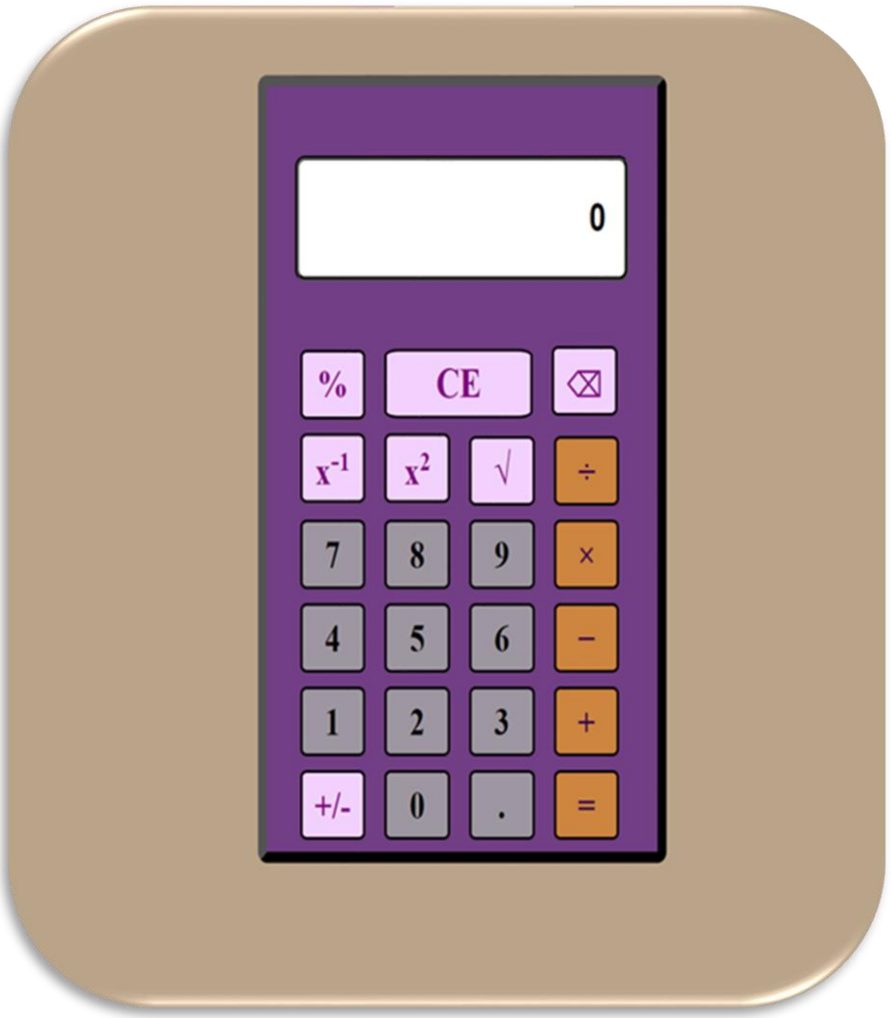


Assignment (2)

Calculator

- **NAME:**

Ranime Ahmed Elsayed Shehata.

- **ID:**

21010531

✦ **Problem Statement:**

Build a web-based calculator similar to that of windows.

- The buttons should be web buttons.
- No need for fancy styling.
- Calculation should be done on the server side.
- For simplicity, you can ignore the difference between the C and CE buttons.
- Repeating pressing the = button does not issue new calculations Handle exceptions such as dividing by 0, by displaying an E.

✦ **Notes:**

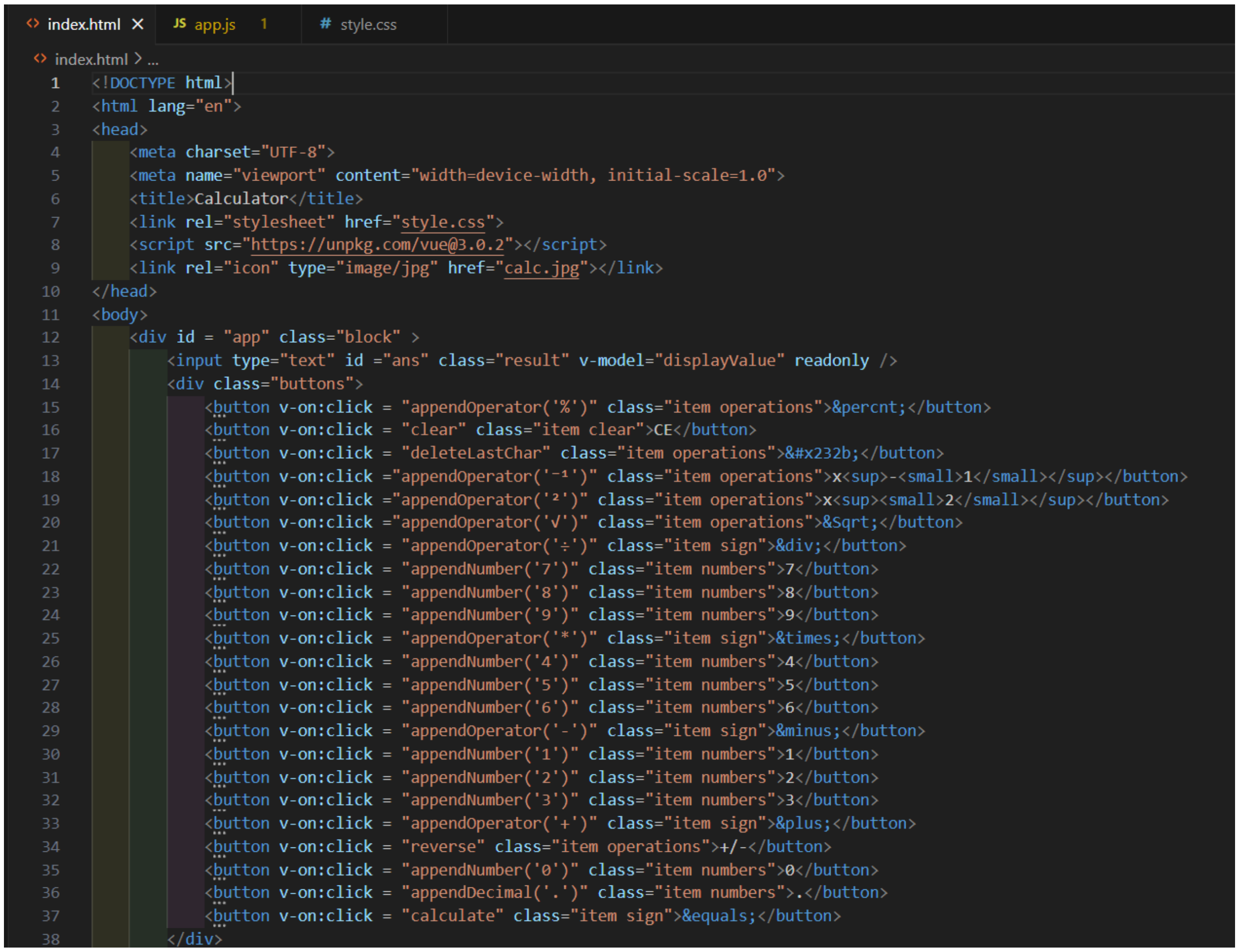
The calculator is implemented using **Vue.js** framework for frontend and all calculation are done on the server side using **Spring-Boot** framework.

- The buttons of the calculator are web buttons.

