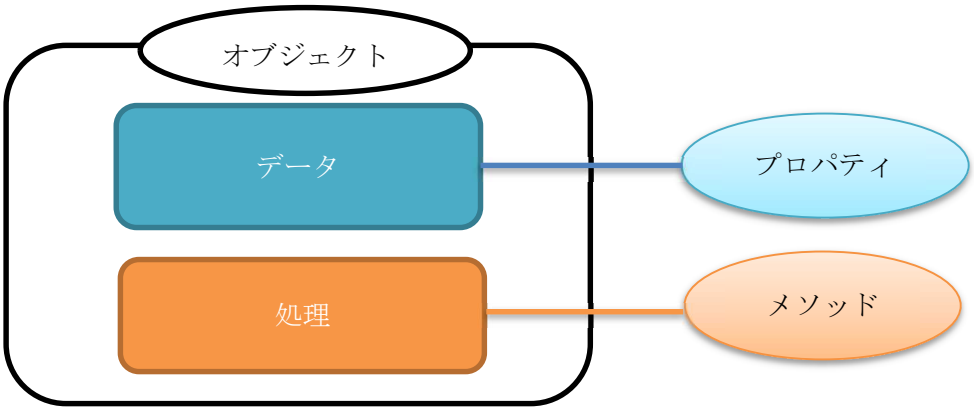
2023 年 6 月 1 日（木）実施

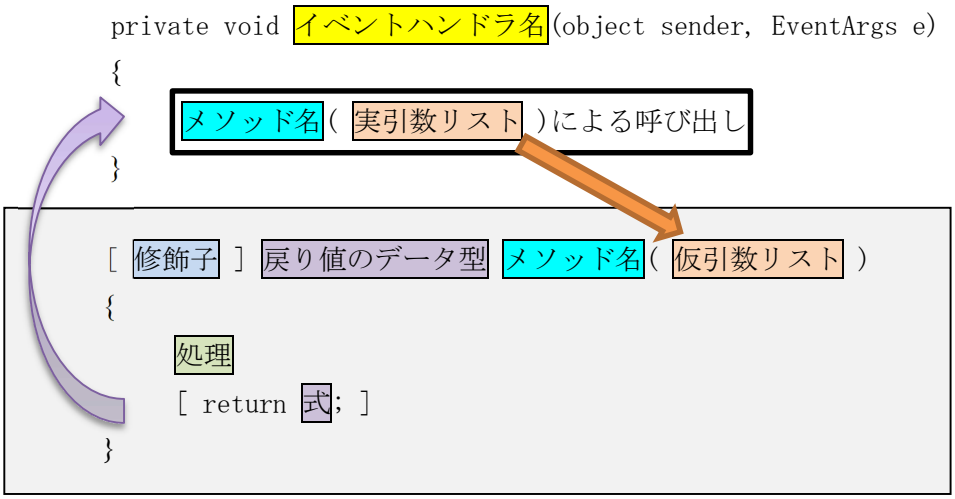
オブジェクト指向とメソッド

前回，手続き型のプログラムは，順次，選択，反復という標準的な制御構造の組み合わせで記述されることを学んだが，Windows フォームアプリケーションを作成していくには，更に**オブジェクト指向プログラミング**という考え方が必要になってくる。オブジェクトとは，操作の対象を意味している言葉であり，オブジェクト指向のプログラムでは，データと処理とをひとまとまりのオブジェクトとして捉える。



例えば, button1.Text とは, button1 というオブジェクトに含まれるデータのうちの一つで Text と名付けられた**プロパティ**であり，MessageBox.Show("Hello! ")とは，MessageBox というオブジェクトに含まれる処理のうち Show と名付けられた**メソッド**（画面にメッセージボックスを開き，指定された文字列を表示する処理）である。

メソッドとはプログラムの中で複数回利用する処理をひとまとめにしたもので，あるクラスの機能として，プログラム作成者が自ら作成することが出来る。Windows フォームアプリケーションでは，イベントハンドラからメソッドを呼び出して利用する。



ここで，[ ]内は省略可能であり，修飾子が無いメソッド，引数が無いメソッド，return 文のな

いメソッドもある。戻り値のデータ型は，**return 文**で戻されるデータの型となり，戻り値は呼び出し側に戻される。また，return 文の無い場合の戻り値のデータ型は **void** と表される。なお，仮引数リストには，メソッド内で利用する仮引数をそのデータ型と共に指定する。

例 1）
```
public int wa (int x, int y) {
    return x+y;
}
```

＊ 修飾子 public は，このメソッドがどのクラスからも利用出来ることを指定する。

例 2）
```
private void dispsum (int x, int y) {
    MessageBox.Show(x +"と"+ y +"との和は"+ (x+y) +"です。");
}
```

