Course: EGDF20
Module: EGE202 Application Programming

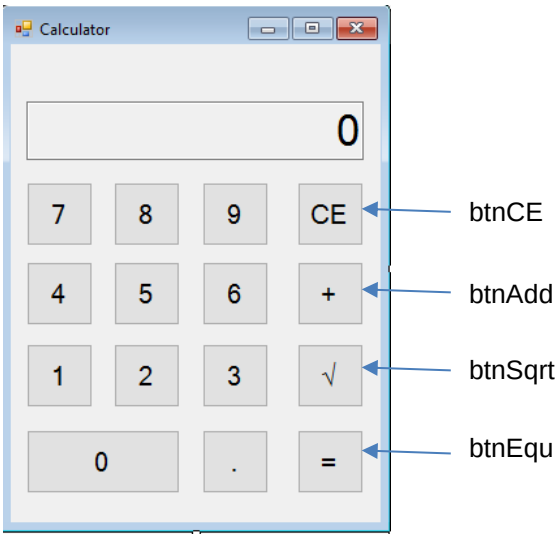
Practical 5b: Calculator Application: Implementing Coding Logic for Calculation

Objectives: At the end of this lab, the student should be able to describe some of the core elements and operations involved in developing a GUI software application. It will highlight some of advance techniques in event handling and understand how to develop and effective GUI.

Exercise 1 – Implementing the Calculator's Operators Logic

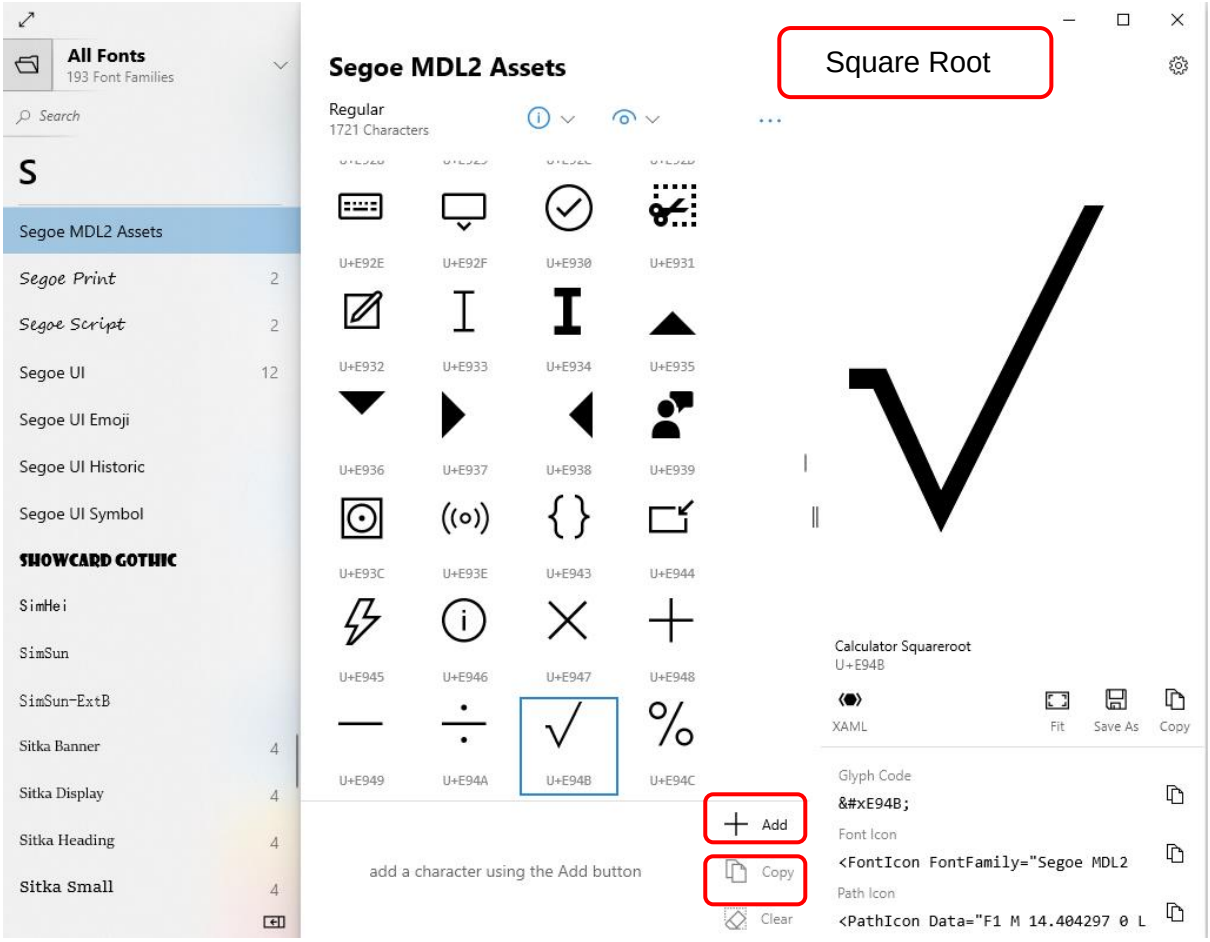
Part 1: Adding Basic Calculator Operators

1. Using the same solution/project from Practical 4a open ***Form Designer*** and copy and paste 4 *Button* controls into the *MainForm* window. Modify the *text* and *name* properties based on the information below. (Windows **'Character Map'** program can be used to insert non-ASCII characters)



NEW!

