```cpp
#include "Interface.h"

using namespace System;
using namespace System::ComponentModel::Composition;
using namespace System::Collections::Generic;

namespace SimpleCalculator {

  [Export(ICalculator::typeid)]
  ref class Calculator : ICalculator {
  private:
    [ImportMany]
    IEnumerable<System::Lazy<IOperation^, IOperationData^>^>^ operations_;

  public:
    // Symbolが一致するIOperationを見つけ、実行する
    virtual int Calculate(int left, System::String^ operation, int right) {
      for each (Lazy<IOperation^, IOperationData^>^ item in operations_) {
        if ( item->Metadata->Symbol == operation )
          return item->Value->Operate(left, right);
      }
      throw gcnew NotSupportedException(operation);
    }

    // Symbolの列挙を返す
    virtual IEnumerable<System::String^>^ Symbols() {
      auto result = gcnew List<String^>();
      for each (Lazy<IOperation^, IOperationData^>^ item in operations_) {
        result->Add(item->Metadata->Symbol);
      }
      return result;
    }

  };

}
```

```
#include "CalculatorForm.h"

using namespace System;
using namespace System::ComponentModel::Composition;
using namespace System::ComponentModel::Composition::Hosting;

namespace SimpleCalculator {

  CalculatorForm::CalculatorForm() {
    InitializeComponent();

    // Import/Export カタログをつくる
    auto catalog = gcnew AggregateCatalog();
    // まずは自分自身のアセンブリから
    catalog->Catalogs->Add(gcnew AssemblyCatalog(CalculatorForm::typeid->Assembly));

    // 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
    String^ myLocation = System::IO::Path::GetDirectoryName(
                           System::Reflection::MethodBase::GetCurrentMethod()
                           ->DeclaringType->Assembly->Location);
    catalog->Catalogs->Add(gcnew DirectoryCatalog(myLocation));

    // 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
    AttributedModelServices::ComposeParts(gcnew CompositionContainer(catalog), this);

    // 得られた演算子(Symbol)をComboBoxに追加する
    for each ( String^ symbol in calculator_->Symbols() ) {
      cbxOpr->Items->Add(symbol);
    }
    cbxOpr->SelectedIndex = 0;
  }

  CalculatorForm::~CalculatorForm() {
    if (components) {
      delete components;
    }
  }

  System::Void CalculatorForm::btnExec_Click(System::Object^  sender, System::EventArgs^  e) {
    Int32 left;
    Int32 right;
    // 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
    if ( Int32::TryParse(tbxLeft->Text, left) && Int32::TryParse(tbxRight->Text, right) ) {
      try {
        int result = calculator_->Calculate(left, cbxOpr->SelectedItem->ToString(), right);
        lblResult->Text = result.ToString();
      } catch ( Exception^  ) {
        lblResult->Text = L"error";
      }
    } else {
      lblResult->Text = L"?";
    }
  }

}
```

```
#pragma once
#include "Interface.h"

namespace SimpleCalculator {

  public ref class CalculatorForm : public System::Windows::Forms::Form
  {
  public:
    CalculatorForm();

  protected:
    ~CalculatorForm();

  private: System::Windows::Forms::TextBox^  tbxLeft;
  private: System::Windows::Forms::ComboBox^  cbxOpr;
  private: System::Windows::Forms::TextBox^  tbxRight;
  private: System::Windows::Forms::Button^  btnExec;
  private: System::Windows::Forms::Label^  lblResult;

  private:
    System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
    /// <summary>
    /// 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
    /// コード エディターで変更しないでください。
    /// </summary>
    void InitializeComponent(void)
    {
      System::Windows::Forms::Label^  label2;
      System::Windows::Forms::Label^  label3;
      System::Windows::Forms::Label^  label4;
      this->tbxLeft = (gcnew System::Windows::Forms::TextBox());
      this->cbxOpr = (gcnew System::Windows::Forms::ComboBox());
      this->tbxRight = (gcnew System::Windows::Forms::TextBox());
      this->btnExec = (gcnew System::Windows::Forms::Button());
      this->lblResult = (gcnew System::Windows::Forms::Label());
      label2 = (gcnew System::Windows::Forms::Label());
      label3 = (gcnew System::Windows::Forms::Label());
      label4 = (gcnew System::Windows::Forms::Label());
      this->SuspendLayout();
      //
      // label2
      //
      label2->AutoSize = true;
      label2->Location = System::Drawing::Point(12, 9);
      label2->Name = L"label2";
