Name : Mohamed Elnady Mohamed Gomaa

ID : 20011513

Programming II – Assignment 2 Report

Problem Statement

Building a web-based calculator using Angular and spring boot.

1- Back-End code:

Controller layer:

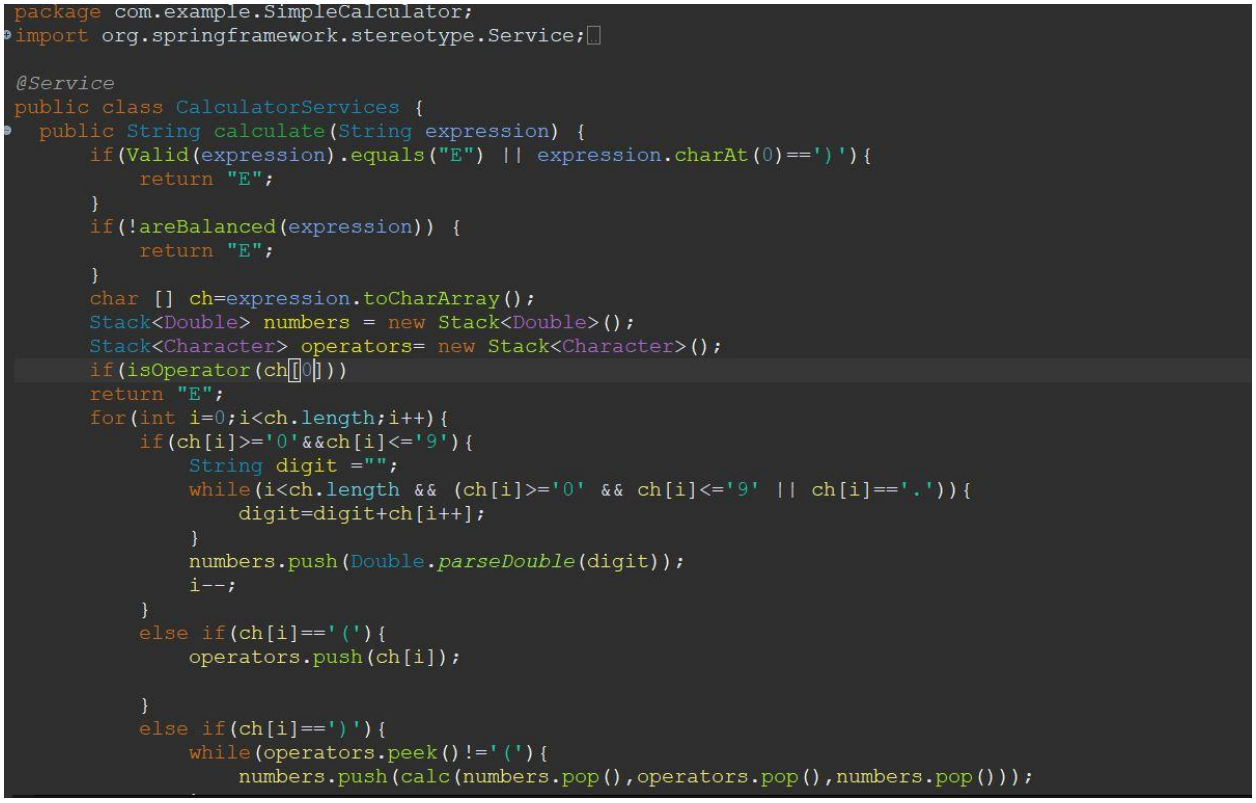
```
package com.example.SimpleCalculator;
import org.springframework.beans.factory.annotation.Autowired;
@CrossOrigin("http://localhost:4200")
@RestController

public class CalculatorController {
    @Autowired
    CalculatorServices service;
    @PostMapping
    public String getData(@RequestBody String expression) {
        String result=service.calculate(expression);
        if(result.equals("Infinity")|| result.equals("-Infinity")) {
            return "E";
        }
        return result;

    }
```

- For define controller layer (layer that deals with requests) we use *RestController* annotation
- *PostMapping* for dealing with requests
- *RequestBody* for receiving objects sent from front-end code .

