2023 年 5 月 18 日（木）実施

式と演算子

式

式とは，1 つの値，オブジェクト，メソッド，または名前空間(*)として評価することの出来る，1 つ以上のオペランド（被演算子）と，0 個以上の演算子の並びである。 式には，リテラル値(**)，メソッドの呼び出し，演算子とそのオペランド，または簡易名を含めることが出来る。 単純な名前には，変数，型メンバー，メソッドパラメーター，名前空間，または型の名前を指定出来る。

(*)**名前空間**　その内部にある識別子（型，関数，変数などの名前）のスコープ（適用範囲）を定める宣言領域で，コードを論理グループにまとめたり，名前の競合を回避したりするために使用される。

(**)**リテラル値**　数値や文字列のデータ値で，すべてのリテラル値には型が関連付けられている。

演算子

演算子とは，式または文の中で 1 つ以上のオペランドに適用されるプログラム要素である。各演算子には優先順位が定義されている。 優先順位のレベルが異なる複数の演算子を含む式の場合，演算子の優先順位によって演算子が評価される順序が決定される。C#言語の主な演算子を優先順位の順に挙げる（赤字が演算子，黒字がオペランドを表す）。各分類の中の優先順位は等しい。

分類	表現	説明
	x.y	メンバーアクセス
	f(x)	メソッドの呼び出し
	a[x]	配列アクセス，インデクサーアクセス
	x++	後置インクリメント（x=x+1）x++の参照は x の値となる。
	x−−	後置デクリメント（x=x−1）
	new T(...)	オブジェクトの作成
単項演算子	!x	論理否定
	++x	前置インクリメント　++x の参照は x+1 の値となる。
	−−x	前置デクリメント
	(T)x	キャスト　x を明示的に T 型に変換する。
乗法演算子	x*y	乗算
	x/y	除算
	x%y	剰余算
加法演算子	x+y	加算，文字列の連結
	x−y	減算

＊ 括弧を使用すると，演算子の優先順位を変更することが出来，括弧内を優先して評価する。

＊＊ 更に優先順位の低い，**関係演算子**，**等値演算子**，**論理演算子**については，次回に取り上げる。

本日の課題

今回の授業では，四則演算（加算，減算，乗算，除算）の式について学ぶ。

手順

1）コントロールの追加

前回と同様にして，JimboCalc.sln を開く。フォームデザイナーで，ボタンを追加する。それぞれのプロパティは次の通りである（Text プロパティの値は半角文字で与える）。

【ボタンのプロパティ】

button11		button12		button13		button14	
Width	40	Width	40	Width	40	Width	40
Height	35	Height	35	Height	35	Height	35
Text	.	Text	+	Text	−	Text	*
X	132	X	192	X	192	X	192
Y	321	Y	321	Y	279	Y	237

button15		button16		button17	
Width	40	Width	100	Width	40
Height	35	Height	35	Height	35
Text	/	Text	=	Text	C
X	192	X	132	X	12
Y	195	Y	153	Y	153

button11 の Text プロパティの値は小数点(ドット)

2) コーディング

前回，変数について学んだが，実数を格納する double 型の変数 num1 及び result と整数を格納する変数 num2 とが次の様に宣言されているとすると，

```
double num1 = 1.23;
int num2 = 10;
double result;
```

num1 と num2 とを参照した値同士の和を result に代入する文は

```
result = num1 + num2;
```

と書ける。この場合，num1 と num2 とではデータ型が異なるが，より精度の高い double 型で加算が行われる。

作成中の電卓では，ボタンを押して得られる，数字を並べた文字列を利用するので，今回はその文字列を数値化したものを num1 に代入し，num2 の値は宣言時の初期化で与えるものとする。

今回の num2 の値は仮のものに固定

【手順】

先ず，3 つの変数 num1，num2 及び result を宣言し，num2 は 10 で初期化する。変数は Form1.cs 内の全体で有効なものとするため，クラスのフィールドに追加して宣言する。

```
13  3 個の参照
    public partial class Form1 : Form
14  {
15      string degits = "";
16      string formula = "";
17      double num1;
18      int num2 = 10;
19      double result;
20
        1 個の参照
21      public Form1()
22      {
```

