Lab 2

Faculty of Engineering and Applied Science

SOFE 3980U : Software Quality

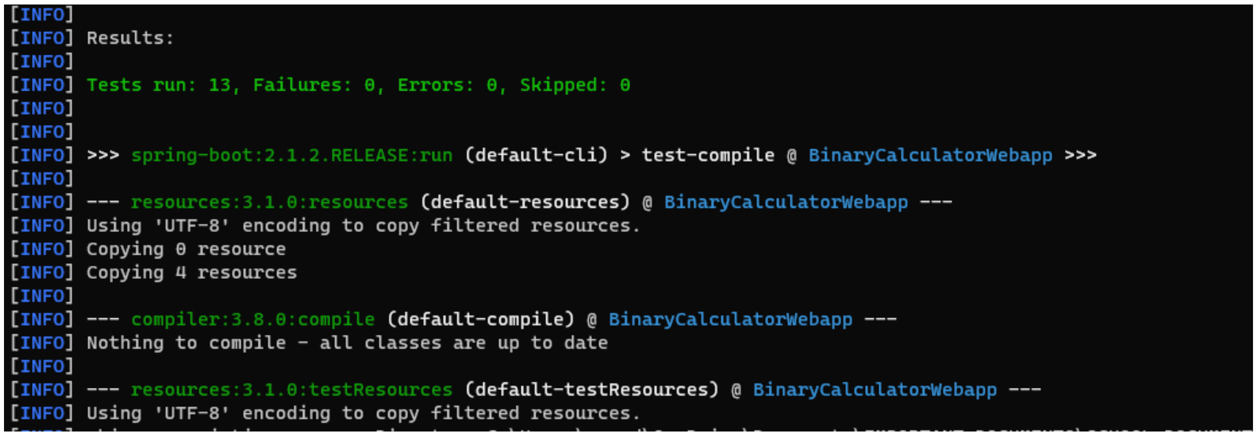
Due: February 16th, 2024

Name: Daniel Amasowomwan
ID: 100787640
Email: daniel.amasowomwan@ontariotechu.net