＊ 修飾子 private は，このメソッドが所属するクラス内でしか利用出来ないことを指定する。

本日の課題

今回の授業では，メソッドの作り方及び使い方について学ぶ。

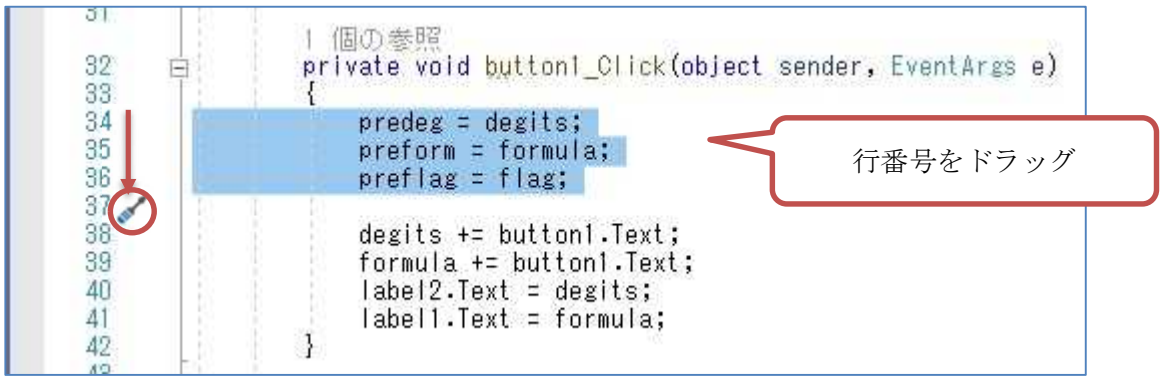
手順

1）コーディング

今回は，メソッドの抽出機能の使い方及びメソッドを一から作成する方法の双方を用いる。
なお，メソッドの作成後に仕上げの処理を追加する。

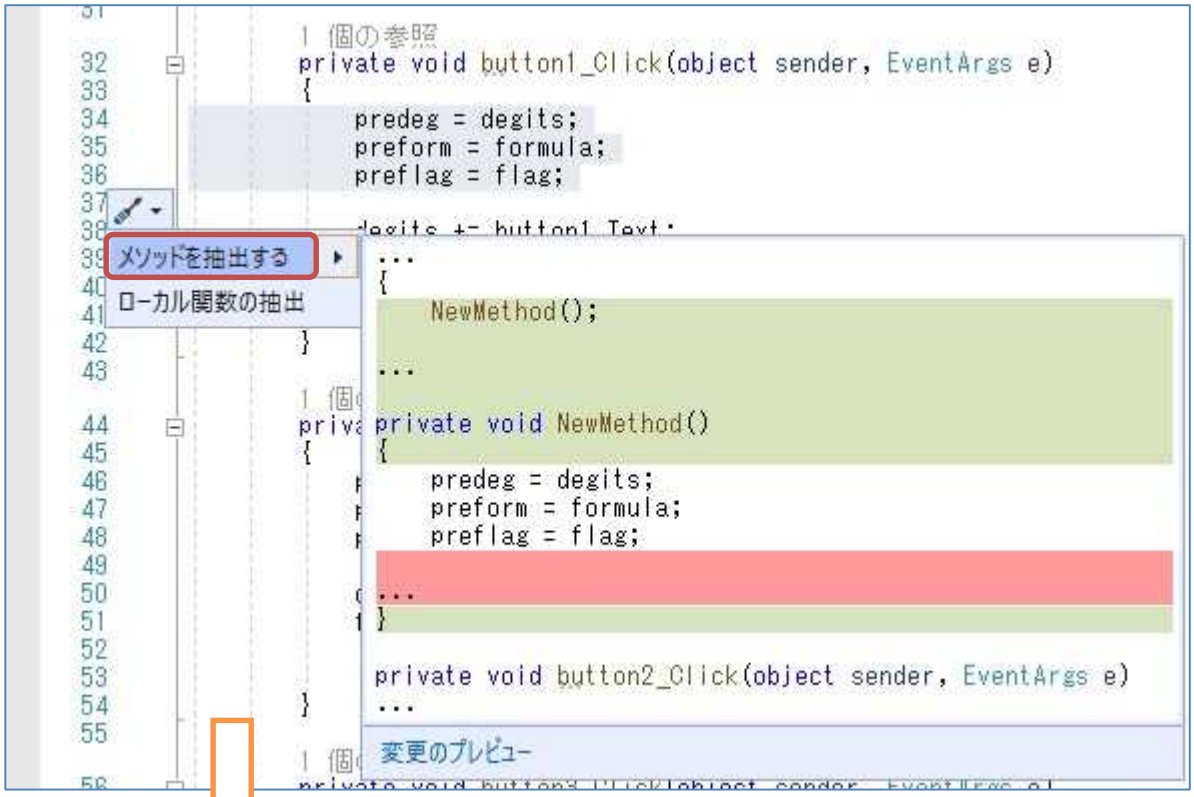