      label2->Size = System::Drawing::Size(22, 12);
      label2->TabIndex = 5;
      label2->Text = L"left";
      //
      // label3
      //
      label3->AutoSize = true;
      label3->Location = System::Drawing::Point(85, 9);
      label3->Name = L"label3";
      label3->Size = System::Drawing::Size(23, 12);
      label3->TabIndex = 6;
      label3->Text = L"opr.";
      //
      // label4
      //
      label4->AutoSize = true;
      label4->Location = System::Drawing::Point(167, 9);
      label4->Name = L"label4";
      label4->Size = System::Drawing::Size(28, 12);
      label4->TabIndex = 7;
      label4->Text = L"right";
      //
      // tbxLeft
      //
      this->tbxLeft->Font = (gcnew System::Drawing::Font(L"MS UI Gothic", 18, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
        static_cast<System::Byte>(128)));
      this->tbxLeft->Location = System::Drawing::Point(12, 32);
      this->tbxLeft->Name = L"tbxLeft";
      this->tbxLeft->Size = System::Drawing::Size(69, 31);
      this->tbxLeft->TabIndex = 0;
      //
      // cbxOpr
      //
      this->cbxOpr->DropDownStyle = System::Windows::Forms::ComboBoxStyle::DropDownList;
      this->cbxOpr->Font = (gcnew System::Drawing::Font(L"MS UI Gothic", 18,
       System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
        static_cast<System::Byte>(128)));
      this->cbxOpr->FormattingEnabled = true;
      this->cbxOpr->Location = System::Drawing::Point(87, 32);
      this->cbxOpr->Name = L"cbxOpr";
      this->cbxOpr->Size = System::Drawing::Size(76, 32);
      this->cbxOpr->Sorted = true;
      this->cbxOpr->TabIndex = 1;
      //
      // tbxRight
      //
      this->tbxRight->Font = (gcnew System::Drawing::Font(L"MS UI Gothic", 18,
       System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
        static_cast<System::Byte>(128)));
      this->tbxRight->Location = System::Drawing::Point(169, 33);
      this->tbxRight->Name = L"tbxRight";
      this->tbxRight->Size = System::Drawing::Size(69, 31);
      this->tbxRight->TabIndex = 2;
      //
      // btnExec
      //
      this->btnExec->Font = (gcnew System::Drawing::Font(L"MS UI Gothic", 18,
       System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
        static_cast<System::Byte>(128)));
      this->btnExec->Location = System::Drawing::Point(244, 30);
      this->btnExec->Name = L"btnExec";
      this->btnExec->Size = System::Drawing::Size(29, 33);
      this->btnExec->TabIndex = 3;
      this->btnExec->Text = L"=";
      this->btnExec->UseVisualStyleBackColor = true;
      this->btnExec->Click += gcnew System::EventHandler(this, &CalculatorForm::btnExec_Click);
      //
      // lblResult
      //
      this->lblResult->BorderStyle = System::Windows::Forms::BorderStyle::FixedSingle;
      this->lblResult->Font = (gcnew System::Drawing::Font(L"MS UI Gothic", 48,
       System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
        static_cast<System::Byte>(128)));
      this->lblResult->Location = System::Drawing::Point(12, 77);
      this->lblResult->Name = L"lblResult";
      this->lblResult->Size = System::Drawing::Size(259, 71);
      this->lblResult->TabIndex = 4;
      this->lblResult->Text = L"0";
      this->lblResult->TextAlign = System::Drawing::ContentAlignment::MiddleCenter;
      //
      // CalculatorForm
      //
      this->AutoScaleDimensions = System::Drawing::SizeF(6, 12);
      this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
      this->ClientSize = System::Drawing::Size(283, 168);
      this->Controls->Add(label4);
      this->Controls->Add(label3);
      this->Controls->Add(label2);
      this->Controls->Add(this->lblResult);
      this->Controls->Add(this->btnExec);
      this->Controls->Add(this->tbxRight);
      this->Controls->Add(this->cbxOpr);
      this->Controls->Add(this->tbxLeft);
      this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::FixedSingle;
      this->MaximizeBox = false;
      this->Name = L"CalculatorForm";
      this->Text = L"Calculator";
      this->ResumeLayout(false);
      this->PerformLayout();

  }
#pragma endregion

  private:
  // '='ボタン・クリックのハンドラ
    System::Void btnExec_Click(System::Object^  sender, System::EventArgs^  e);

    [System::ComponentModel::Composition::Import(ICalculator::typeid)]
    ICalculator^ calculator_;
  };

}