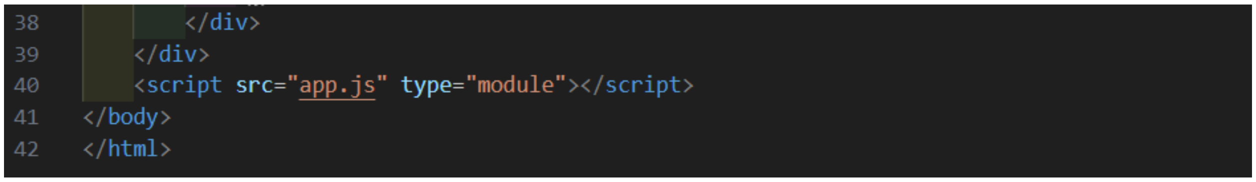
✦ **Source codes:**

They are attached in the same file. However, here are snapshots of them:

⇒**.html file:**

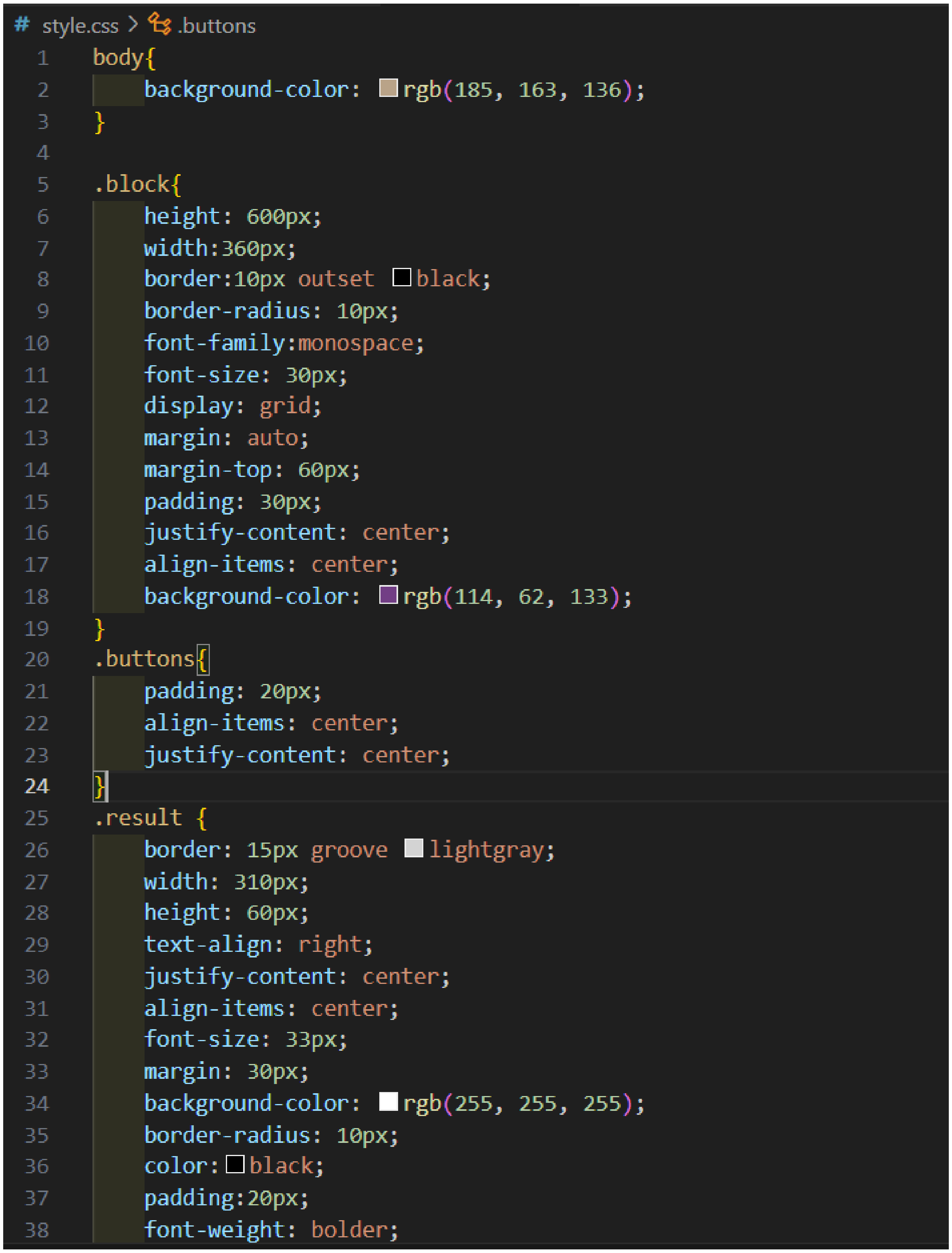


```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Calculator</title>
    <link rel="stylesheet" href="style.css">
    <script src="https://unpkg.com/vue@3.0.2"></script>
    <link rel="icon" type="image/jpg" href="calc.jpg"></link>
</head>
<body>
    <div id = "app" class="block" >
        <input type="text" id ="ans" class="result" v-model="displayValue" readonly />
        <div class="buttons">
            <button v-on:click = "appendOperator('%')" class="item operations">&percnt;</button>
            <button v-on:click = "clear" class="item clear">CE</button>
            <button v-on:click = "deleteLastChar" class="item operations">&#x232b;</button>
            <button v-on:click ="appendOperator('⁻¹')" class="item operations">x<sup>-<small>1</small></sup></button>
            <button v-on:click ="appendOperator('²')" class="item operations">x<sup><small>2</small></sup></button>
            <button v-on:click ="appendOperator('√')" class="item operations">&Sqrt;</button>
            <button v-on:click = "appendOperator('÷')" class="item sign">&div;</button>
            <button v-on:click = "appendNumber('7')" class="item numbers">7</button>
            <button v-on:click = "appendNumber('8')" class="item numbers">8</button>
            <button v-on:click = "appendNumber('9')" class="item numbers">9</button>
            <button v-on:click = "appendOperator('*')" class="item sign">&times;</button>
            <button v-on:click = "appendNumber('4')" class="item numbers">4</button>
            <button v-on:click = "appendNumber('5')" class="item numbers">5</button>
            <button v-on:click = "appendNumber('6')" class="item numbers">6</button>
            <button v-on:click = "appendOperator('-')" class="item sign">&minus;</button>
            <button v-on:click = "appendNumber('1')" class="item numbers">1</button>
            <button v-on:click = "appendNumber('2')" class="item numbers">2</button>
            <button v-on:click = "appendNumber('3')" class="item numbers">3</button>
            <button v-on:click = "appendOperator('+')" class="item sign">&plus;</button>
            <button v-on:click = "reverse" class="item operations">+/-</button>
            <button v-on:click = "appendNumber('0')" class="item numbers">0</button>
            <button v-on:click = "appendDecimal('.')" class="item numbers">.</button>
            <button v-on:click = "calculate" class="item sign">&equals;</button>
        </div>
```

