

```
1 #include "Interface.h"
2
3 using namespace System;
4 using namespace System::ComponentModel::Composition;
5 using namespace System::Collections::Generic;
6
7 namespace SimpleCalculator {
8
9     [Export(ICalculator::typeid)]
10    ref class Calculator : ICalculator {
11    private:
12        [ImportMany]
13        IEnumerable<System::Lazy<IOperation^, IOperationData^>>^ operations_;
14
15    public:
16        // Symbolが一致するIOperationを見つけ、実行する
17        virtual int Calculate(int left, System::String^ operation, int right) {
18            for each (Lazy<IOperation^, IOperationData^>^ item in operations_) {
19                if (item->Metadata->Symbol == operation)
20                    return item->Value->Operate(left, right);
21            }
22            throw gcnew NotSupportedException(operation);
23        }
24
25        // Symbolの列挙を返す
26        virtual IEnumerable<System::String^>^ Symbols() {
27            auto result = gcnew List<String^>();
28            for each (Lazy<IOperation^, IOperationData^>^ item in operations_) {
29                result->Add(item->Metadata->Symbol);
30            }
31            return result;
32        }
33    };
34
35 }
36
37 }
```

```
1 #include "CalculatorForm.h"
2
3 using namespace System;
4 using namespace System::ComponentModel::Composition;
5 using namespace System::ComponentModel::Composition::Hosting;
6
7 namespace SimpleCalculator {
8
9     CalculatorForm::CalculatorForm() {
10         InitializeComponent();
11
12         // Import/Export カタログをつくる
13         auto catalog = gcnew AggregateCatalog();
14         // まずは自分自身のアセンブリから
15         catalog->Catalogs->Add(gcnew AssemblyCatalog(CalculatorForm::typeid->Assembly));
16
17         // そして自分自身の置かれたディレクトリから見つけてくる
18         String^ myLocation = System::IO::Path::GetDirectoryName(
19             System::Reflection::MethodInfo::GetCurrentMethod()
20             ->DeclaringType->Assembly->Location);
21         catalog->Catalogs->Add(gcnew DirectoryCatalog(myLocation));
22
23         // カタログから作られたコンテナを基にImport/Exportを結びつける
24         AttributedModelServices::ComposeParts(gcnew CompositionContainer(catalog), this);
25
26         // 得られた演算子(Symbol)をComboBoxに追加する
27         for each (String^ symbol in calculator->Symbols()) {
28             cbxOpr->Items->Add(symbol);
29         }
30         cbxOpr->SelectedIndex = 0;
31     }
32
33     CalculatorForm::~CalculatorForm() {
34         if (components) {
35             delete components;
36         }
37     }
38
39     System::Void CalculatorForm::btnExec_Click(System::Object^ sender, System::EventArgs^ e) {
40         Int32 left;
41         Int32 right;
42         // フォームから 左辺/右辺/演算子を取り出し、計算して結果を表示する
43         if (Int32::TryParse(tbxLeft->Text, left) && Int32::TryParse(tbxRight->Text, right)) {
44             try {
45                 int result = calculator->Calculate(left, cbxOpr->SelectedItem->ToString(), right);
46                 lblResult->Text = result.ToString();
47             } catch (Exception^ ) {
48                 lblResult->Text = L"error";
49             }
50         } else {
51             lblResult->Text = L"?";
52         }
53     }
54 }
55
56 }
```

```

1  #pragma once
2  #include "Interface.h"
3
4  namespace SimpleCalculator {
5
6      public ref class CalculatorForm : public System::Windows::Forms::Form
7      {
8      public:
9          CalculatorForm();
10
11      protected:
12          ~CalculatorForm();
13
14      private: System::Windows::Forms::TextBox^ tbxLeft;
15      private: System::Windows::Forms::ComboBox^ cbxOpr;
16      private: System::Windows::Forms::TextBox^ tbxRight;
17      private: System::Windows::Forms::Button^ btnExec;
18      private: System::Windows::Forms::Label^ lblResult;
19
20      private:
21          System::ComponentModel::Container ^components;
22
23      #pragma region Windows Form Designer generated code
24      /// <summary>
25      /// デザイナー サポートに必要なメソッドです。このメソッドの内容を
26      /// コード エディターで変更しないでください。
27      /// </summary>
28      void InitializeComponent(void)
29      {
30          System::Windows::Forms::Label^ label2;
31          System::Windows::Forms::Label^ label3;
32          System::Windows::Forms::Label^ label4;