【手順】

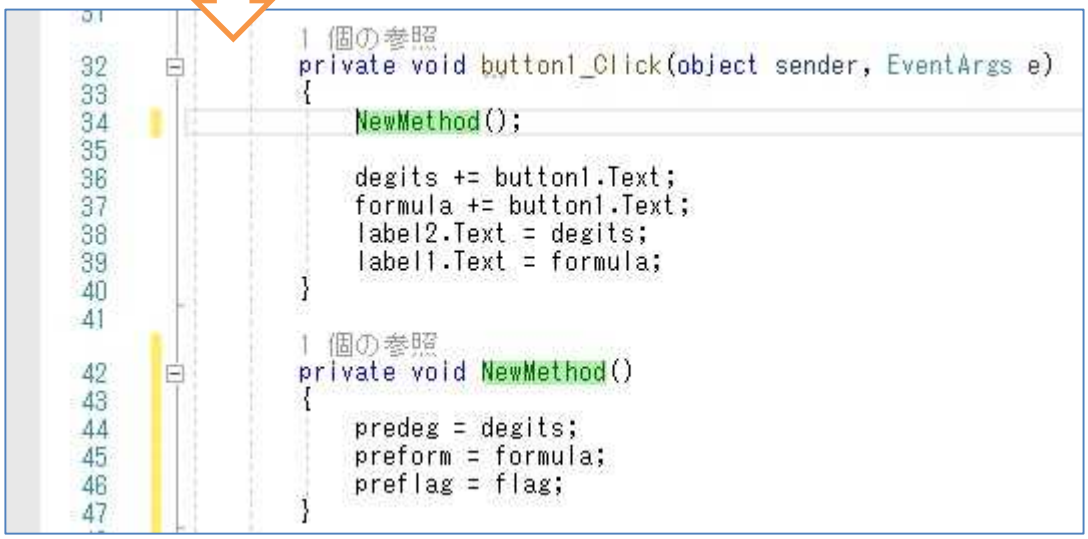
先ず，メソッドの抽出機能を利用する。抽出してメソッド化したい範囲（ここでは predeg～flag;）の行番号の箇所をドラッグするとマイナスドライバーのアイコンが現れる。



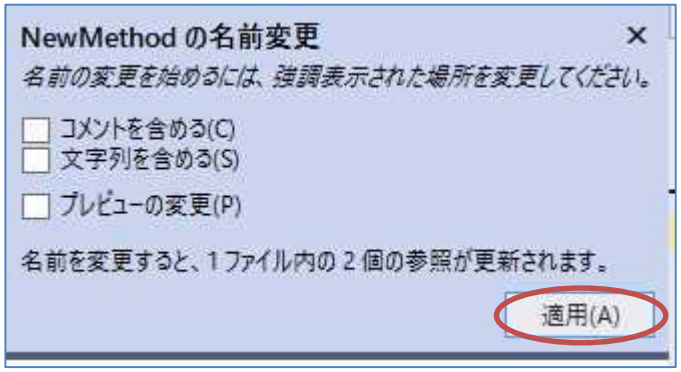
このマイナスドライバーのアイコンをクリックするとメニュー及びプレビューが表示されるので，「メソッドを抽出する」をクリックする。