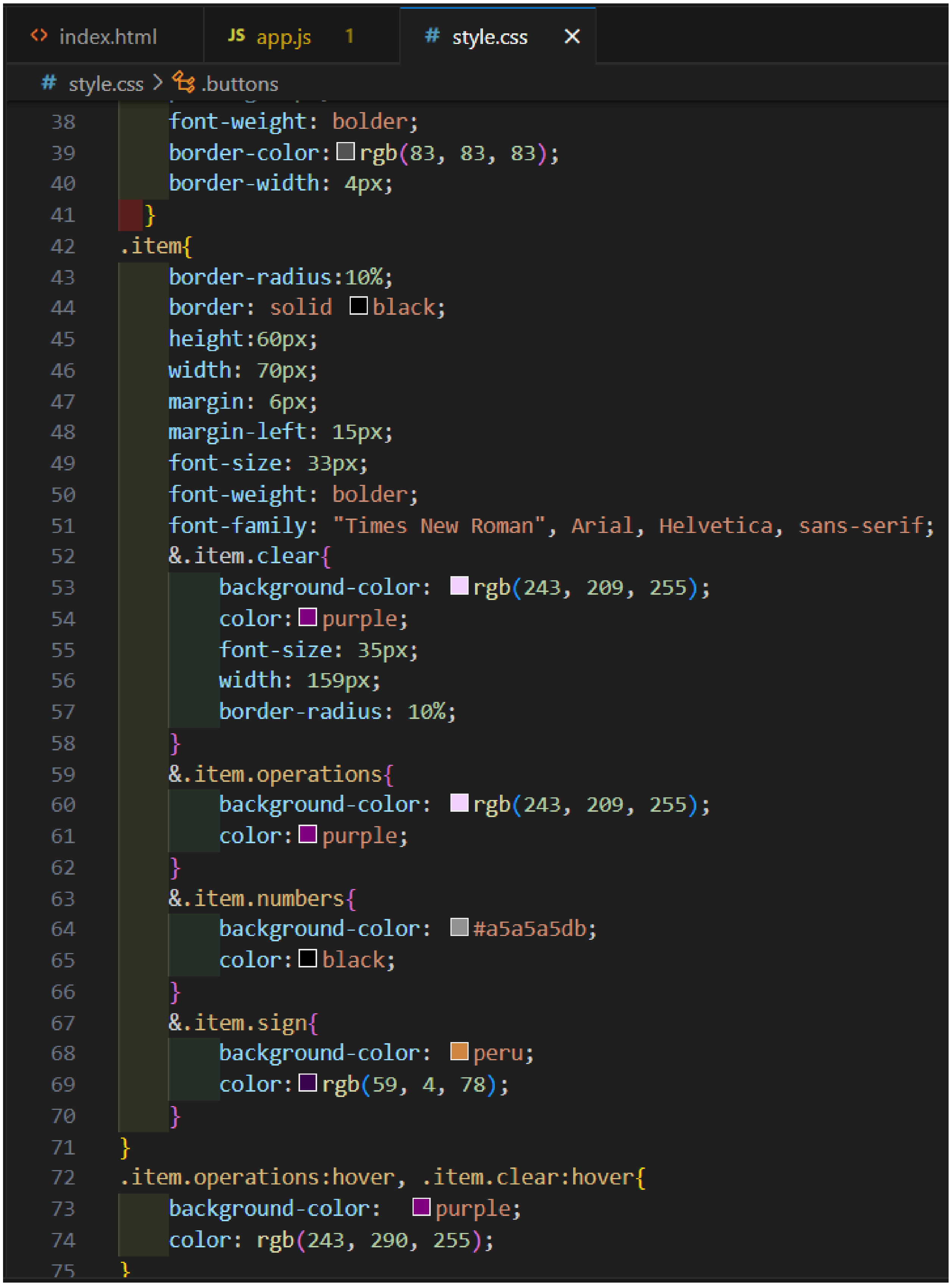


```html
        </div>
    </div>
    <script src="app.js" type="module"></script>
</body>
</html>
```

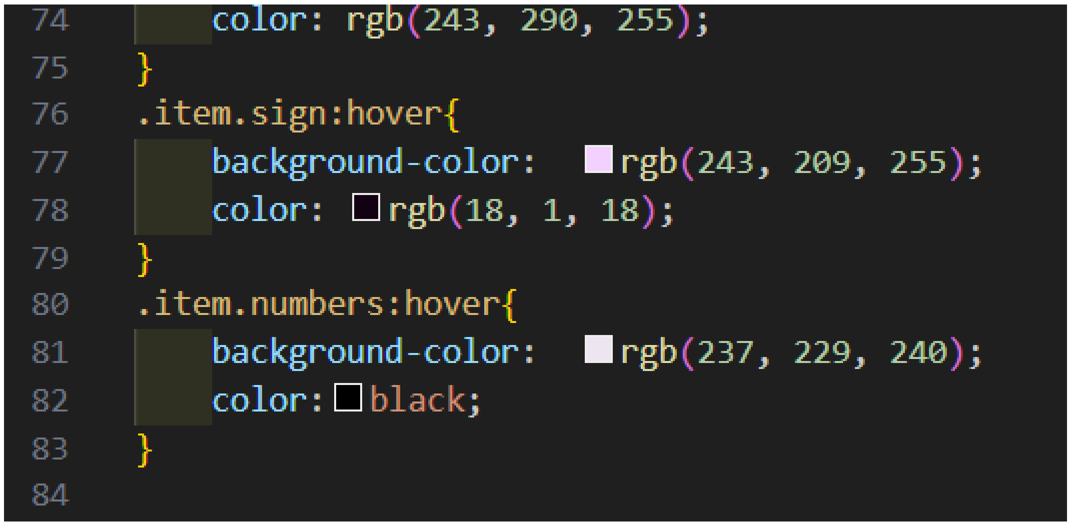
⇒**.css file:**



```
# style.css > .buttons
body{
    background-color: rgb(185, 163, 136);
}

.block{
    height: 600px;
    width:360px;
    border:10px outset black;
    border-radius: 10px;
    font-family:monospace;
    font-size: 30px;
    display: grid;
    margin: auto;
    margin-top: 60px;
    padding: 30px;
    justify-content: center;
    align-items: center;
    background-color: rgb(114, 62, 133);
}
.buttons{
    padding: 20px;
    align-items: center;
    justify-content: center;
}
.result {
    border: 15px groove lightgray;
    width: 310px;
    height: 60px;
    text-align: right;
    justify-content: center;
    align-items: center;
    font-size: 33px;
    margin: 30px;
    background-color: rgb(255, 255, 255);
    border-radius: 10px;
    color: black;
    padding:20px;
    font-weight: bolder;
```

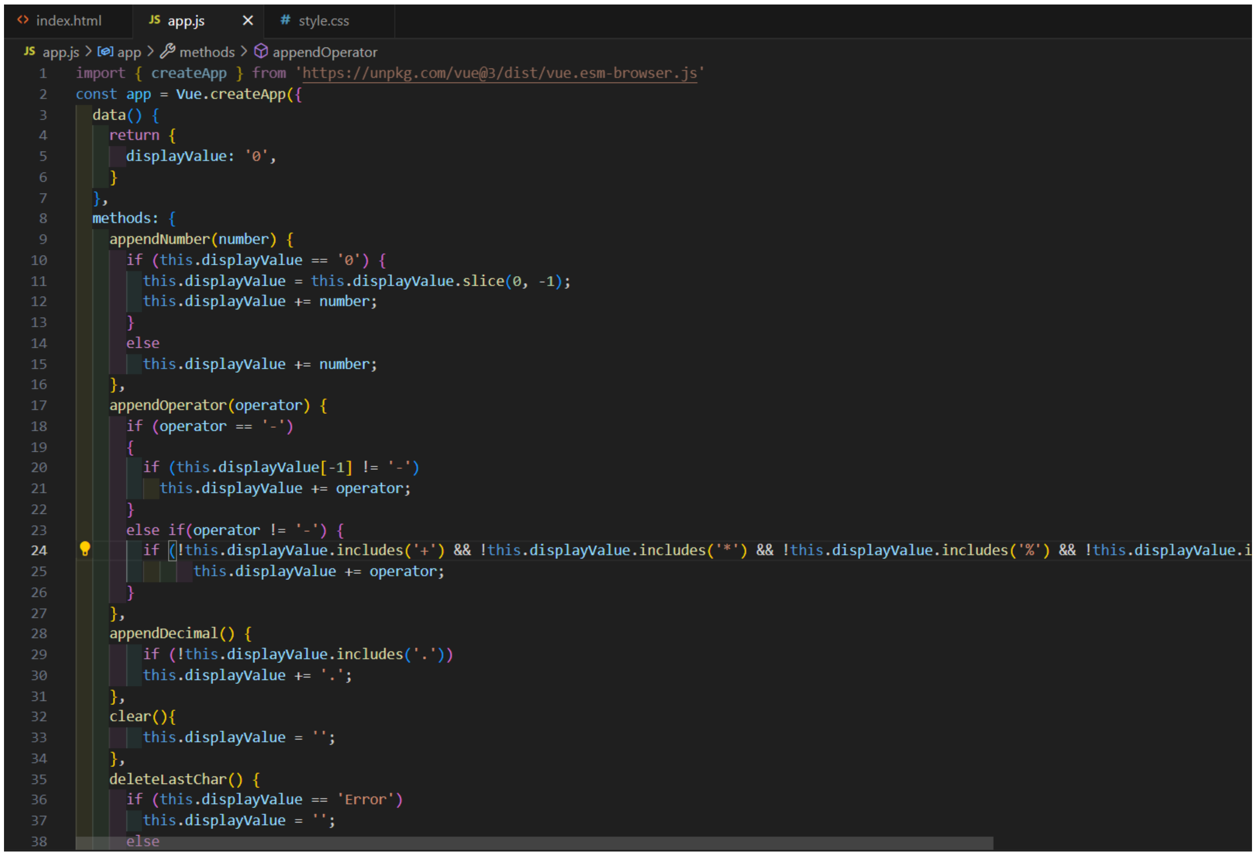


```
    font-weight: bolder;
    border-color: rgb(83, 83, 83);
    border-width: 4px;
}
.item{
    border-radius:10%;
    border: solid black;
    height:60px;
    width: 70px;
    margin: 6px;
    margin-left: 15px;
    font-size: 33px;
    font-weight: bolder;
    font-family: "Times New Roman", Arial, Helvetica, sans-serif;
    &.item.clear{
        background-color: rgb(243, 209, 255);
        color: purple;
        font-size: 35px;
        width: 159px;
        border-radius: 10%;
    }
    &.item.operations{
        background-color: rgb(243, 209, 255);
        color: purple;
    }
    &.item.numbers{
        background-color: #a5a5a5db;
        color: black;
    }
    &.item.sign{
        background-color: peru;
        color: rgb(59, 4, 78);
    }
}
.item.operations:hover, .item.clear:hover{
    background-color: purple;
    color: rgb(243, 290, 255);
}
```