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

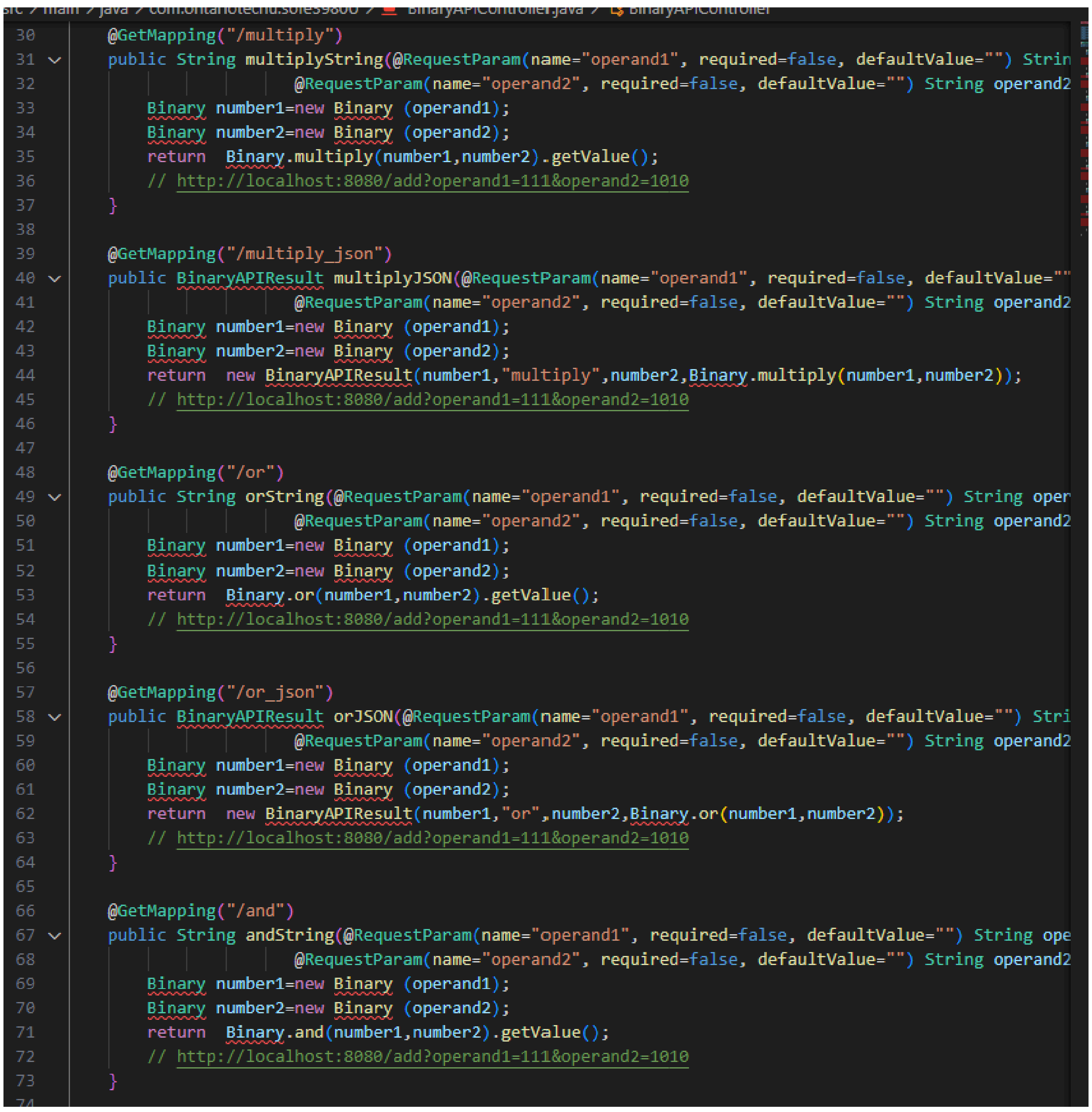
Video Link: yUw3muBG2q.mp4

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.ontariotechu.sofe3980U > BinaryAPIController.java > BinaryAPIController