次に，フォームデザイナーに戻って，button11 をダブルクリックして，イベントハンドラを作成し，処理を記述する。

```
        1 個の参照
106     private void button11_Click(object sender, EventArgs e)
107     {
108         degits += button11.Text;
109         formula += button11.Text;
110         label2.Text = degits;
111         label1.Text = formula;
112     }
```

続けて，演算子のボタン button12～button15 のそれぞれをダブルクリックして，イベントハンドラを作成し，処理を記述する。

なお，ここで double.Parse(degits) は degits に格納されている数字を並べた文字列を解析して，数値化している。また，num2.ToString() は num2 に格納されている整数に対して，文字列に変換するメソッドを適用している。

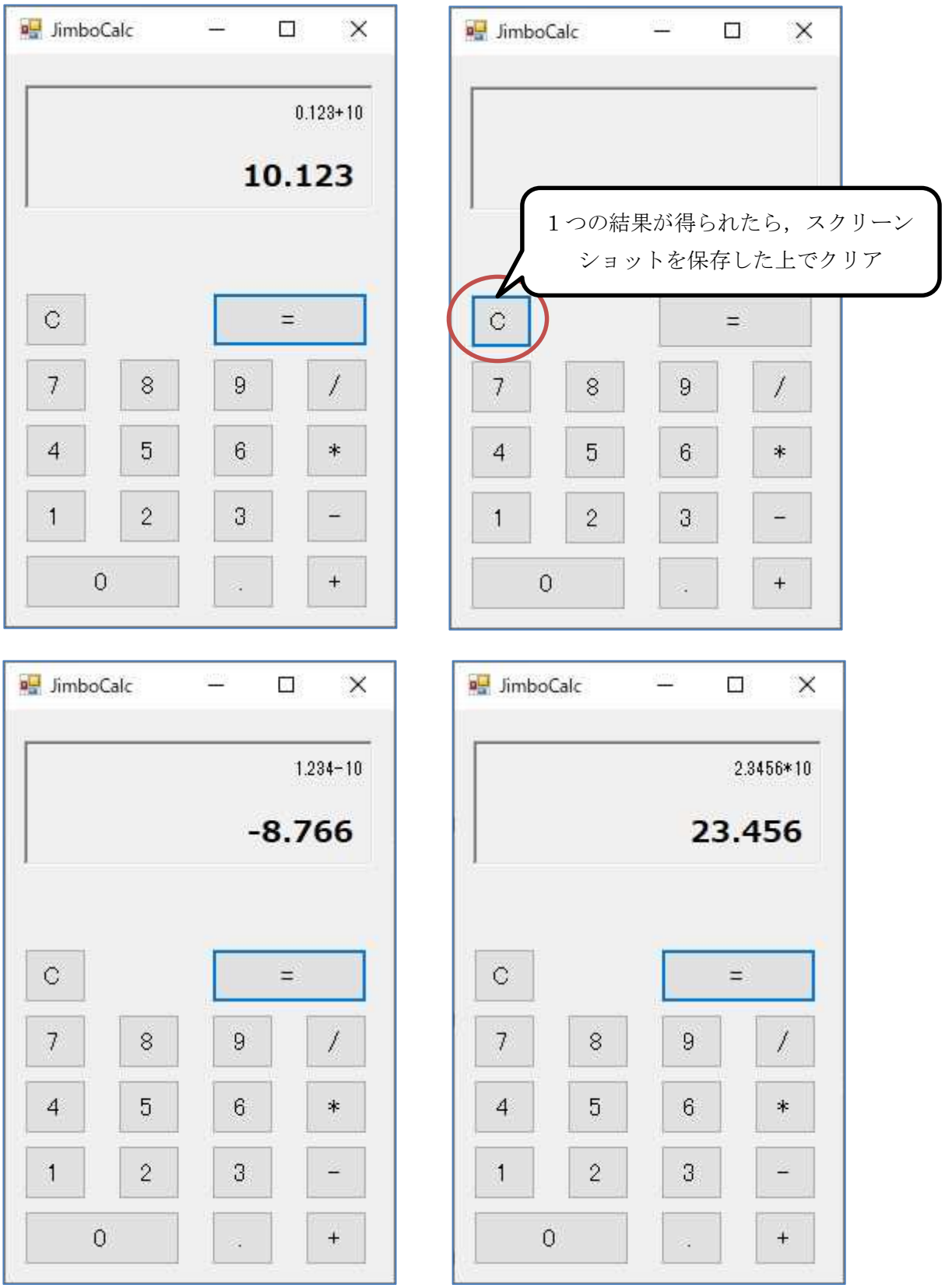
（図は次のページ）

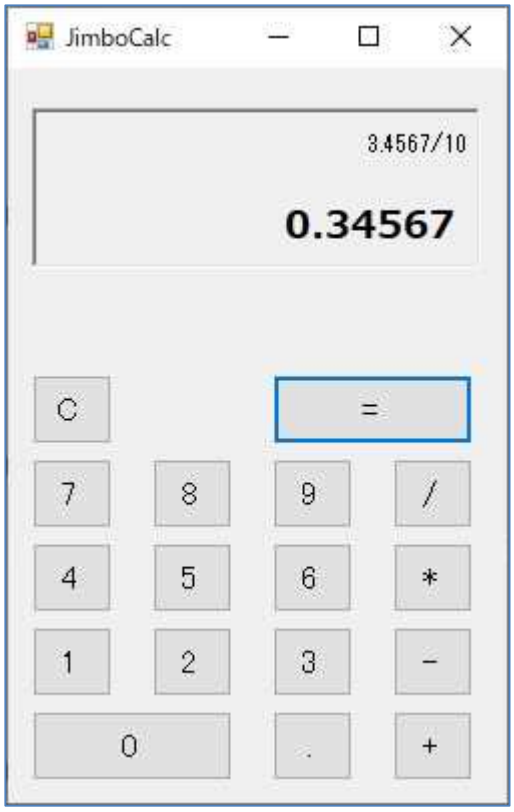
```
        1 個の参照
114     private void button12_Click(object sender, EventArgs e)
115     {
116         num1 = double.Parse(degits);
117         formula += button12.Text;
118         formula += num2.ToString();
119         label2.Text = "";
120         label1.Text = formula;
121         result = num1 + num2;
122     }
123
        1 個の参照
124     private void button13_Click(object sender, EventArgs e)
125     {
126         num1 = double.Parse(degits);
127         formula += button13.Text;
128         formula += num2.ToString();
129         label2.Text = "";
130         label1.Text = formula;
131         result = num1 - num2;
132     }
133
        1 個の参照
134     private void button14_Click(object sender, EventArgs e)
135     {
136         num1 = double.Parse(degits);
137         formula += button14.Text;