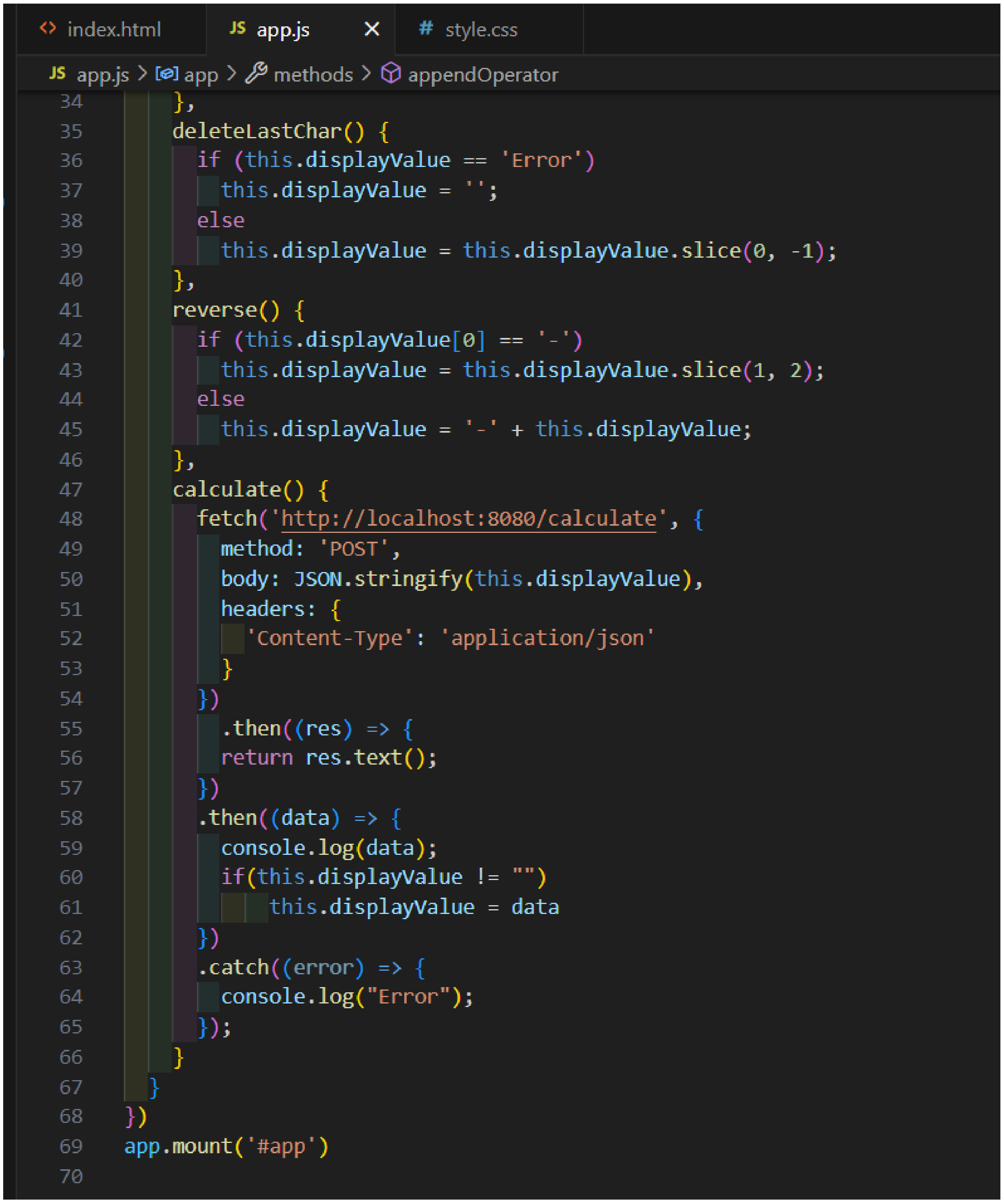


```css
    color: rgb(243, 290, 255);
}
.item.sign:hover{
    background-color:  rgb(243, 209, 255);
    color: rgb(18, 1, 18);
}
.item.numbers:hover{
    background-color:  rgb(237, 229, 240);
    color: black;
}