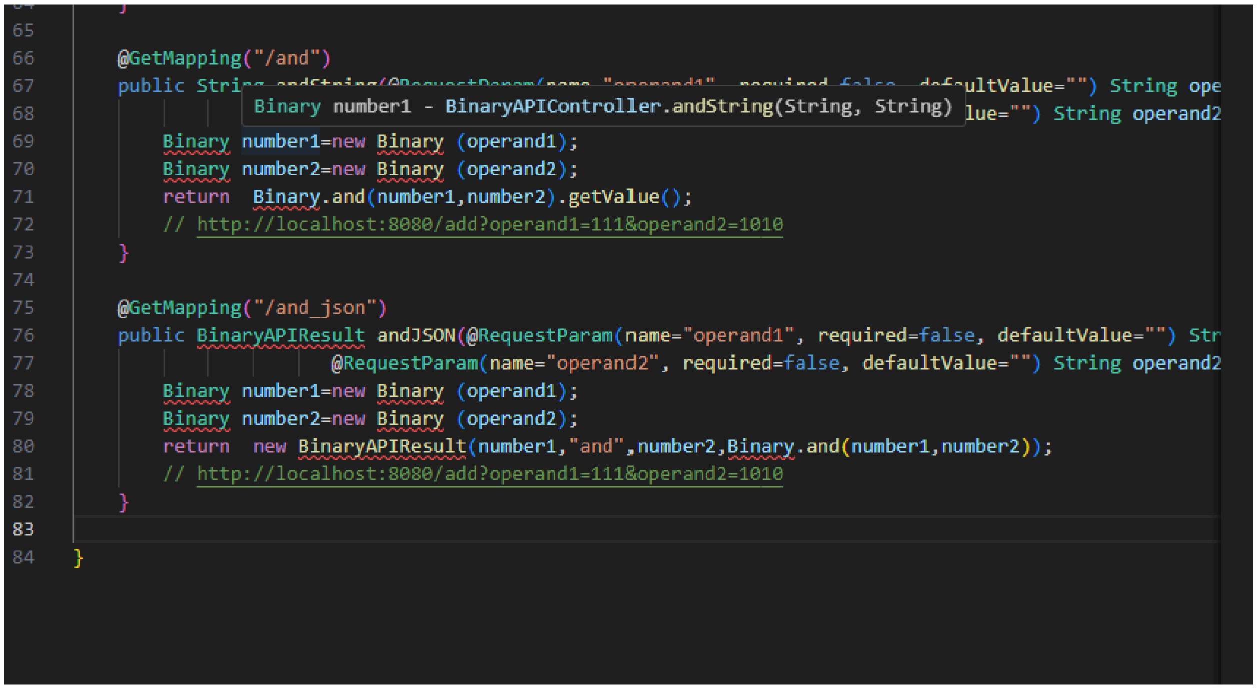
```java
    @GetMapping("/multiply")
    public String multiplyString(@RequestParam(name="operand1", required=false, defaultValue="") Strin
                    @RequestParam(name="operand2", required=false, defaultValue="") String operand2
        Binary number1=new Binary (operand1);
        Binary number2=new Binary (operand2);
        return  Binary.multiply(number1,number2).getValue();
        // http://localhost:8080/add?operand1=111&operand2=1010
    }

    @GetMapping("/multiply_json")
    public BinaryAPIResult multiplyJSON(@RequestParam(name="operand1", required=false, defaultValue=""
                    @RequestParam(name="operand2", required=false, defaultValue="") String operand2
        Binary number1=new Binary (operand1);
        Binary number2=new Binary (operand2);
        return  new BinaryAPIResult(number1,"multiply",number2,Binary.multiply(number1,number2));
        // http://localhost:8080/add?operand1=111&operand2=1010
    }

    @GetMapping("/or")
    public String orString(@RequestParam(name="operand1", required=false, defaultValue="") String oper
                    @RequestParam(name="operand2", required=false, defaultValue="") String operand2
        Binary number1=new Binary (operand1);
        Binary number2=new Binary (operand2);
        return  Binary.or(number1,number2).getValue();
        // http://localhost:8080/add?operand1=111&operand2=1010
    }

    @GetMapping("/or_json")
    public BinaryAPIResult orJSON(@RequestParam(name="operand1", required=false, defaultValue="") Stri
                    @RequestParam(name="operand2", required=false, defaultValue="") String operand2
        Binary number1=new Binary (operand1);
        Binary number2=new Binary (operand2);
        return  new BinaryAPIResult(number1,"or",number2,Binary.or(number1,number2));
        // http://localhost:8080/add?operand1=111&operand2=1010
    }

    @GetMapping("/and")
    public String andString(@RequestParam(name="operand1", required=false, defaultValue="") String ope
                    @RequestParam(name="operand2", required=false, defaultValue="") String operand2
        Binary number1=new Binary (operand1);
        Binary number2=new Binary (operand2);
        return  Binary.and(number1,number2).getValue();
        // http://localhost:8080/add?operand1=111&operand2=1010
    }
```