（図は次のページ）

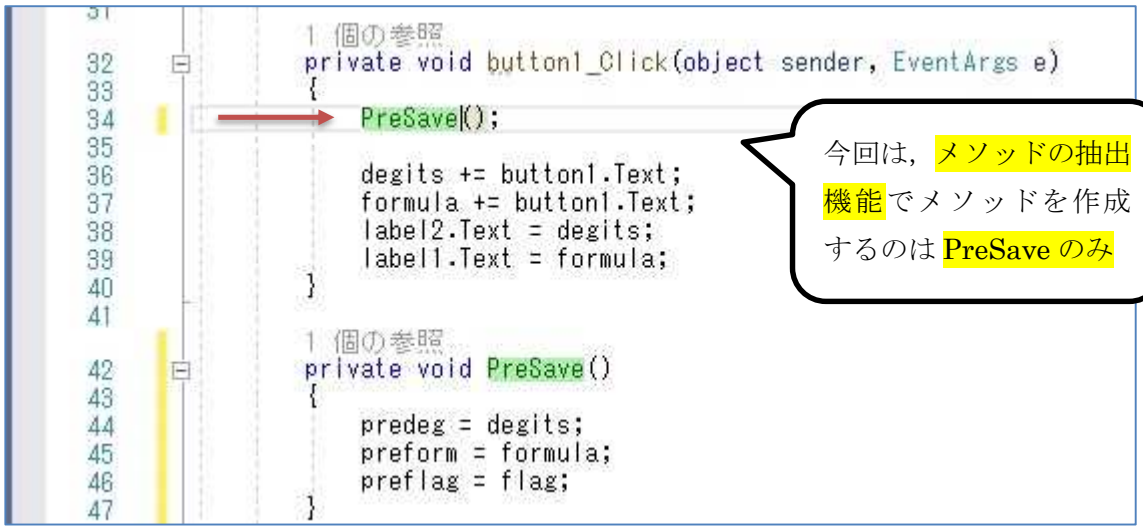




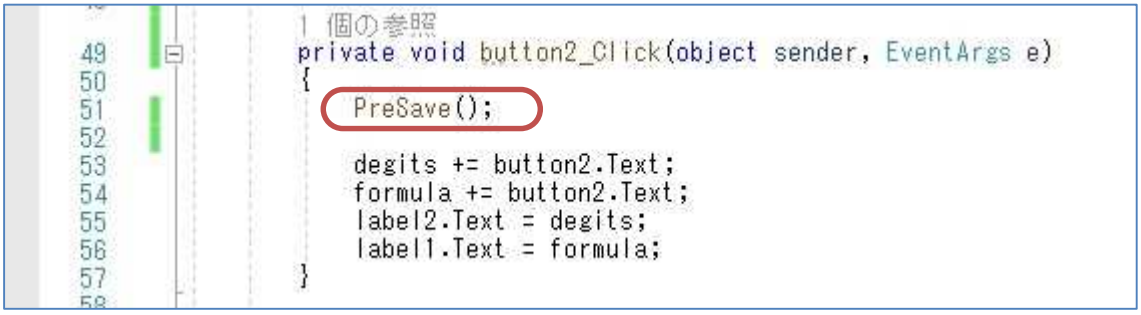
NewMethod という仮の名前が付いているが，これを PreSave に書き換えてから『NewMethod の名前変更』の『適用』をクリックする。



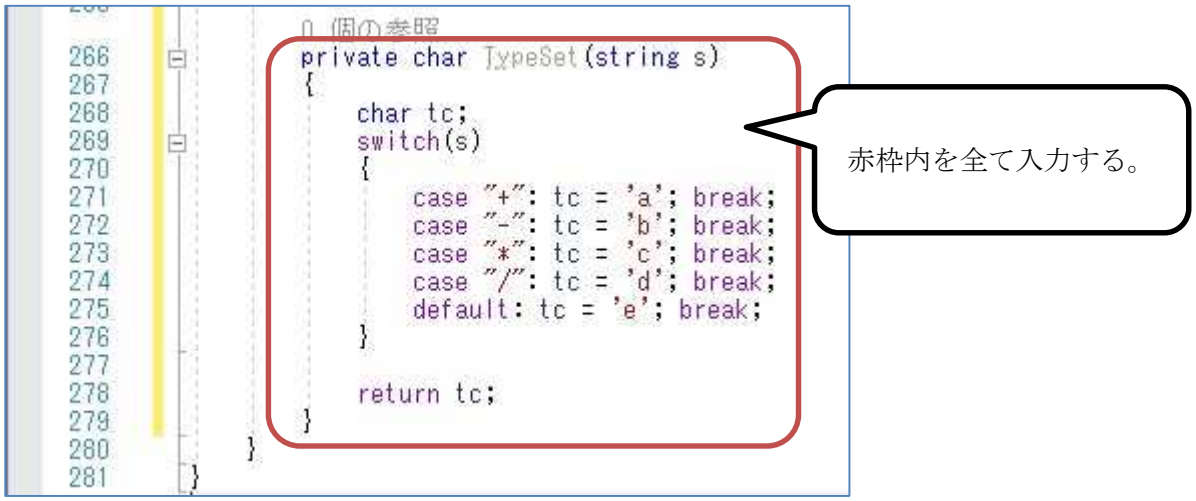
（図は右及び次のページ）



続けて，button2_Click～button15_Click の中で，PreSave メソッドに抽出した 3 行と同じ内容（predeg～flag;）が記述されている箇所をメソッドの呼び出し PreSave(); に書き換える。