```

⇒**.js file:**

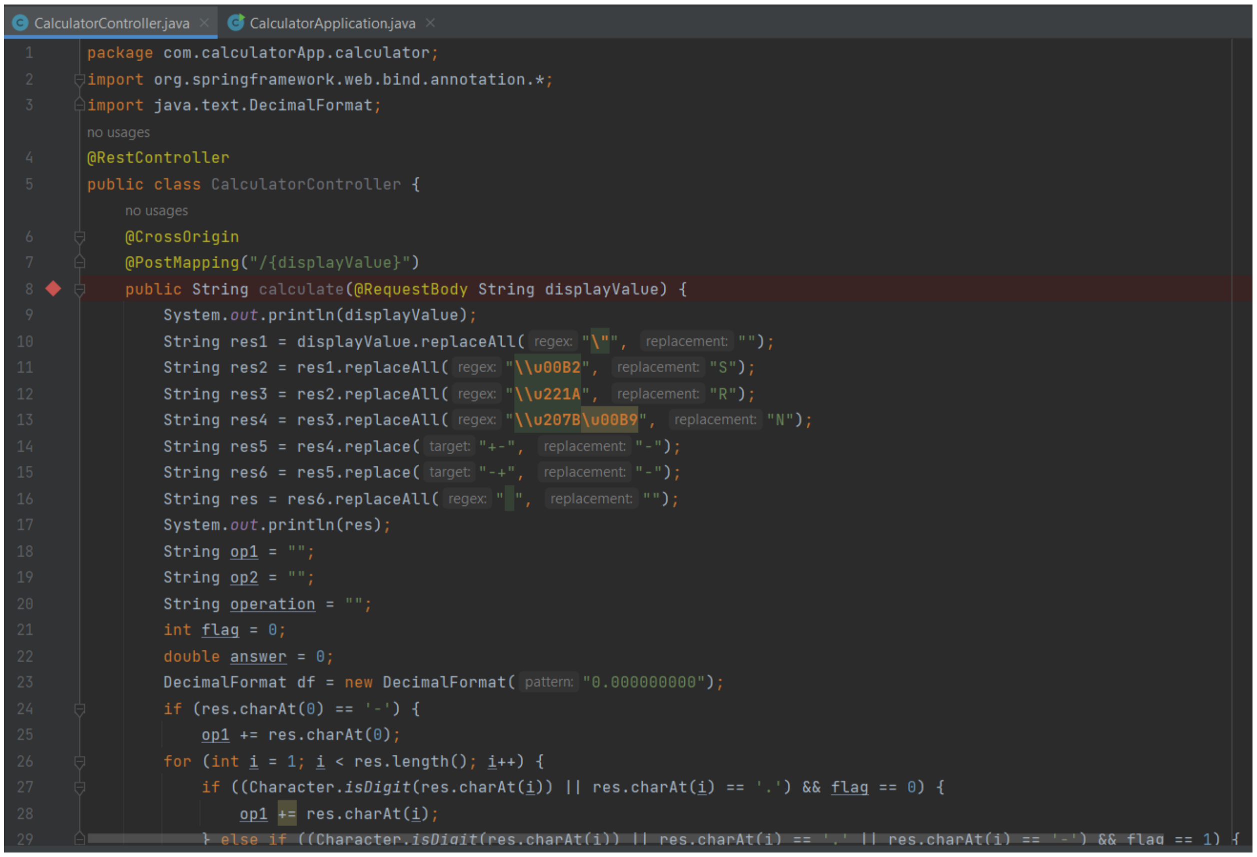


```javascript
import { createApp } from 'https://unpkg.com/vue@3/dist/vue.esm-browser.js'
const app = Vue.createApp({
  data() {
    return {
      displayValue: '0',
    }
  },
  methods: {
    appendNumber(number) {
      if (this.displayValue == '0') {
        this.displayValue = this.displayValue.slice(0, -1);
        this.displayValue += number;
      }
      else
        this.displayValue += number;
    },
    appendOperator(operator) {
      if (operator == '-')
      {
        if (this.displayValue[-1] != '-')
          this.displayValue += operator;
      }
      else if(operator != '-') {
        if (!this.displayValue.includes('+') && !this.displayValue.includes('*') && !this.displayValue.includes('%') && !this.displayValue.i
              this.displayValue += operator;
      }
    },
    appendDecimal() {
        if (!this.displayValue.includes('.'))
        this.displayValue += '.';
    },
    clear(){
        this.displayValue = '';
    },
    deleteLastChar() {
      if (this.displayValue == 'Error')
        this.displayValue = '';
      else
```

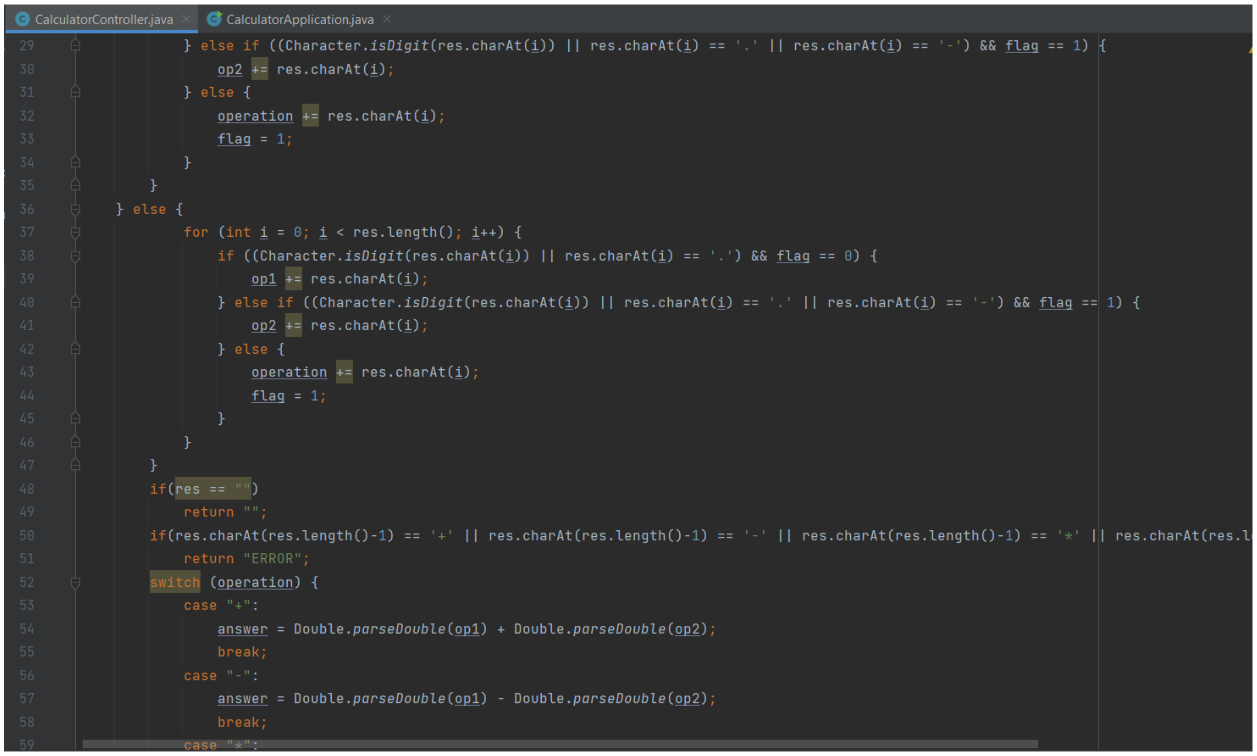


```
    },
    deleteLastChar() {
      if (this.displayValue == 'Error')
        this.displayValue = '';
      else
        this.displayValue = this.displayValue.slice(0, -1);
    },
    reverse() {
      if (this.displayValue[0] == '-')
        this.displayValue = this.displayValue.slice(1, 2);
      else
        this.displayValue = '-' + this.displayValue;
    },
    calculate() {
      fetch('http://localhost:8080/calculate', {
        method: 'POST',
        body: JSON.stringify(this.displayValue),
        headers: {
          'Content-Type': 'application/json'
        }
      })
        .then((res) => {
        return res.text();
      })
      .then((data) => {
        console.log(data);
        if(this.displayValue != "")
          this.displayValue = data
      })
      .catch((error) => {
        console.log("Error");
      });
    }
  }
})
app.mount('#app')
```

