```
using System;
using System.Windows.Forms;
namespace RCalc {
    public partial class MainForm : Form {
        private ICalculable    calc;
        public MainForm() {
            this.InitializeComponent();
            this.InitializeCalcButton(this.addButton, new CalcAdd());
            this.InitializeCalcButton(this.subButton, new CalcSub());
            this.InitializeCalcButton(this.mulButton, new CalcMul());
            this.InitializeCalcButton(this.divButton, new CalcDiv());
            this.InitializeNumericButtons(
                this.numericButton0, this.numericButton1, this.numericButton2, this.numericButton3,
                this.numericButton4, this.numericButton5, this.numericButton6, this.numericButton7,
                this.numericButton8, this.numericButton9);
            this.endButton.Click += (sender, e) => this.Close();
            this.backSpaceButton.Click += (sender, e) =>
                this.editBuffer = this.editBuffer.Length > 0 ? this.editBuffer.Substring(1) : this.editBuffer;
            this.clearEditButton.Click += (sender, e) => this.editBuffer = String.Empty;
            this.clearButton.Click += (sender, e) => this.Clear();
            this.eqaulButton.Click += (sender, e) => this.ExecuteCalc();
            this.Clear();
        }

        private void InitializeCalcButton(Button button, ICalculable calc) {
            button.Tag = calc;
            button.Click += (sender, e) => {
                this.ExecuteCalc();
                this.calc = (ICalculable)button.Tag;
            };
        }

        private void InitializeNumericButtons(params Button[] buttons) {
            for(var i = 0; i < buttons.Length; ++i) {
                buttons[i].Text    = i.ToString();
                buttons[i].Click += (sender, e) => this.editBuffer += ((Button)sender).Text;
            }
        }

        private void ExecuteCalc() {
            if (this.editBuffer == String.Empty)  return;
            this.calc.Value1 = this.register;
            this.calc.Value2 = this.editBuffer;
            this.calc.Execute();
            this.editBuffer = String.Empty;
            this.register = this.calc.Result;
        }

        private void Clear() {
            this.calc = new Calculate();
            this.register = this.editBuffer = String.Empty;
        }

        private string _editBuffer;
        private string editBuffer {
            get { return this._editBuffer; }
            set {
                this._editBuffer = value;
                this.SetDisplay(value);
            }
        }

        private string _register;
        private string register {
            get { return this._register; }
            set {
                this._register = value;
```

```
68                  this.SetDisplay(value);
69              }
70          }
71
72          private void SetDisplay(string value) { this.displayTextBox.Text = value == String.Empty ? "0" : value; }
73      }
74  }
75  ////////////////////////////////////////////////////////////////////////////////////////////////////////////////
76  using System;
77  namespace RCalc {
78      static class Extensions { // 拡張メソッドを使ってみた!!!
79          public static decimal ToDecimal(this char c) {
80              var r = 0m;
81              if (decimal.TryParse(c.ToString(), out r)) return r;
82              throw new ArgumentException();
83          }
84      }
85      interface ICalculable {// 自然数のみ計算対象、それ以外の結果なら Empty を返す
86          string   Value1 { get; set; }
87          string   Value2 { get; set; }
88          string   Result { get; }
89          void Clear();
90          void Execute();
91      }
92      class Calculate : ICalculable {
93          public Calculate() { this.Clear(); }
94
95          public void Clear()  { this.Result = this.Value1 =  this.Value2 = String.Empty; }
96
97          public string Value1 { get; set; } // 自動プロパティを使ってみた。
98          public string Value2 { get; set; } // インターフェイスと間違えやすいので注意！！
99          public string Result { get; protected set; }
100         public void Execute() { this.Result = this.Execute(this.Value1, this.Value2); }
101         public virtual string Execute(string value1, string value2) { return value1 == String.Empty ? value2 : value1; }
102         protected decimal GetOneDigit(string value, int digitIndex) {
103             if (digitIndex < 0 || digitIndex > value.Length - 1) return 0m;
104             return value[value.Length - (digitIndex + 1)].ToDecimal(); // 拡張メソッドのおかげ・・・
105         }