次に，それぞれの演算子ボタンから Text プロパティを受け取って，種別を 1 文字分戻す TypeSet メソッドを一から作成する。default を設定しているのは，Text プロパティの設定ミスに対応するためである。

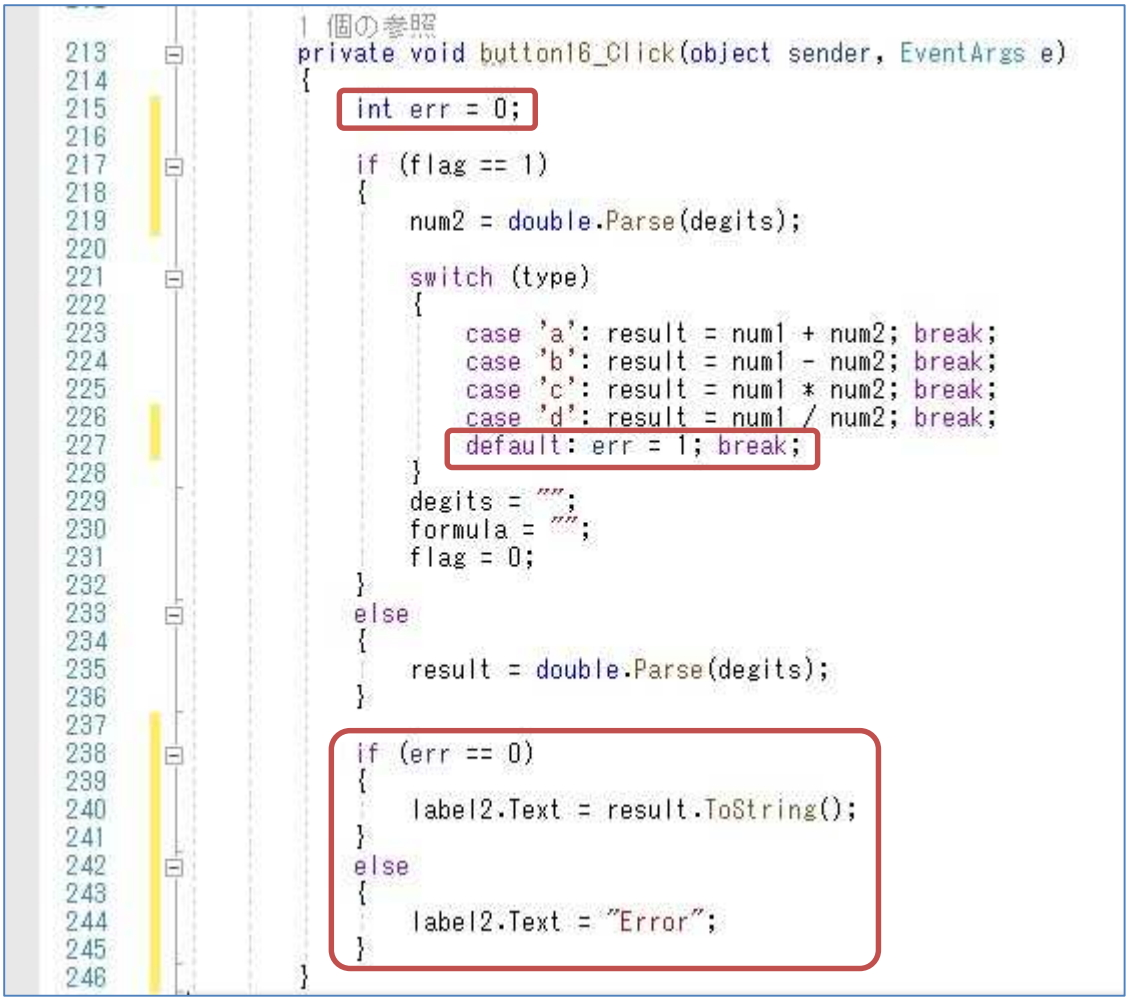


演算子ボタン button12～button15 のイベントハンドラで，変数 type への代入文の右辺を TypeSet メソッドの呼び出しに書き換える。実引数がそれぞれのボタンの Text プロパティであることに注意する。（図は button12 及び button13 のイベントハンドラ；button14 及び button15 のイベントハンドラも同様に書き換える。）

```
        1 個の参照
149     private void button12_Click(object sender, EventArgs e)
150     {
151         if (flag == 0)
152         {
153             PreSave();
154
155             num1 = double.Parse(degits);
156             formula += button12.Text;
157             label2.Text = "";
158             label1.Text = formula;
159             degits = "";
160             flag = 1;
161             type = TypeSet(button12.Text);
162         }
163     }
164
        1 個の参照
165     private void button13_Click(object sender, EventArgs e)
166     {
167         if (flag == 0)
168         {
169             PreSave();
170
171             num1 = double.Parse(degits);
172             formula += button13.Text;
173             label2.Text = "";
174             label1.Text = formula;
175             degits = "";
176             flag = 1;
177             type = TypeSet(button13.Text);
178         }
179     }
```