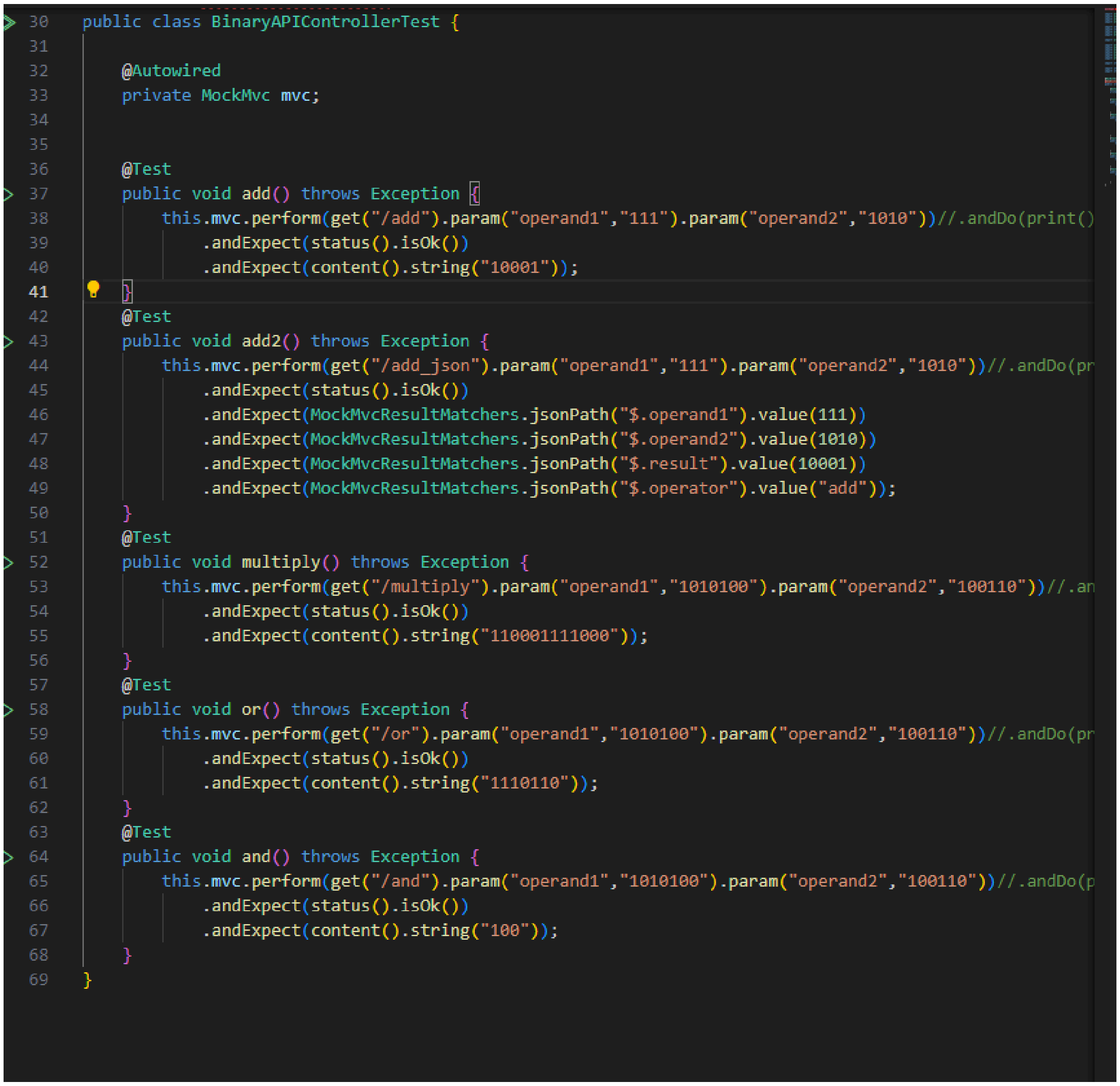
```
    }

    @GetMapping("/and")
    public String andString(@RequestParam(name="operand1", required=false, defaultValue="") String ope
            Binary number1 - BinaryAPIController.andString(String, String) lue="") String operand2
        Binary number1=new Binary (operand1);
        Binary number2=new Binary (operand2);
        return  Binary.and(number1,number2).getValue();
        // http://localhost:8080/add?operand1=111&operand2=1010
    }

    @GetMapping("/and_json")
    public BinaryAPIResult andJSON(@RequestParam(name="operand1", required=false, defaultValue="") Str
                    @RequestParam(name="operand2", required=false, defaultValue="") String operand2
        Binary number1=new Binary (operand1);
        Binary number2=new Binary (operand2);
        return  new BinaryAPIResult(number1,"and",number2,Binary.and(number1,number2));
        // http://localhost:8080/add?operand1=111&operand2=1010
    }

}
```