Service Layer:



```
package com.example.SimpleCalculator;
import org.springframework.stereotype.Service;

@Service
public class CalculatorServices {
  public String calculate(String expression) {
      if(Valid(expression).equals("E") || expression.charAt(0)==')'){
          return "E";
      }
      if(!areBalanced(expression)) {
          return "E";
      }
      char [] ch=expression.toCharArray();
      Stack<Double> numbers = new Stack<Double>();
      Stack<Character> operators= new Stack<Character>();
      if(isOperator(ch[0]))
      return "E";
      for(int i=0;i<ch.length;i++){
          if(ch[i]>='0'&&ch[i]<='9'){
              String digit ="";
              while(i<ch.length && (ch[i]>='0' && ch[i]<='9' || ch[i]=='.')){
                  digit=digit+ch[i++];
              }
              numbers.push(Double.parseDouble(digit));
              i--;
          }
          else if(ch[i]=='('){
              operators.push(ch[i]);

          }
          else if(ch[i]==')'){
              while(operators.peek()!='('){
                  numbers.push(calc(numbers.pop(),operators.pop(),numbers.pop()));
```

-*Service* annotation for defining service layer in which business logic is written.

<u>Approach of Calculating expressions:</u>

Expression to be calculated is in string format, we have two stacks data structures, 1 for numbers and the other for operators.

Note: *Autowired* annotation is used in controller layer for <u>dependency injection</u> to make an instance of service layer in controller layer.

2- Front-End code:

C	%	1/x	←
x^2	√x	.	/
7	8	9	*
4	5	6	-
1	2	3	+
(	0	)	=

Description for how it works:

When a user clicks on a button the number of button is added to a string which appears on screen, when equal button is clicked, expression is sent to server and calculations is done and finally it returns back the result as a string on screen.

HTML code:

```
<div class="calculator">
    <div class="screen">{{expression}}</div>
     <div class="buttons">
        <button class="clr" (click)="clearAll()">C</button>
        <button (click)="showOn('%')">%</button>
        <button (click)="reciprocal(expression)">1/x</button>
        <button (click)="undo()">&#8592;</button>
        <button (click)="square(expression)">x <sup>2</sup></button>
        <button (click)="showOn('√')">&#8730;x</button>
        <button (click)="showOn('.')">.</button>
        <button (click)="showOn('/')">/</button>
        <button (click)="showOn('7')">7</button>
        <button (click)="showOn('8')">8</button>
        <button (click)="showOn('9')">9</button>
        <button (click)="showOn('*')">*</button>
        <button (click)="showOn('4')">4</button>
        <button (click)="showOn('5')">5</button>
        <button (click)="showOn('6')">6</button>
        <button (click)="showOn('-')">-</button>
        <button (click)="showOn('1')">1</button>
        <button (click)="showOn('2')">2</button>
        <button (click)="showOn('3')">3</button>
        <button (click)="showOn('+')">+</button>
        <button (click)="showOn('(')">(</button>
        <button (click)="showOn('0')">0</button>
        <button (click)="showOn(')')">)</button>
        <button class="eq" (click)="evaluate()">=</button>
```

CSS styling code:

```css
.buttons {
    display: grid;
    border-bottom: 1px solid #999;
    border-left: 1px solid #999;
    grid-template-columns: repeat(4, 1fr);
}
button {
    border: 1.5px solid #999;
    line-height: 89px;
    text-align: center;
    font-size: 28px;
    cursor: pointer;
}
button:hover{
    background-color: bisque;
}
button.eq{
    background-color: rgb(98, 159, 159);
}
.screen{

    text-align: right;
    height: 100px;
    line-height: 70px;
    padding: 16px 8px;
    font-size: 25px;
    background-color: rgb(230 230 230);
```

TypeScript code:

```typescript
import { Component } from '@angular/core';
import { HttpClient} from '@angular/common/http';

@Component({
  selector: 'calculator',
  templateUrl: './calculator.component.html',
  styleUrls: ['./calculator.component.css']
})
export class CalculatorComponent {
  constructor(private http:HttpClient){

  };
  url="http://localhost:9091";
 expression='';
 clearAll(){
  this.expression='';
 }
 showOn(num:String){
  this.expression=this.expression+num;
 }
 reciprocal(x:string){
  this.clearAll();
  this.expression="1"+"/"+x;
 }
 square(x:string){
  this.clearAll();
```

How it is sent to the server:

Importing http modules to use post method to send the expression to server on a specific localhost & subscribe function to receive the response from server(result).

-we use **response type** as a text since subscribe function returns Observables type and we want to return a string.

```
}
square(x:string){
 this.clearAll();
 this.expression=x+"^"+"2";
}
undo(){
 this.expression= this.expression.slice(0, this.expression.length - 1);

}
valuate(){
 this.http.post(this.url,this.expression,{responseType:'text'}).subscribe((result)=>{
   this.expression=result;
 });
 }

}
```

3- **Errors Handling:**

- Division by zero gets ‘E’ output .
- Square root of –ve number gets ‘E’ output.
- Pressing = many times doesn’t give any result.

4- **Sample Runs:**