button16_Click では TypeSet メソッドの default により，type に 'e' がセットされた場合に備えて，エラー用のフラグ err を宣言し，初期値を 0 にセットする。Type を判定する switch 文に default を付け加え，'a' ～ 'd' 以外の値の場合には，err の値を 1 に書き換えてフラグを立てる。フラグが立っているかどうかによって，結果を表示するか『Error』の文字列を表示するかの場合分けをする if 文を挿入する。

（図は次のページ）



```
1 個の参照
private void button16_Click(object sender, EventArgs e)
{
    int err = 0;

    if (flag == 1)
    {
        num2 = double.Parse(degits);

        switch (type)
        {
            case 'a': result = num1 + num2; break;
            case 'b': result = num1 - num2; break;
            case 'c': result = num1 * num2; break;
            case 'd': result = num1 / num2; break;
            default: err = 1; break;
        }
        degits = "";
        formula = "";
        flag = 0;
    }
    else
    {
        result = double.Parse(degits);
    }

    if (err == 0)
    {
        label2.Text = result.ToString();
    }
    else
    {
        label2.Text = "Error";
    }
}
```

仕上げの処理の 1 つ目は，”0123” の様な入力をしてしまう場合への対処である。このままでも計算は可能であるが，”123” に直せる様にしたい。そのためには，先ず，button1 のイベントハンドラで，次の書き換えを行う。