BinaryAPIControllerTest.java

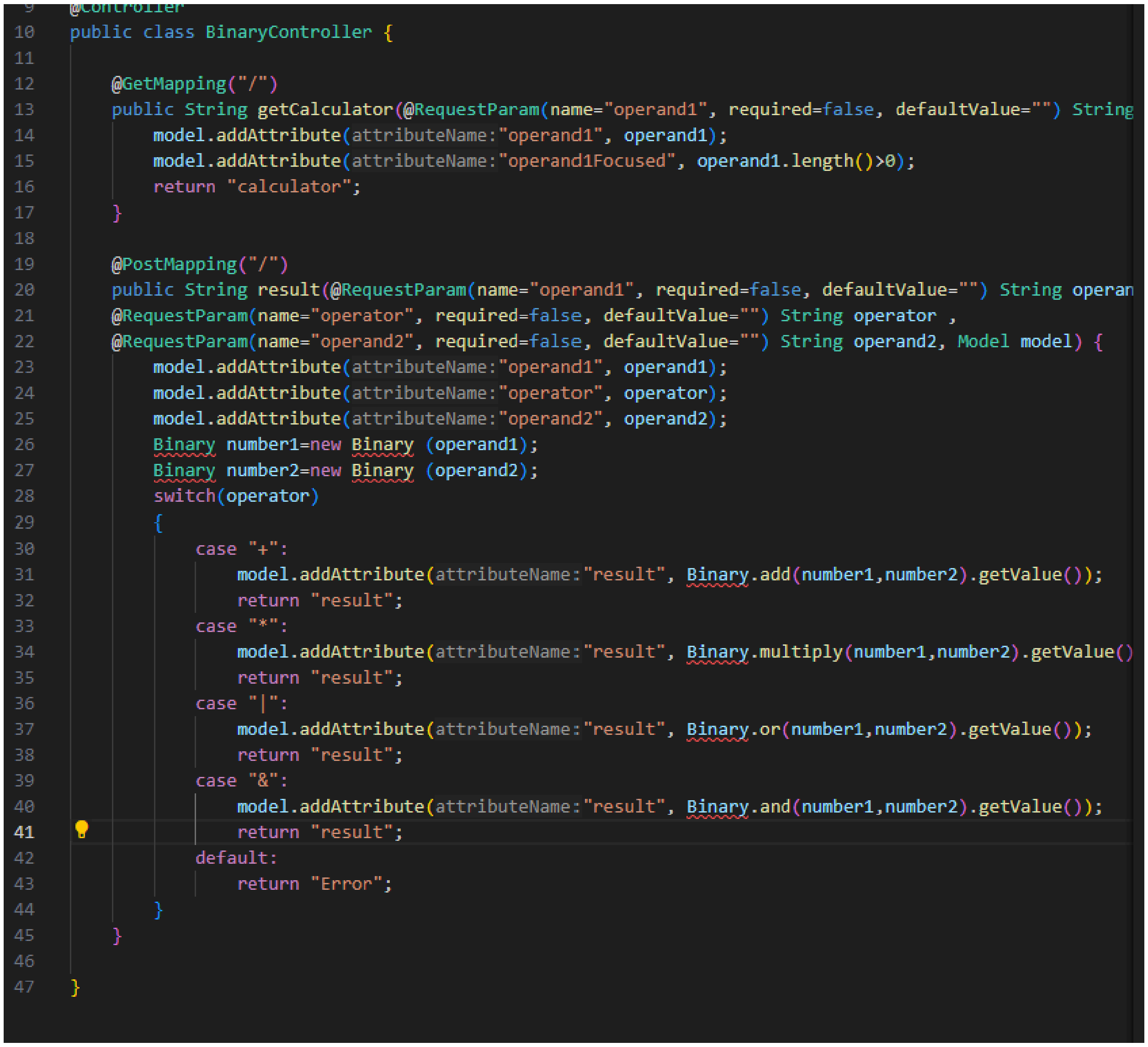


```java
public class BinaryAPIControllerTest {

    @Autowired
    private MockMvc mvc;


    @Test
    public void add() throws Exception {
        this.mvc.perform(get("/add").param("operand1","111").param("operand2","1010"))//.andDo(print()
            .andExpect(status().isOk())
            .andExpect(content().string("10001"));
    }
    @Test
    public void add2() throws Exception {
        this.mvc.perform(get("/add_json").param("operand1","111").param("operand2","1010"))//.andDo(pr
            .andExpect(status().isOk())
            .andExpect(MockMvcResultMatchers.jsonPath("$.operand1").value(111))
            .andExpect(MockMvcResultMatchers.jsonPath("$.operand2").value(1010))
            .andExpect(MockMvcResultMatchers.jsonPath("$.result").value(10001))
            .andExpect(MockMvcResultMatchers.jsonPath("$.operator").value("add"));
    }
    @Test
    public void multiply() throws Exception {
        this.mvc.perform(get("/multiply").param("operand1","1010100").param("operand2","100110"))//.an
            .andExpect(status().isOk())
            .andExpect(content().string("110001111000"));
    }
    @Test
    public void or() throws Exception {
        this.mvc.perform(get("/or").param("operand1","1010100").param("operand2","100110"))//.andDo(pr
            .andExpect(status().isOk())
            .andExpect(content().string("1110110"));
    }
    @Test
    public void and() throws Exception {
        this.mvc.perform(get("/and").param("operand1","1010100").param("operand2","100110"))//.andDo(p
            .andExpect(status().isOk())
            .andExpect(content().string("100"));
    }
}
```