33          this->tbxLeft = (gcnew System::Windows::Forms::TextBox());
34          this->cbxOpr = (gcnew System::Windows::Forms::ComboBox());
35          this->tbxRight = (gcnew System::Windows::Forms::TextBox());
36          this->btnExec = (gcnew System::Windows::Forms::Button());
37          this->lblResult = (gcnew System::Windows::Forms::Label());
38          label2 = (gcnew System::Windows::Forms::Label());
39          label3 = (gcnew System::Windows::Forms::Label());
40          label4 = (gcnew System::Windows::Forms::Label());
41          this->SuspendLayout();
42          //
43          // label2
44          //
45          label2->AutoSize = true;
46          label2->Location = System::Drawing::Point(12, 9);
47          label2->Name = L"label2";
48          label2->Size = System::Drawing::Size(22, 12);
49          label2->TabIndex = 5;
50          label2->Text = L"left";
51          //
52          // label3
53          //
54          label3->AutoSize = true;
55          label3->Location = System::Drawing::Point(85, 9);
56          label3->Name = L"label3";
57          label3->Size = System::Drawing::Size(23, 12);
58          label3->TabIndex = 6;
59          label3->Text = L"opr.";
60          //
61          // label4
62          //
63          label4->AutoSize = true;
64          label4->Location = System::Drawing::Point(167, 9);
65          label4->Name = L"label4";
66          label4->Size = System::Drawing::Size(28, 12);
67          label4->TabIndex = 7;
68          label4->Text = L"right";
69          //
70          // tbxLeft
71          //
72          this->tbxLeft->Font = (gcnew System::Drawing::Font(L"MS UI Gothic", 18,
73              System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
74              static_cast<System::Byte>(128)));
75          this->tbxLeft->Location = System::Drawing::Point(12, 32);
76          this->tbxLeft->Name = L"tbxLeft";
77          this->tbxLeft->Size = System::Drawing::Size(69, 31);
78          this->tbxLeft->TabIndex = 0;
79          //
80          // cbxOpr
81          //
82          this->cbxOpr->DropDownStyle = System::Windows::Forms::ComboBoxStyle::DropDownList;
83          this->cbxOpr->Font = (gcnew System::Drawing::Font(L"MS UI Gothic", 18,

```

```

83     System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
84     static_cast<System::Byte>(128)));
85     this->cbxOpr->FormattingEnabled = true;
86     this->cbxOpr->Location = System::Drawing::Point(87, 32);
87     this->cbxOpr->Name = L"cbxOpr";
88     this->cbxOpr->Size = System::Drawing::Size(76, 32);
89     this->cbxOpr->Sorted = true;
90     this->cbxOpr->TabIndex = 1;
91     //
92     // tbxRight
93     this->tbxRight->Font = (gcnew System::Drawing::Font(L"MS UI Gothic", 18,
94     System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
95     static_cast<System::Byte>(128)));
96     this->tbxRight->Location = System::Drawing::Point(169, 33);
97     this->tbxRight->Name = L"tbxRight";
98     this->tbxRight->Size = System::Drawing::Size(69, 31);
99     this->tbxRight->TabIndex = 2;
100    //
101    // btnExec
102    this->btnExec->Font = (gcnew System::Drawing::Font(L"MS UI Gothic", 18,
103    System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
104    static_cast<System::Byte>(128)));
105    this->btnExec->Location = System::Drawing::Point(244, 30);
106    this->btnExec->Name = L"btnExec";
107    this->btnExec->Size = System::Drawing::Size(29, 33);
108    this->btnExec->TabIndex = 3;
109    this->btnExec->Text = L"";
110    this->btnExec->UseVisualStyleBackColor = true;
111    this->btnExec->Click += gcnew System::EventHandler(this, &CalculatorForm::btnExec_Click);
112    //
113    // lblResult
114    this->lblResult->BorderStyle = System::Windows::Forms::BorderStyle::FixedSingle;
115    this->lblResult->Font = (gcnew System::Drawing::Font(L"MS UI Gothic", 48,
116    System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
117    static_cast<System::Byte>(128)));
118    this->lblResult->Location = System::Drawing::Point(12, 77);
119    this->lblResult->Name = L"lblResult";
120    this->lblResult->Size = System::Drawing::Size(259, 71);
121    this->lblResult->TabIndex = 4;
122    this->lblResult->Text = L"0";
