



Lab 2

Faculty of Engineering and Applied Science

SOFE 3980U : Software Quality

Due: February 16th, 2024

Name: Daniel Amasowomwan

ID: 100787640

Email: daniel.amasowomwan@ontariotechu.net

Github Repository Link: <https://github.com/Daniel-Amas/SOFE3980U-Quality--Lab2.git>

Video Link: [yUw3muBG2q.mp4](#)

Tests Results

```
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 13, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] >>> spring-boot:2.1.2.RELEASE:run (default-cli) > test-compile @ BinaryCalculatorWebapp >>>
[INFO]
[INFO] --- resources:3.1.0:resources (default-resources) @ BinaryCalculatorWebapp ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 0 resource
[INFO] Copying 4 resources
[INFO]
[INFO] --- compiler:3.8.0:compile (default-compile) @ BinaryCalculatorWebapp ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- resources:3.1.0:testResources (default-testResources) @ BinaryCalculatorWebapp ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
```

BinaryApiController.java

```
src > main > java > com.brianandtech.soles9800 > BinaryApiController.java > BinaryApiController

30  @GetMapping("/multiply")
31  ✓ public String multiplyString(@RequestParam(name="operand1", required=false, defaultValue="") String operand1,
32     @RequestParam(name="operand2", required=false, defaultValue="") String operand2) {
33     Binary number1=new Binary (operand1);
34     Binary number2=new Binary (operand2);
35     return Binary.multiply(number1,number2).getValue();
36     // http://localhost:8080/add?operand1=111&operand2=1010
37 }
38
39  @GetMapping("/multiply_json")
40  ✓ public BinaryAPIResult multiplyJSON(@RequestParam(name="operand1", required=false, defaultValue="") String operand1,
41     @RequestParam(name="operand2", required=false, defaultValue="") String operand2) {
42     Binary number1=new Binary (operand1);
43     Binary number2=new Binary (operand2);
44     return new BinaryAPIResult(number1,"multiply",number2,Binary.multiply(number1,number2));
45     // http://localhost:8080/add?operand1=111&operand2=1010
46 }
47
48  @GetMapping("/or")
49  ✓ public String orString(@RequestParam(name="operand1", required=false, defaultValue="") String operand1,
50     @RequestParam(name="operand2", required=false, defaultValue="") String operand2) {
51     Binary number1=new Binary (operand1);
52     Binary number2=new Binary (operand2);
53     return Binary.or(number1,number2).getValue();
54     // http://localhost:8080/add?operand1=111&operand2=1010
55 }
56
57  @GetMapping("/or_json")
58  ✓ public BinaryAPIResult orJSON(@RequestParam(name="operand1", required=false, defaultValue="") String operand1,
59     @RequestParam(name="operand2", required=false, defaultValue="") String operand2) {
60     Binary number1=new Binary (operand1);
61     Binary number2=new Binary (operand2);
62     return new BinaryAPIResult(number1,"or",number2,Binary.or(number1,number2));
63     // http://localhost:8080/add?operand1=111&operand2=1010
64 }
65
66  @GetMapping("/and")
67  ✓ public String andString(@RequestParam(name="operand1", required=false, defaultValue="") String operand1,
68     @RequestParam(name="operand2", required=false, defaultValue="") String operand2) {
69     Binary number1=new Binary (operand1);
70     Binary number2=new Binary (operand2);
71     return Binary.and(number1,number2).getValue();
72     // http://localhost:8080/add?operand1=111&operand2=1010
73 }
74
```

```

65
66     @GetMapping("/and")
67     public String andString(@RequestParam(name="operand1", required=false, defaultValue="") String operand1,
68                             Binary number1 = BinaryAPIController.andString(String, String) lue="") String operand2
69         Binary number1=new Binary (operand1);
70         Binary number2=new Binary (operand2);
71         return Binary.and(number1,number2).getValue();
72         // http://localhost:8080/add?operand1=111&operand2=1010
73     }
74
75     @GetMapping("/and_json")
76     public BinaryAPIResult andJSON(@RequestParam(name="operand1", required=false, defaultValue="") String operand1,
77                                    @RequestParam(name="operand2", required=false, defaultValue="") String operand2)
78         Binary number1=new Binary (operand1);
79         Binary number2=new Binary (operand2);
80         return new BinaryAPIResult(number1,"and",number2,Binary.and(number1,number2));
81         // http://localhost:8080/add?operand1=111&operand2=1010
82     }
83
84 }