BinaryController.java



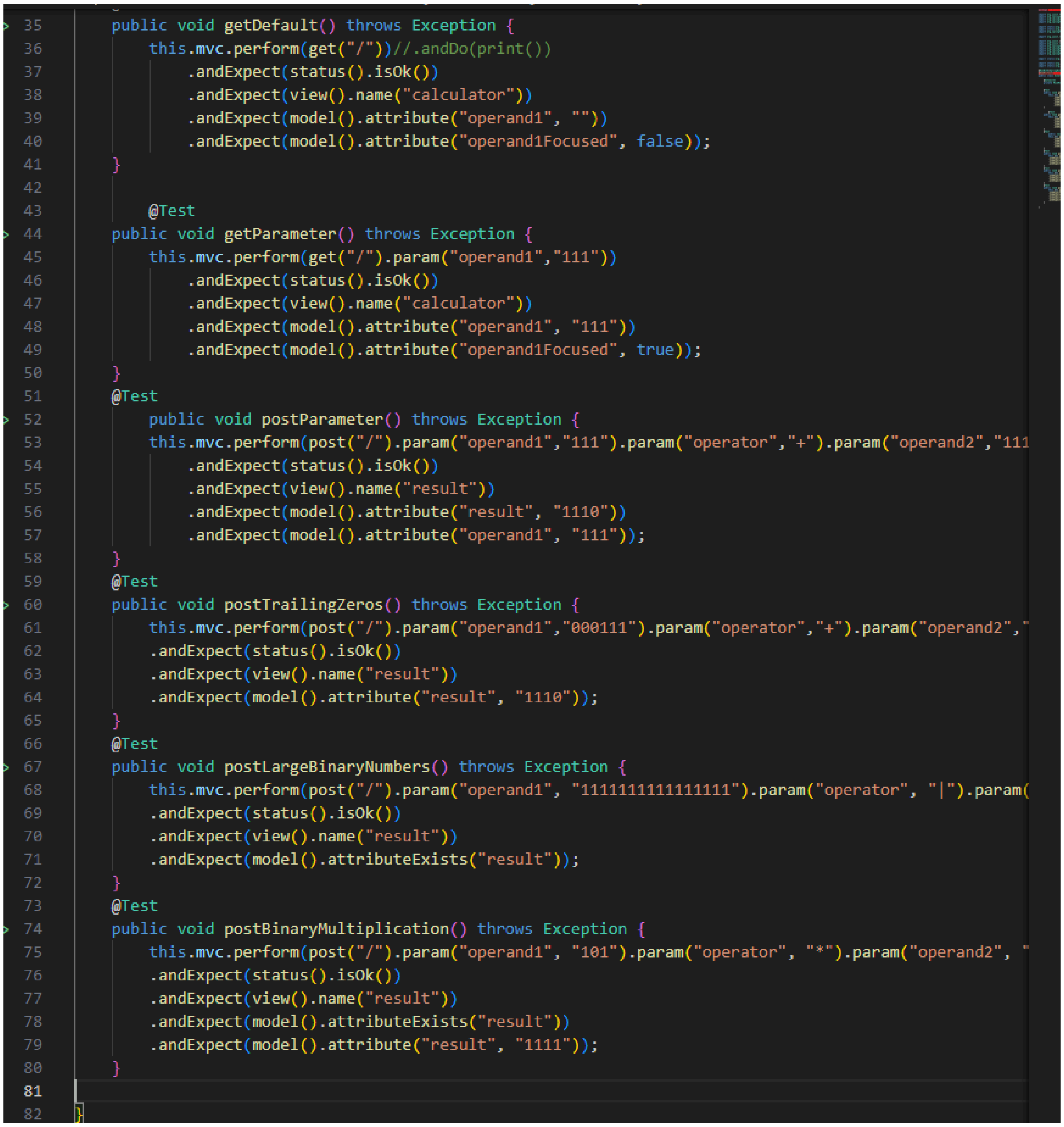
```
@Controller
public class BinaryController {

    @GetMapping("/")
    public String getCalculator(@RequestParam(name="operand1", required=false, defaultValue="") String
        model.addAttribute(attributeName:"operand1", operand1);
        model.addAttribute(attributeName:"operand1Focused", operand1.length()>0);
        return "calculator";
    }

    @PostMapping("/")
    public String result(@RequestParam(name="operand1", required=false, defaultValue="") String operan
    @RequestParam(name="operator", required=false, defaultValue="") String operator ,
    @RequestParam(name="operand2", required=false, defaultValue="") String operand2, Model model) {
        model.addAttribute(attributeName:"operand1", operand1);
        model.addAttribute(attributeName:"operator", operator);
        model.addAttribute(attributeName:"operand2", operand2);
        Binary number1=new Binary (operand1);
        Binary number2=new Binary (operand2);
        switch(operator)
        {
            case "+":
                model.addAttribute(attributeName:"result", Binary.add(number1,number2).getValue());
                return "result";
            case "*":
                model.addAttribute(attributeName:"result", Binary.multiply(number1,number2).getValue()
                return "result";
            case "|":
                model.addAttribute(attributeName:"result", Binary.or(number1,number2).getValue());
                return "result";
            case "&":
                model.addAttribute(attributeName:"result", Binary.and(number1,number2).getValue());
                return "result";
            default:
                return "Error";
        }
    }

}
```