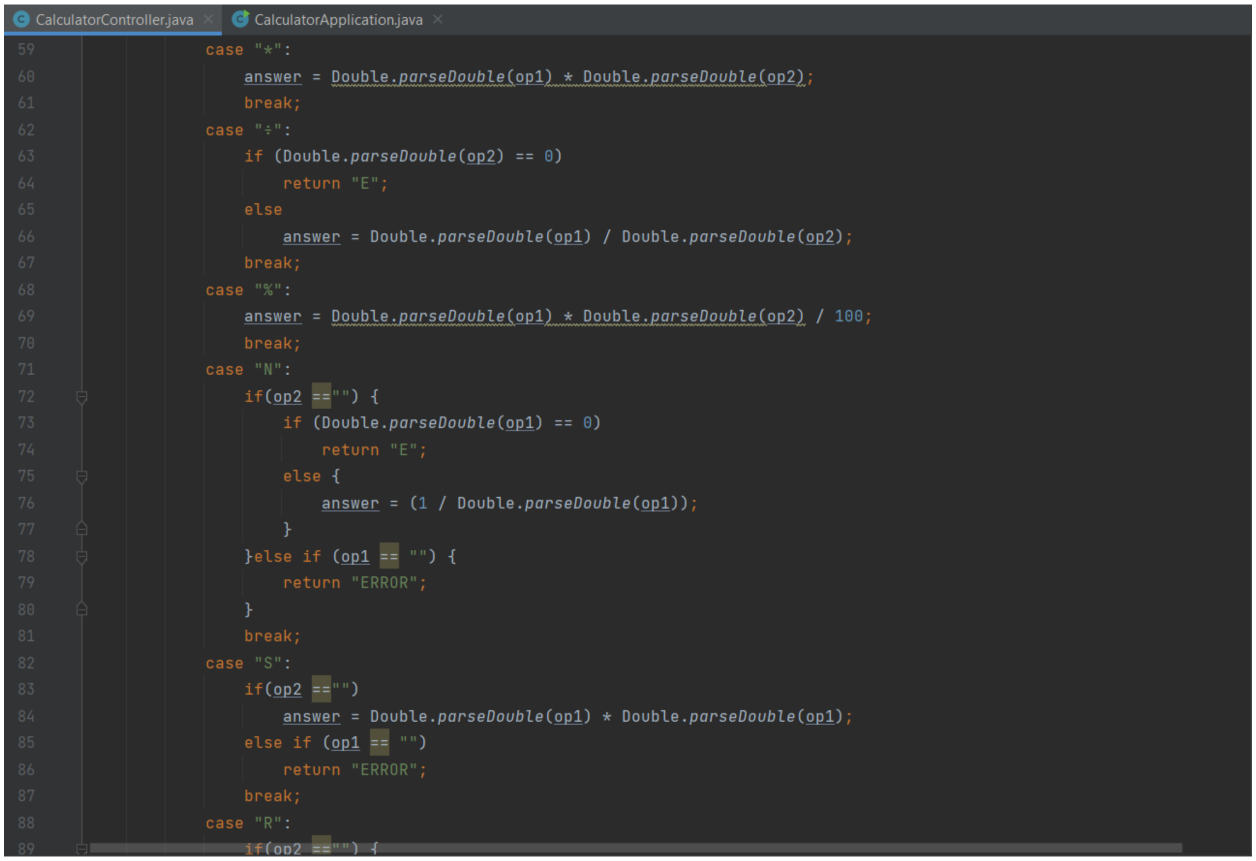
⇒.java file (controller):


CalculatorController.java × CalculatorApplication.java ×

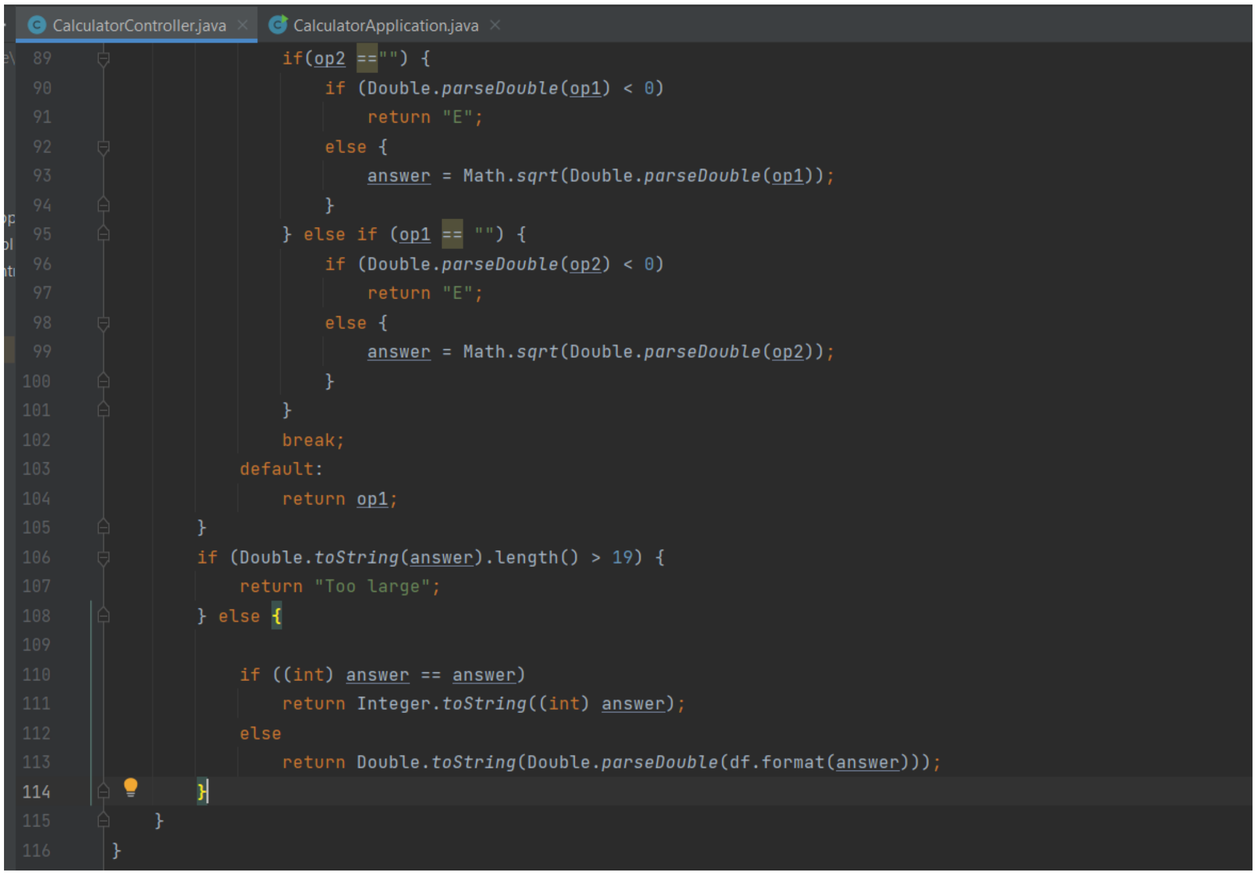
```java
package com.calculatorApp.calculator;
import org.springframework.web.bind.annotation.*;
import java.text.DecimalFormat;
no usages
@RestController
public class CalculatorController {
    no usages
    @CrossOrigin
    @PostMapping("/{displayValue}")
    public String calculate(@RequestBody String displayValue) {
        System.out.println(displayValue);
        String res1 = displayValue.replaceAll( regex: "\"", replacement: "");
        String res2 = res1.replaceAll( regex: "\\u00B2", replacement: "S");
        String res3 = res2.replaceAll( regex: "\\u221A", replacement: "R");
        String res4 = res3.replaceAll( regex: "\\u207B\u00B9", replacement: "N");
        String res5 = res4.replace( target: "+-", replacement: "-");
        String res6 = res5.replace( target: "-+", replacement: "-");
        String res = res6.replaceAll( regex: " ", replacement: "");
        System.out.println(res);
        String op1 = "";
        String op2 = "";
        String operation = "";
        int flag = 0;
        double answer = 0;
        DecimalFormat df = new DecimalFormat( pattern: "0.000000000");
        if (res.charAt(0) == '-') {
            op1 += res.charAt(0);
        for (int i = 1; i < res.length(); i++) {
            if ((Character.isDigit(res.charAt(i)) || res.charAt(i) == '.') && flag == 0) {
                op1 += res.charAt(i);
```