106         protected string Simplecalculation(string value1, string value2, Func<decimal, decimal, decimal, decimal> func){
107             var count = Math.Max(value1.Length, value2.Length);  // ラムダっちゃからデリゲートw
108             return this.Simplecalculation(
109                 count, (n, i) => func(n, this.GetOneDigit(value1, i), this.GetOneDigit(value2, i)));
110         }
111         protected string Simplecalculation(int count, Func<decimal, int, decimal> func) {
112             var r  = String.Empty;
113             var n = 0m;
114             for (var i = 0; i < count; ++i) {
115                 n = func(n, i);
116                 r =  (n % 10).ToString() + r;
117                 n = Math.Floor(n / 10);
118             }
119             if (n > 0) r = n.ToString() + r;
120             return r.TrimStart('0');
121         }
122         protected string GetZeroString(int value) { return String.Join("0", new String[value + 1]); }
123         protected int Compare(string value1, string value2) {
124             if (value1.Length < value2.Length) return -1;  // value1 < value2
125             if (value1.Length > value2.Length) return 1;   // value1 > value2
126             for(int i = 0; i < value1.Length; ++i) {       // value1 と value2 は 同じ桁数
127                 var d1 = value1[i].ToDecimal(); // 拡張メソッドのおかげ・・・
128                 var d2 = value2[i].ToDecimal();
129                 if (d1 < d2) return -1; // value1 < value2
130                 if (d1 > d2) return 1;  // value1 > value2
131             }
132             return 0;// value1 == value2
133         }
134     }
```

```
135     class CalcAdd : Calculate {
136         public override string Execute(string value1, string value2) {
137             return base.Simplecalculation(value1, value2, (n, d1, d2) => n + d1 + d2); // ラムダっちゃ
138         }
139     }
140     class CalcSub : Calculate {
141         public override string Execute(string value1, string value2) {
142             if (base.Compare(value1, value2) <= 0) return String.Empty;
143             return base.Simplecalculation(
144                 value1, value2,
145                 (n, d1, d2) => {
146                     var r = d1 - d2 - n;
147                     return r < 0 ? r + 20: r;
148                 });  // ラムダっちゃ
149         }
150     }
151     class CalcMul : Calculate {
152         public override string Execute(string value1, string value2) {
153             var r = String.Empty;
154             var add = new CalcAdd();
155             for (var i = 0; i < value1.Length; ++i) {
156                 var t = this.MultiplicationSingleValue(base.GetOneDigit(value1, i), value2);
157                 t += base.GetZeroString(i);
158                 r = add.Execute(r, t);
159             }
160             return r;
161         }
162         private string MultiplicationSingleValue(decimal value1, string value2) {
163             return base.Simplecalculation(value2.Length, (n, i) => n + value1 * base.GetOneDigit(value2, i));
164         }
165     }
166     class CalcDiv : Calculate {
167         public override string Execute(string value1, string value2) {
168             var r = String.Empty;
169             if (value2.TrimStart('0') == String.Empty) return r; // ゼロでは割れない
170             var add = new CalcAdd();
171             var subtract = new CalcSub();
172             var multiplication = new CalcMul();
173 
174             var v = value1;
175             while (base.Compare(v, value2) >= 0) {
176                 var l = this.GetCutLength(v, value2);
177                 var d = v.Substring(0, l);
178                 var t = this.Division(d, value2);
179                 r = add.Execute(r, t + this.GetZeroString(v.Length - l));
180                 v = subtract.Execute(d, multiplication.Execute(t, value2)) + v.Substring(l);
181             }
182             return r.TrimStart('0');
183         }
184         private int GetCutLength(string value1, string value2) {
185             var r = value2.Length;
186             while (base.Compare(value1.Substring(0, r), value2) < 0) ++r;
187             return r;
188         }
189         private string Division(string value1, string value2) {
190             var subtract = new CalcSub();
191             var r = value1;
192             var m = (new CalcMul()).Execute(r, value2);
193             while (base.Compare(m, value1) > 0) {
194                 r = subtract.Execute(r, "1");
195                 m = subtract.Execute(m, value2);
196             }
197             return r;
198         }
199     }
200 }
```