BinaryControllerTest.java



```
    public void getDefault() throws Exception {
        this.mvc.perform(get("/"))//.andDo(print())
            .andExpect(status().isOk())
            .andExpect(view().name("calculator"))
            .andExpect(model().attribute("operand1", ""))
            .andExpect(model().attribute("operand1Focused", false));
    }

        @Test
    public void getParameter() throws Exception {
        this.mvc.perform(get("/").param("operand1","111"))
            .andExpect(status().isOk())
            .andExpect(view().name("calculator"))
            .andExpect(model().attribute("operand1", "111"))
            .andExpect(model().attribute("operand1Focused", true));
    }
    @Test
        public void postParameter() throws Exception {
        this.mvc.perform(post("/").param("operand1","111").param("operator","+").param("operand2","111
            .andExpect(status().isOk())
            .andExpect(view().name("result"))
            .andExpect(model().attribute("result", "1110"))
            .andExpect(model().attribute("operand1", "111"));
    }
    @Test
    public void postTrailingZeros() throws Exception {
        this.mvc.perform(post("/").param("operand1","000111").param("operator","+").param("operand2","
        .andExpect(status().isOk())
        .andExpect(view().name("result"))
        .andExpect(model().attribute("result", "1110"));
    }
    @Test
    public void postLargeBinaryNumbers() throws Exception {
        this.mvc.perform(post("/").param("operand1", "1111111111111111").param("operator", "|").param(
        .andExpect(status().isOk())
        .andExpect(view().name("result"))
        .andExpect(model().attributeExists("result"));
    }
    @Test
    public void postBinaryMultiplication() throws Exception {
        this.mvc.perform(post("/").param("operand1", "101").param("operator", "*").param("operand2", "
        .andExpect(status().isOk())
        .andExpect(view().name("result"))
        .andExpect(model().attributeExists("result"))
        .andExpect(model().attribute("result", "1111"));
    }

}
```

Binary.java