123    this->lblResult->TextAlign = System::Drawing::ContentAlignment::MiddleCenter;
124    //
125    // CalculatorForm
126    this->AutoScaleDimensions = System::Drawing::SizeF(6, 12);
127    this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
128    this->ClientSize = System::Drawing::Size(283, 168);
129    this->Controls->Add(label4);
130    this->Controls->Add(label3);
131    this->Controls->Add(label2);
132    this->Controls->Add(this->lblResult);
133    this->Controls->Add(this->btnExec);
134    this->Controls->Add(this->tbxRight);
135    this->Controls->Add(this->cbxOpr);
136    this->Controls->Add(this->tbxLeft);
137    this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::FixedSingle;
138    this->MaximizeBox = false;
139    this->Name = L"CalculatorForm";
140    this->Text = L"Calculator";
141    this->ResumeLayout(false);
142    this->PerformLayout();
143    }
144    #pragma endregion
145    private:
146    // '=' ボタン・クリックのハンドラ
147    System::Void btnExec_Click(System::Object^ sender, System::EventArgs^ e);
148    [System::ComponentModel::Composition::Import(ICalculator::typeid)]
149    ICalculator^ calculator_;
150    };
151    };
152    };
153    };
154    };
155    };
156    };

```

```
1 #ifndef INTERFACE_H_
2 #define INTERFACE_H_
3
4 namespace SimpleCalculator {
5
6     public interface class IOperation {
7         | int Operate(int left, int right);
8     };
9
10    public interface class IOperationData {
11        | property System::String^ Symbol { System::String^ get(); }
12    };
13
14    public interface class ICalculator {
15        | int Calculate(int left, System::String^ opr, int right);
16        | System::Collections::Generic::IEnumerable<System::String^> Symbols();
17    };
18
19 }
20
21 #endif
22
```

```
1  #include "Interface.h"
2
3  using namespace System::ComponentModel::Composition;
4
5  namespace SimpleCalculator {
6
7      [Export(IOperation::typeid)]
8      [ExportMetadata(L"Symbol", L"+")]
9      ref class Add : IOperation {
10     public:
11         virtual int Operate(int left, int right) {
12             return left + right;
13         }
14     };
15
16     [Export(IOperation::typeid)]
17     [ExportMetadata(L"Symbol", L"-")]
18     ref class Subtract : IOperation {
19     public:
20         virtual int Operate(int left, int right) {
21             return left - right;
22         }
23     };
24
25 }
26
```

```
1 #include "CalculatorForm.h"
2
3 using namespace System;
4
5 [STAThread]
6 int main(array<String^>^ args) {
7     using namespace System::Windows::Forms;
8     Application::EnableVisualStyles();
9     Application::SetCompatibleTextRenderingDefault(false);
10    Application::Run(gcnew SimpleCalculator::CalculatorForm());
11    return 0;
12 }
13
14
```

```
1 #include "stdafx.h"
2
3 namespace ExtendedOperations {
4
5     using namespace System::ComponentModel::Composition;
6     using namespace SimpleCalculator;
7
8     [Export(IOperation::typeid)]
9     [ExportMetadata(L"Symbol", L"*")]
10    ref class Multiple : IOperation {
11    public:
12        virtual int Operate(int left, int right) {
13            return left * right;
14        }
15    };
16
17    [Export(IOperation::typeid)]
18    [ExportMetadata(L"Symbol", L"/")]
19    ref class Subtract : IOperation {
20    public:
21        virtual int Operate(int left, int right) {
22            return left / right;
23        }
24    };
25
26 }
27
```



```
1 using System.ComponentModel.Composition;
2 using SimpleCalculator;
3
4 namespace ExtendedOperations {
5
6     [Export(typeof(IOperation))]
7     [ExportMetadata("Symbol", "%")]
8     class Modulus : IOperation {
9         public int Operate(int left, int right) {
10             return left % right;
11         }
12     };
13
14     [Export(typeof(IOperation))]
15     [ExportMetadata("Symbol", "MIN")]
16     class Minimum : IOperation {
17         public int Operate(int left, int right) {
18             return left < right ? left : right;
19         }
20     };
21
22     [Export(typeof(IOperation))]
23     [ExportMetadata("Symbol", "MAX")]
24     class Maximum : IOperation {
25         public int Operate(int left, int right) {
26             return left > right ? left : right;
27         }
28     };
29 }
30
```