CalculatorController.java × CalculatorApplication.java ×

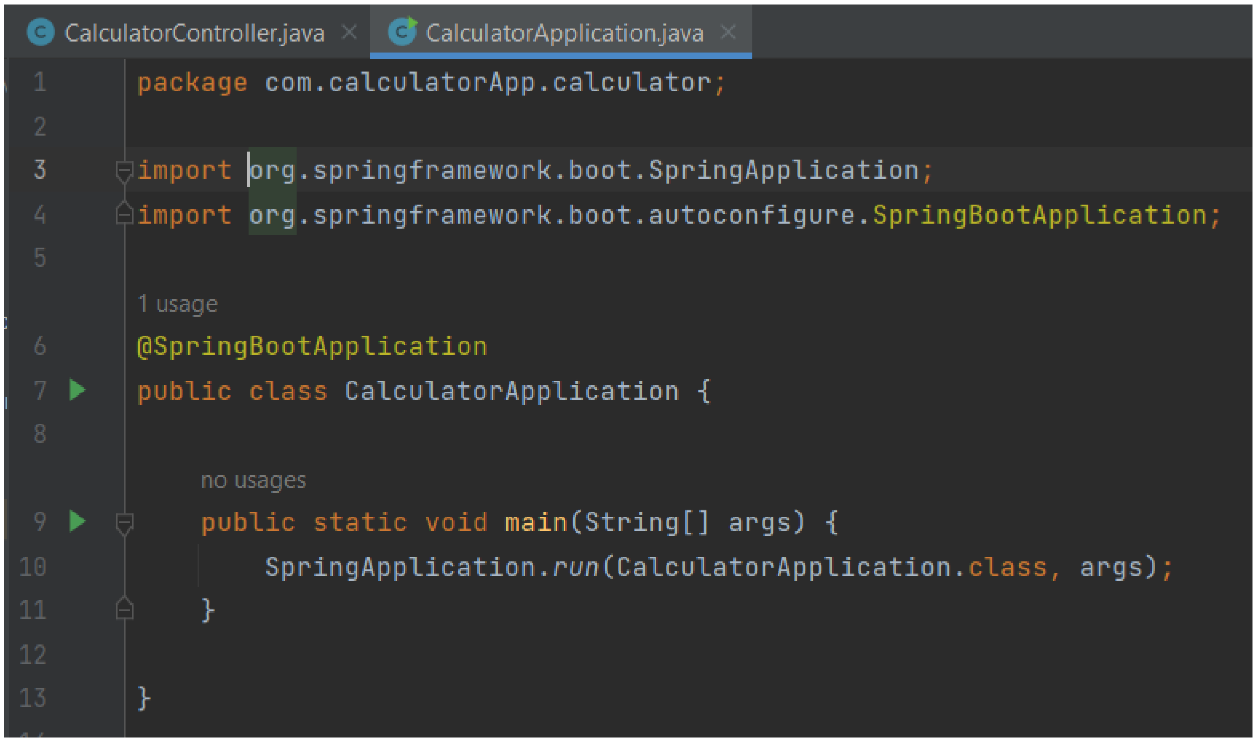
```java
            } else if ((Character.isDigit(res.charAt(i)) || res.charAt(i) == '.' || res.charAt(i) == '-') && flag == 1) {
                op2 += res.charAt(i);
            } else {
                operation += res.charAt(i);
                flag = 1;
            }
        }
    } else {
            for (int i = 0; i < res.length(); i++) {
                if ((Character.isDigit(res.charAt(i)) || res.charAt(i) == '.') && flag == 0) {
                    op1 += res.charAt(i);
                } else if ((Character.isDigit(res.charAt(i)) || res.charAt(i) == '.' || res.charAt(i) == '-') && flag == 1) {
                    op2 += res.charAt(i);
                } else {
                    operation += res.charAt(i);
                    flag = 1;
                }
            }
        }
        if(res == "")
            return "";
        if(res.charAt(res.length()-1) == '+' || res.charAt(res.length()-1) == '-' || res.charAt(res.length()-1) == '*' || res.charAt(res.l
            return "ERROR";
        switch (operation) {
            case "+":
                answer = Double.parseDouble(op1) + Double.parseDouble(op2);
                break;
            case "-":
                answer = Double.parseDouble(op1) - Double.parseDouble(op2);
                break;
            case "*":
```



```
            case "*":
                answer = Double.parseDouble(op1) * Double.parseDouble(op2);
                break;
            case "÷":
                if (Double.parseDouble(op2) == 0)
                    return "E";
                else
                    answer = Double.parseDouble(op1) / Double.parseDouble(op2);
                break;
            case "%":
                answer = Double.parseDouble(op1) * Double.parseDouble(op2) / 100;
                break;
            case "N":
                if(op2 ==""){
                    if (Double.parseDouble(op1) == 0)
                        return "E";
                    else {
                        answer = (1 / Double.parseDouble(op1));
                    }
                }else if (op1 == "") {
                    return "ERROR";
                }
                break;
            case "S":
                if(op2 =="")
                    answer = Double.parseDouble(op1) * Double.parseDouble(op1);
                else if (op1 == "")
                    return "ERROR";
                break;
            case "R":
                if(op2 ==""){
```



```java
                if(op2 =="") {
                    if (Double.parseDouble(op1) < 0)
                        return "E";
                    else {
                        answer = Math.sqrt(Double.parseDouble(op1));
                    }
                } else if (op1 == "") {
                    if (Double.parseDouble(op2) < 0)
                        return "E";
                    else {
                        answer = Math.sqrt(Double.parseDouble(op2));
                    }
                }
                break;
            default:
                return op1;
        }
        if (Double.toString(answer).length() > 19) {
            return "Too large";
        } else {

            if ((int) answer == answer)
                return Integer.toString((int) answer);
            else
                return Double.toString(Double.parseDouble(df.format(answer)));
        }
    }
}
```



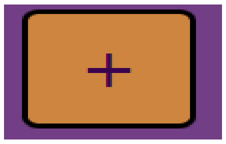
```java
package com.calculatorApp.calculator;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

1 usage
@SpringBootApplication
public class CalculatorApplication {

    no usages
    public static void main(String[] args) {
        SpringApplication.run(CalculatorApplication.class, args);
    }

}
```

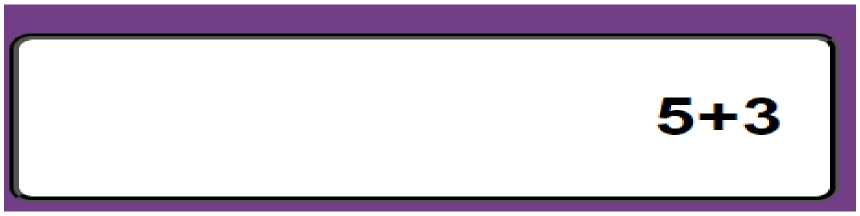
✦ Assumptions & User Manual:

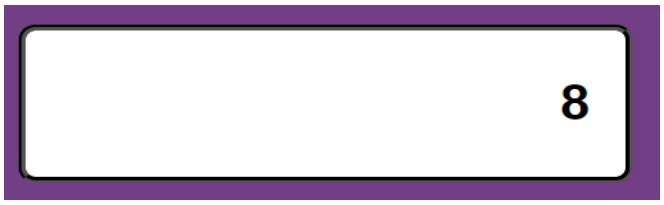
The calculator is a simple calculator that execute only simple operations and you should press equal for the operation to be executed.