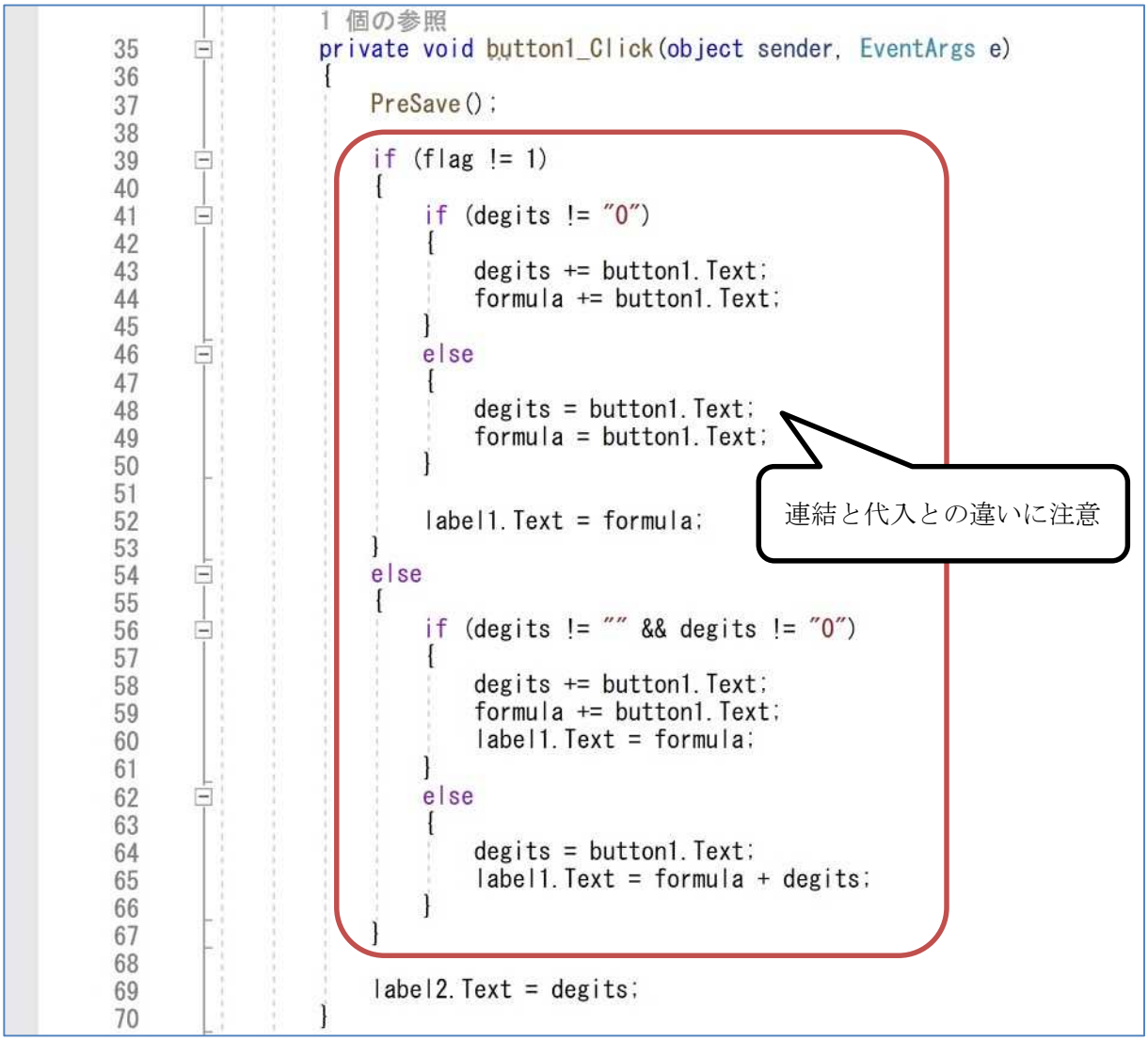
1) degits が num1 の入力の場合（flag の値は 0）：

degits の値が ”0” でなければこれまで通りボタンの Text プロパティの値を degits 及び formula に連結するが，degits の値が ”0” の場合には，ボタンの Text プロパティの値を degits 及び formula に代入し直す様に書き換える。

2) degits が num2 の入力の場合（flag の値は 1）：

演算子の入力の時点で degits の値は空の文字列になっている。その場合と，演算子の直後の入力が ”0” の場合には，ボタンの Text プロパティの値を degits に代入し直す。この場合には num1 の入力時と異なり，formula には num1 と演算子とが結合した文字列が格納されているので，その状態を保たなければならず，formula の値は変えないで，label1 の Text プロパティの値として formula と degits とを結合した文字列を設定している。また，degits の値が空の文字列でも”0”でもない場合には，ボタンの Text プロパティの値を degits 及び formula に連結する。

（図は次のページ）



＊ 上述の degits が num2 の入力の場合へのゼロ抑制の結果として，formula には演算子の直後に ”0” が結合されなくなるので，小数点がその後に入力された場合には ”0” を補ってやる必要が生じる。この様な副作用の解消は後程，小数点の扱いを改良する際に並行して行う。

次に，button2～button10 のイベントハンドラで degits の値が ”0” でなければこれまで通りボタンの Text プロパティの値を degits 及び formula に連結する。degits の値が ”0” の場合には，ボタンの Text プロパティの値を degits に代入し直す様に書き換えるが，formula の値は演算子の前後で扱いを変える。

（次のページの図は button2 のイベントハンドラ；button3～button10 のイベントハンドラも同様に書き換える。）

```
1 個の参照
80      private void button2_Click(object sender, EventArgs e)
81      {
82          PreSave();
83
84          if (degits != "0")
85          {
86              degits += button2.Text;
87              formula += button2.Text;
88          }
89          else
90          {
91              degits = button2.Text;
92
93              if (flag != 1)
94              {
95                  formula = button2.Text;
96              }
97              else
98              {
99                  formula += button2.Text;
100             }
101         }
102
103         label2.Text = degits;
104         label1.Text = formula;
105     }
```

仕上げの処理の 2 つ目は，小数点のドットが degits に格納された文字列の中に 2 個以上入らない様にするための対処である。そのために，先ず，フィールドに dotflag 及び predotf という int 型のフラグを宣言し，初期値を 0 としておく。

```
3 個の参照
13  public partial class Form1 : Form
14  {
15      string degits = "";
16      string formula = "";
17      double num1;
18      double num2;
19      double result;
20
21      int flag = 0;
22      char type;
23      string predeg = "";
24      string preform = "";
25      int preflag = 0;
26
27      int dotflag = 0;
28      int predotf = 0;
29
        1 個の参照
```