```

BinaryAPIControllerTest.java

```
> 30 public class BinaryAPIControllerTest {
31
32     @Autowired
33     private MockMvc mvc;
34
35
36     @Test
37     public void add() throws Exception {
38         this.mvc.perform(get("/add").param("operand1", "111").param("operand2", "1010"))//.andDo(print())
39             .andExpect(status().isOk())
40             .andExpect(content().string("10001"));
41     }
42
43     @Test
44     public void add2() throws Exception {
45         this.mvc.perform(get("/add_json").param("operand1", "111").param("operand2", "1010"))//.andDo(print())
46             .andExpect(status().isOk())
47             .andExpect(MockMvcResultMatchers.jsonPath("$.operand1").value(111))
48             .andExpect(MockMvcResultMatchers.jsonPath("$.operand2").value(1010))
49             .andExpect(MockMvcResultMatchers.jsonPath("$.result").value(10001))
50             .andExpect(MockMvcResultMatchers.jsonPath("$.operator").value("add"));
51     }
52
53     @Test
54     public void multiply() throws Exception {
55         this.mvc.perform(get("/multiply").param("operand1", "1010100").param("operand2", "100110"))//.andDo(print())
56             .andExpect(status().isOk())
57             .andExpect(content().string("110001111000"));
58     }
59
60     @Test
61     public void or() throws Exception {
62         this.mvc.perform(get("/or").param("operand1", "1010100").param("operand2", "100110"))//.andDo(print())
63             .andExpect(status().isOk())
64             .andExpect(content().string("1110110"));
65     }
66
67     @Test
68     public void and() throws Exception {
69         this.mvc.perform(get("/and").param("operand1", "1010100").param("operand2", "100110"))//.andDo(print())
70             .andExpect(status().isOk())
71             .andExpect(content().string("100"));
72     }
73 }
```

BinaryController.java

```
9  @Controller
10 public class BinaryController {
11
12     @GetMapping("/")
13     public String getCalculator(@RequestParam(name="operand1", required=false, defaultValue="") String operand1) {
14         model.addAttribute(attributeName:"operand1", operand1);
15         model.addAttribute(attributeName:"operand1Focused", operand1.length()>0);
16         return "calculator";
17     }
18
19     @PostMapping("/")
20     public String result(@RequestParam(name="operand1", required=false, defaultValue="") String operand1,
21                         @RequestParam(name="operator", required=false, defaultValue="") String operator,
22                         @RequestParam(name="operand2", required=false, defaultValue="") String operand2, Model model) {
23         model.addAttribute(attributeName:"operand1", operand1);
24         model.addAttribute(attributeName:"operator", operator);
25         model.addAttribute(attributeName:"operand2", operand2);
26         Binary number1=new Binary (operand1);
27         Binary number2=new Binary (operand2);
28         switch(operator)
29         {
30             case "+":
31                 model.addAttribute(attributeName:"result", Binary.add(number1,number2).getValue());
32                 return "result";
33             case "*":
34                 model.addAttribute(attributeName:"result", Binary.multiply(number1,number2).getValue());
35                 return "result";
36             case "|":
37                 model.addAttribute(attributeName:"result", Binary.or(number1,number2).getValue());
38                 return "result";
39             case "&":
40                 model.addAttribute(attributeName:"result", Binary.and(number1,number2).getValue());
41                 return "result";
42             default:
43                 return "Error";
44         }
45     }
46
47 }
```

BinaryControllerTest.java

```
35     public void getDefault() throws Exception {
36         this.mvc.perform(get("/"))//.andDo(print())
37             .andExpect(status().isOk())
38             .andExpect(view().name("calculator"))
39             .andExpect(model().attribute("operand1", ""))
40             .andExpect(model().attribute("operand1Focused", false));
41     }
42
43     @Test
44     public void getParameter() throws Exception {
45         this.mvc.perform(get("/").param("operand1","111"))
46             .andExpect(status().isOk())
47             .andExpect(view().name("calculator"))
48             .andExpect(model().attribute("operand1", "111"))
49             .andExpect(model().attribute("operand1Focused", true));
50     }
51     @Test
52     public void postParameter() throws Exception {
53         this.mvc.perform(post("/").param("operand1","111").param("operator","+").param("operand2","111"))
54             .andExpect(status().isOk())
55             .andExpect(view().name("result"))
56             .andExpect(model().attribute("result", "1110"))
57             .andExpect(model().attribute("operand1", "111"));
58     }
59     @Test
60     public void postTrailingZeros() throws Exception {
61         this.mvc.perform(post("/").param("operand1","000111").param("operator","+").param("operand2","111"))
62             .andExpect(status().isOk())
63             .andExpect(view().name("result"))
64             .andExpect(model().attribute("result", "1110"));
65     }
66     @Test
67     public void postLargeBinaryNumbers() throws Exception {
68         this.mvc.perform(post("/").param("operand1", "1111111111111111").param("operator", "|").param("operand2", "1111111111111111"))
69             .andExpect(status().isOk())
70             .andExpect(view().name("result"))
71             .andExpect(model().attributeExists("result"));
72     }
73     @Test
74     public void postBinaryMultiplication() throws Exception {
75         this.mvc.perform(post("/").param("operand1", "101").param("operator", "*").param("operand2", "111"))
76             .andExpect(status().isOk())
77             .andExpect(view().name("result"))
78             .andExpect(model().attributeExists("result"))
79             .andExpect(model().attribute("result", "1111"));
80     }
81
82 }
```

Binary.java