```

```
#ifndef INTERFACE_H_
#define INTERFACE_H_

namespace SimpleCalculator {

  public interface class IOperation {
    int Operate(int left, int right);
  };

  public interface class IOperationData {
    property System::String^ Symbol { System::String^ get(); }
  };

  public interface class ICalculator {
    int Calculate(int left, System::String^ opr, int right);
    System::Collections::Generic::IEnumerable<System::String^>^ Symbols();
  };

}

#endif

```

```
#include "Interface.h"

using namespace System::ComponentModel::Composition;

namespace SimpleCalculator {

  [Export(IOperation::typeid)]
  [ExportMetadata(L"Symbol", L"+")]
  ref class Add : IOperation {
  public:
    virtual int Operate(int left, int right) {
      return left + right;
    }
  };

  [Export(IOperation::typeid)]
  [ExportMetadata(L"Symbol", L"-")]
  ref class Subtract : IOperation {
  public:
    virtual int Operate(int left, int right) {
      return left - right;
    }
  };

}

```

```
#include "CalculatorForm.h"

using namespace System;

[STAThread]
int main(array<String^>^ args) {
  using namespace System::Windows::Forms;
  Application::EnableVisualStyles();
  Application::SetCompatibleTextRenderingDefault(false);
  Application::Run(gcnew SimpleCalculator::CalculatorForm());
  return 0;
}


```

```
#include "stdafx.h"

namespace ExtendedOperations {

  using namespace System::ComponentModel::Composition;
  using namespace SimpleCalculator;

  [Export(IOperation::typeid)]
  [ExportMetadata(L"Symbol", L"*")]
  ref class Multiple : IOperation {
  public:
    virtual int Operate(int left, int right) {
      return left * right;
    }
  };

  [Export(IOperation::typeid)]
  [ExportMetadata(L"Symbol", L"/")]
  ref class Subtract : IOperation {
  public:
    virtual int Operate(int left, int right) {
      return left / right;
    }
  };

}

```

```csharp
using System.ComponentModel.Composition;
using SimpleCalculator;

namespace ExtendedOperations {

  [Export(typeof(IOperation))]
  [ExportMetadata("Symbol", "%")]
  class Modulus : IOperation {
    public int Operate(int left, int right) {
      return left % right;
    }
  };

  [Export(typeof(IOperation))]
  [ExportMetadata("Symbol", "MIN")]
  class Minimum : IOperation {
    public int Operate(int left, int right) {
      return left < right ? left : right;
    }
  };

  [Export(typeof(IOperation))]
  [ExportMetadata("Symbol", "MAX")]
  class Maxumum : IOperation {
    public int Operate(int left, int right) {
      return left > right ? left : right;
    }
  };
}
```