public void or2() throws Exception {
        this.mvc.perform(get(urlTemplate:"/or_json").param(name:"operand1",...values:"111").param(name:"operand2",...values:"1010"))//.andDo(print())
            .andExpect(status().isOk())
            .andExpect(MockMvcResultMatchers.jsonPath(expression:"$.operand1").value(expectedValue:111))
            .andExpect(MockMvcResultMatchers.jsonPath(expression:"$.operand2").value(expectedValue:1010))
            .andExpect(MockMvcResultMatchers.jsonPath(expression:"$.result").value(expectedValue:1111))
            .andExpect(MockMvcResultMatchers.jsonPath(expression:"$.operator").value(expectedValue:"or"));
    }

    @Test
    public void multiply() throws Exception {
        this.mvc.perform(get(urlTemplate:"/multiply").param(name:"operand1",...values:"1010").param(name:"operand2",...values:"101"))//.andDo(print())
            .andExpect(status().isOk())
            .andExpect(content().string(expectedContent:"110010"));
    }

    @Test
    public void multiply2() throws Exception {
        this.mvc.perform(get(urlTemplate:"/multiply_json").param(name:"operand1",...values:"1010").param(name:"operand2",...values:"101"))//.andDo(print())
            .andExpect(status().isOk())
            .andExpect(MockMvcResultMatchers.jsonPath(expression:"$.operand1").value(expectedValue:1010))
            .andExpect(MockMvcResultMatchers.jsonPath(expression:"$.operand2").value(expectedValue:101))
            .andExpect(MockMvcResultMatchers.jsonPath(expression:"$.result").value(expectedValue:110010))
            .andExpect(MockMvcResultMatchers.jsonPath(expression:"$.operator").value(expectedValue:"multiply"));
    }
}
```

BinaryController, let’s us run the same tests on the UI and model itself, letting us verify that the UI also functions correctly, for the new functionality added.

```java
    @Test
    public void andParameter() throws Exception {
        this.mvc.perform(post(urlTemplate:"/").param(name:"operand1",...values:"111").param(name:"operator",...values:"&").param(n…"operand2","111"))//.andDo(print())
            .andExpect(status().isOk())
            .andExpect(view().name(expectedViewName:"result"))
            .andExpect(model().attribute(name:"result", value:"111"))
            .andExpect(model().attribute(name:"operand1", value:"111"));
    }

    @Test
    public void orParameter() throws Exception {
        this.mvc.perform(post(urlTemplate:"/").param(name:"operand1",...values:"111").param(name:"operator",...values:"|").param(n…"operand2","111"))//.andDo(print())
            .andExpect(status().isOk())
            .andExpect(view().name(expectedViewName:"result"))
            .andExpect(model().attribute(name:"result", value:"111"))
            .andExpect(model().attribute(name:"operand1", value:"111"));
    }

    @Test
    public void multiplyParameter() throws Exception {
        this.mvc.perform(post(urlTemplate:"/").param(name:"operand1",...values:"1010").param(name:"operator",...values:"*").param(n…"operand2","101"))//.andDo(print())
            .andExpect(status().isOk())
            .andExpect(view().name(expectedViewName:"result"))
            .andExpect(model().attribute(name:"result", value:"110010"))
            .andExpect(model().attribute(name:"operand1", value:"1010"));
    }

}
```

Image of successful tests