Operations supported:

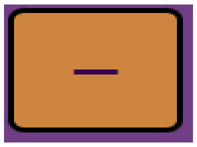
1. **Addition (+):**

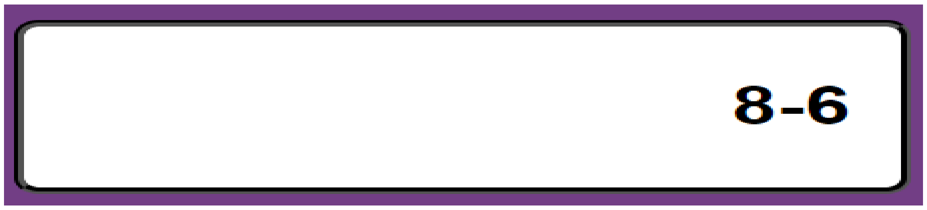


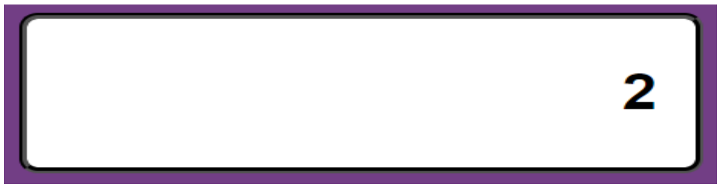


EQUALS ->


2. **Subtraction (-):**





EQUALS ->


3-8

EQUALS -> -5

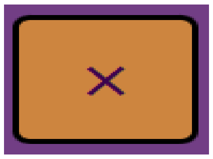
-6+-9

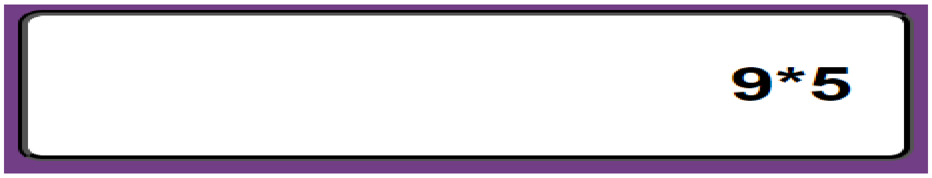
EQUALS -> -15

-6--3

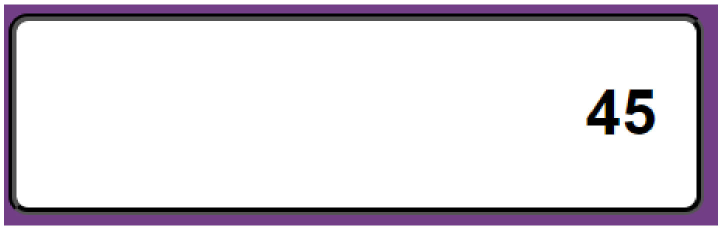
EQUALS -> -3

3. **<u>Multiplication (x):</u>**

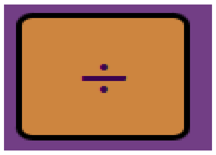


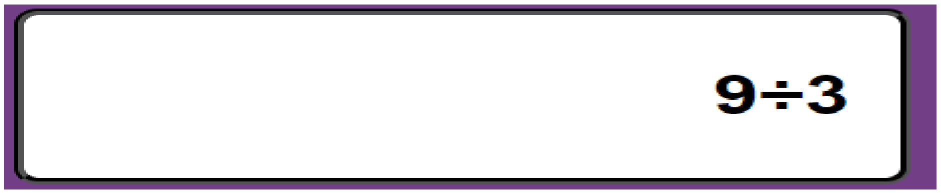


EQUALS ->

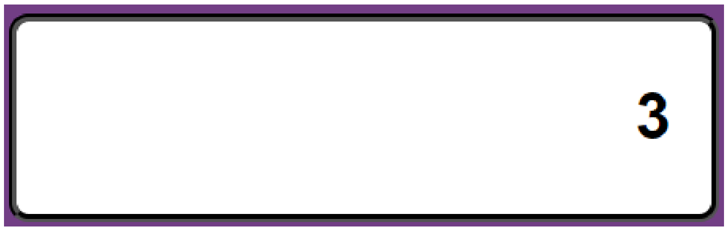


4. **<u>Division (÷):</u>**





EQUALS ->



5. **Squaring (^{2}):**

x^2

6^2

EQUALS -> 36

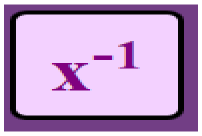
6. **Square Root (√):**

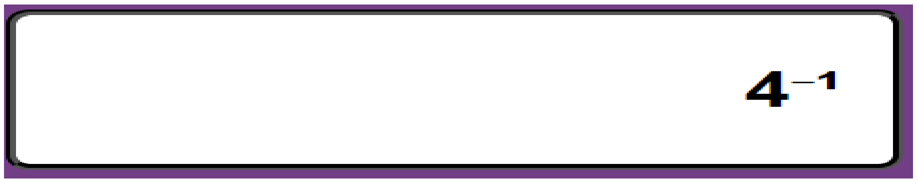
√

$\sqrt{81}$

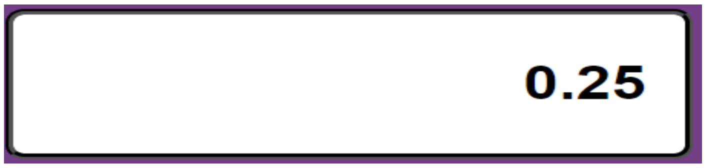
EQUALS -> 9

7. **Inverse ($^{-\ 1}$):**

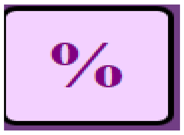


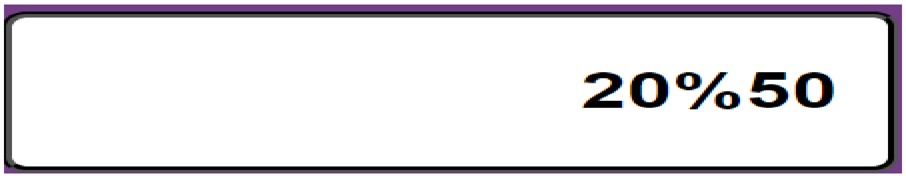


EQUALS ->

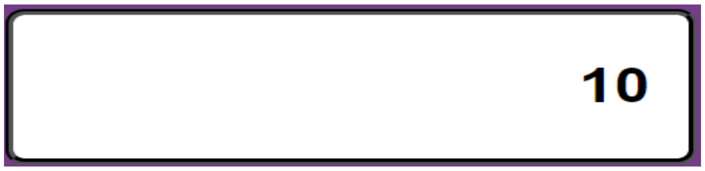


8. **Percentage sign (%):**



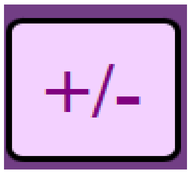


EQUALS ->



9. **Reverse/Negation sign (+/-):**

This button is used to get the negative of the number if it's positive and vice versa.

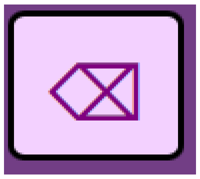


10. <u>Clear (CE):</u>

Clear button is used to clear the screen.

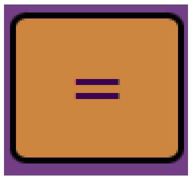
11. <u>Delete last character:</u>

Delete button is used to delete the last element from the right whatever it was a number or an operator.



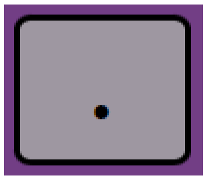
12. <u>Equal (=):</u>

Equal button is used to evaluate any valid expression then show the result on screen.



13. <u>Decimal (.):</u>

All these operations are supported on decimal numbers as well as integers.



✦ Errors handled:

- All the following are errors handled on my calculator web app by showing either "***E***" or "***Error***" on the screen.
  1. When dividing any number by zero.
  2. Square root to any negative number.
  3. Getting the inverse of zero.
- If the result of number is larger than 19 digits a "***Too large***" message is displayed on screen.

✦ Some corner cases or extra features:

- If the result is integer value, it's displayed as integer. Meanwhile, if the result is in float, it's displayed in decimal form.
- Performing operations on negative numbers is handled same as positive ones.
- When the user presses the "EQUAL" button consecutively more than once, no new calculations are issued.
- When the user presses an operator (FOR EXAMPLE: +) and then presses another operator whatever it was, similar or different, it's not appended to the expression. (No two operations are appended consecutively). Except for negative sign (illustrated in the next point).
- If the user enters two consecutive NEGATIVE SIGNS (--), this operation is evaluated as addition.
- If the user enters (+-) or (-+), this operation is evaluated as subtraction.
- No two decimal points are appended consecutively in the expression.

All these corner cases and features are clarified in the demo video attached in the same file.