{Name} From	{Text}	{Tag}
btnCE	CE	CE
btnAdd	+	Add
btnSqrt	√	Sqrt
btnEqu	=	Equ



2. Double click on button '=' to create a *btnEqu_Click(…)* event handler. Add the following codes.

```
string opr = "";
double operand = 0;
private void btnEqu_Click(object sender, EventArgs e)
{
    double operand2 = Double.Parse(txtResults.Text);
    switch (opr)
    {
        case "Add":
            operand = operand + operand2;
            txtResults.Text = operand.ToString();
            break;
        default:
            break;
    }
    opr = "";
}
```

3. At the ***Form Designer***, select button **'+'**. At the **properties panel** click and select the ***event*** button. Next to the Click event, type in *operator_Click* and press ENTER key.

4. Add the following codes *operator_Click(. . .)*

```
bool flagOpPressed = false;
private void operator_Click(object sender, EventArgs e)
{
    // get the operand value
    operand = Double.Parse(txtResults.Text);

    // get the operator
    Button btn = (Button)sender;
    opr = btn.Tag.ToString();
        flagOpPressed = true;
}
```

5. From here the code editor, make the following modification to *numPad_Click (…)* event handler.

```
private void numPad_Click(object sender, EventArgs e)
{
    Button btn = (Button)sender;
    string num = btn.Text;

    string temp = txtResults.Text;

    //clear display if operator is pressed
    if (flagOpPressed == true)
    {
        temp = "";
        flagOpPressed = false;
    }

    if (temp == "0")

    . . . .
```

6. Double click on the button **'CE'** in the **Form Designer** to create *btnCE_Click(…)* event handler and add the following codes.

```
private void btnCE_Click(object sender, EventArgs e)
{
    opr = "";
    operand = 0;
    flagOpPressed = false;
    txtResults.Text = "0";
}
```

7. Build and test your application.

No	Actions	Observation
1	Perform the following calculation: 1 + 2, then press =	

2	Perform the following calculation: 1 + 2 + 3, then press =	Can it perform multiple addition?
	Perform the following calculation: 1 + 2, then press equal, then + 3, then press equal	Can it perform multiple addition?

8. In order to perform multiple addition, computation must be done each time the user click the '+' button. Add the following codes to *operator_Click(. . .)*

```
private void operatorClick(object sender, EventArgs e)
{
    btnEqu.PerformClick();

    // get the operand value
    operand = Double.Parse(txtResults.Text);

    // get the operator
    Button btn = (Button)sender;
    opr = btn.Tag.ToString();
    flagOpPressed = true;
}
```

9. Build and test your application.

"+" is a binary operation which means it requires 2 operands.
"√" is a unary operation which means it requires only 1 operand.

10. At the ***Form Designer***, select button **'√'**. At the **properties panel** click and select the ***event*** button. Next to the Click event, type *u_operatorClick* and press ENTER key.
11. Modify *u_operatorClick(. . .)* and to include the following codes to handle square root operation. (Note: Square root only operates on one operand as oppose to addition which operates on two operands)

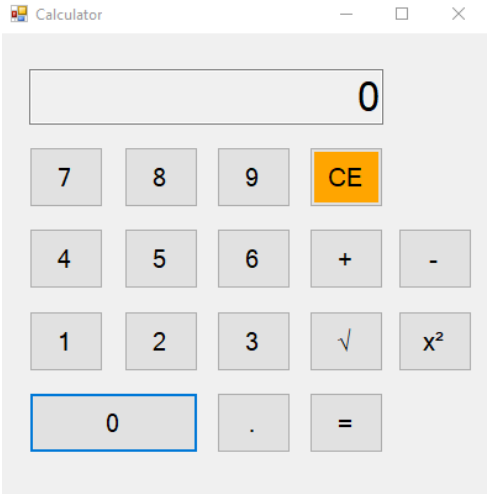
```
private void u_operatorClick(object sender, EventArgs e)
{
    Button btn = (Button)sender;
    string u_opr = btn.Tag.ToString();

    double value = Double.Parse(txtResults.Text);
    string results;
    switch (u_opr)
    {
        case "Sqrt":
            results = Math.Sqrt(value).ToString("N10");
            txtResults.Text = results.TrimEnd('0').TrimEnd('.');
            break;
    }
}
```

No	Actions	Explanation
1	What does `ToString("N10")` do?	
2	What does `TrimEnd('0').TrimEnd('.')` do?	

12. Build and test your application.

(Assignment) Implementing Enhancement on Calculator App



1. (Functionality Enhancement)
   - Implement two more buttons (subtraction and square) (**Hint:** Square in ***Math*** Library is Power of 2)
2. (Visual Enhancement)
   - Change the background color for button 'CE' (**Hint:** use ***BackColor*** property)
   - Use an image rather than text for the square button (**Hint:** use **CharacterMap** to look for **Superscript 2**)