PreSave メソッドに predotf に処理を行う直前の dotflag の値を代入する文を付け加える。

```
15 個の参照
72      private void PreSave()
73      {
74          predeg = degits;
75          preform = formula;
76          preflag = flag;
77          predotf = dotflag;
78      }
79
```

button11 のイベントハンドラで，dotflag の値が 0 の場合のみこれまで記述されていた処理を行う様に if 文を用いて書き換え，フラグを立てる dotflag = 1; の代入文を付け加える。また，上述のゼロ抑制の副作用の解消も行う。

```
        1 個の参照
323     private void button11_Click(object sender, EventArgs e)
324     {
325         if (dotflag == 0)
326         {
327             PreSave();
328
329             degits += button11.Text;
330
331             if (flag == 1 && predeg == "0")
332             {
333                 formula += button1.Text + button11.Text;
334             }
335             else
336             {
337                 formula += button11.Text;
338             }
339
340             label2.Text = degits;
341             label1.Text = formula;
342             dotflag = 1;
343         }
344     }
```

button12～button15 のイベントハンドラで，dotflag の値を初期値に戻す代入文を付け加える。

（図は button12 のイベントハンドラ ； button13～button15 のイベントハンドラのイベントハンドラも同様に書き換える。）

```
        1 個の参照
346     private void button12_Click(object sender, EventArgs e)
347     {
348         if (flag == 0)
349         {
350             PreSave();
351
352             num1 = double.Parse(degits);
353             formula += button12.Text;
354             label2.Text = "";
355             label1.Text = formula;
356             degits = "";
357             flag = 1;
358             type = TypeSet(button12.Text);
359             dotflag = 0;
360         }
361     }
```

button16 のイベントハンドラで，dotflag の値を初期値に戻す代入文を付け加える。

（図は次のページ）

```
414         1 個の参照
            private void button16_Click(object sender, EventArgs e)
415         {
416             int err = 0;
417
418             if (flag == 1)
419             {
420                 num2 = double.Parse(degits);
421
422                 switch (type)
423                 {
424                     case 'a': result = num1 + num2; break;
425                     case 'b': result = num1 - num2; break;
426                     case 'c': result = num1 * num2; break;
427                     case 'd': result = num1 / num2; break;
428                     default: err = 1; break;
429                 }
430                 degits = "";
431                 formula = "";
432                 flag = 0;
433                 dotflag = 0;
434             }
435             else
436             {
437                 result = double.Parse(degits);
438             }
439
440             if (err == 0)
441             {
442                 label2.Text = result.ToString();
443             }
444             else
445             {
446                 label2.Text = "Error";
447             }
448         }
```

button17 のイベントハンドラで，predotf 及び dotflag の値を初期値に戻す代入文を付け加える。

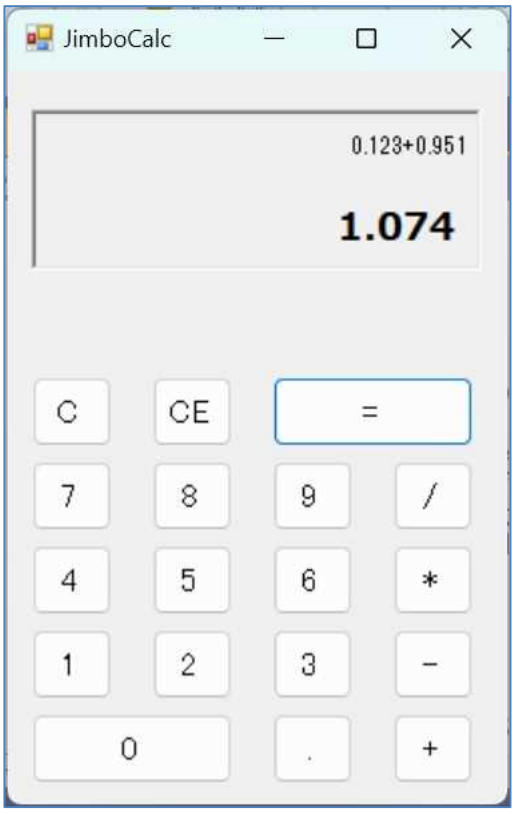
```
450         1 個の参照
            private void button17_Click(object sender, EventArgs e)
451         {
452             predeg = "";
453             preform = "";
454             preflag = 0;
455             flag = 0;
456             predotf = 0;
457             dotflag = 0;
458
459             degits = "";
460             formula = "";
461             label1.Text = "";
462             label2.Text = "";
463         }
```

button18 のイベントハンドラで，dotflag の値を直前の値に戻す代入文を付け加える。

（図は次のページ）

```
        1 個の参照
471     private void button18_Click(object sender, EventArgs e)
472     {
473         degits = predeg;
474         formula = preform;
475         flag = preflag;
476         dotflag = predotf;
477         label2.Text = degits;
478         label1.Text = formula;
479     }
```

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，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）教材通りの加算の実行結果のスクリーンショット第 6 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）質問を記述したファイル Prog2_Questions_6th.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。