138         formula += num2.ToString();
139         label2.Text = "";
140         label1.Text = formula;
141         result = num1 * num2;
142     }
143
        1 個の参照
144     private void button15_Click(object sender, EventArgs e)
145     {
146         num1 = double.Parse(degits);
147         formula += button15.Text;
148         formula += num2.ToString();
149         label2.Text = "";
150         label1.Text = formula;
151         result = num1 / num2;
152     }
```

更に，フォームデザイナーに戻って，結果表示の為の『=』と書かれた button16 及びクリアの為の『C』と書かれた button をダブルクリック，イベントハンドラを作成し，処理を記述する。

```
        1 個の参照
154     private void button16_Click(object sender, EventArgs e)
155     {
156         label2.Text = result.ToString();
157     }
158
        1 個の参照
159     private void button17_Click(object sender, EventArgs e)
160     {
161         degits = "";
162         formula = "";
163         label1.Text = "";
164         label2.Text = "";
165     }
```

ここで，『Form1.cs の保存』ボタンを押してから，『開始』ボタンを押して，プログラムを実行して，四則演算（加算，減算，乗算，除算）のそれぞれについて，動作を確かめる。





提出物：

1) フォームのデザインファイル Form1.Designer.cs をメールに添付して提出する。
2) コードエディタで編集したソースファイル Form1.cs をメールに添付して提出する。
3) 実行の加算の結果のスクリーンショット第 4 回加算実行結果.jpg（.png も可）をメールに添付して提出する。
4) 実行の減算の結果のスクリーンショット第 4 回減算実行結果.jpg（.png も可）をメールに添付して提出する。
5) 実行の乗算の結果のスクリーンショット第 4 回乗算実行結果.jpg（.png も可）をメールに添付して提出する。
6) 実行の除算の結果のスクリーンショット第 4 回除算実行結果.jpg（.png も可）をメールに添付して提出する。
7) 質問を記述したファイル Prog2_Questions_4th.txt に解答を書き込んで保存し，メールに添付して提出する。