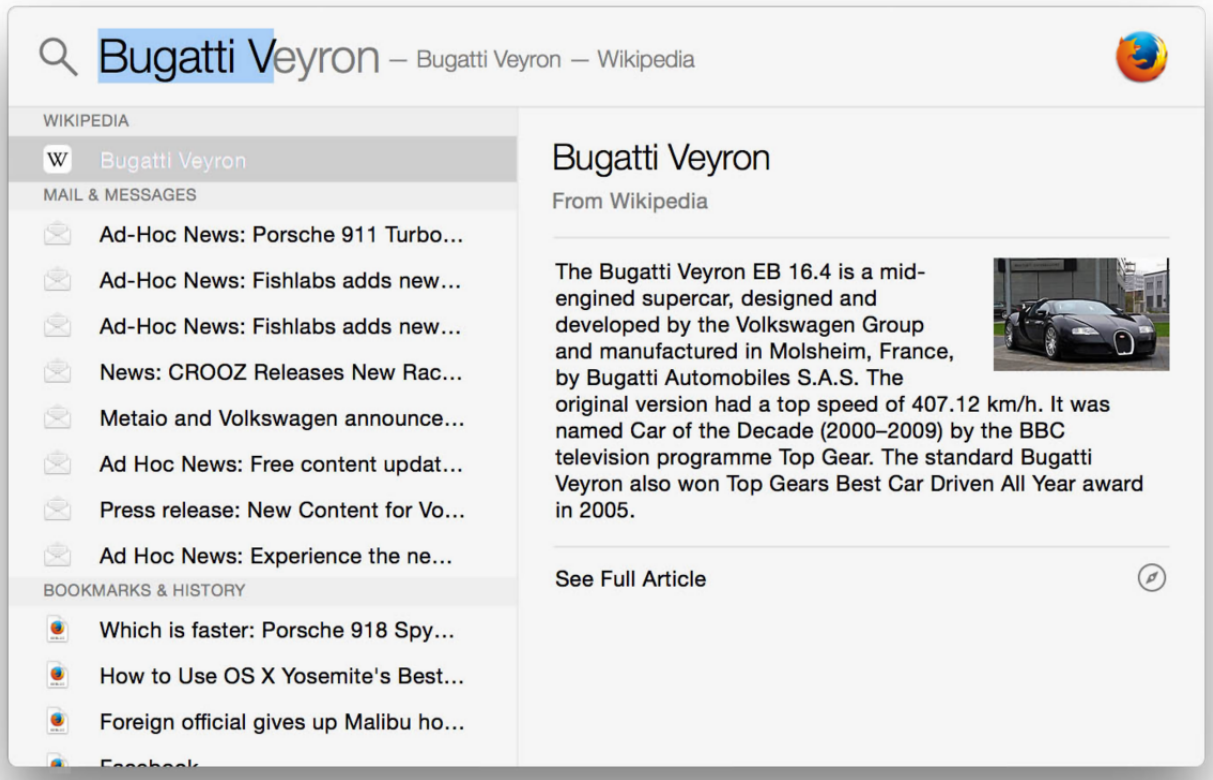
Windows Spotlight

Homework 3

101590306 郭鎧瑋
101590330 劉至峻



Requirement Document

1. Change History

Version	Author	Description of Version	Data Completed
0.1	郭鎧瑋、劉至峻	Problem Statement	2016/03/04
0.2	郭鎧瑋、劉至峻	System Context Diagram Summary of System Features Use Case Diagram Use Cases Non-function Requirement and Constrains Glossary Software Environment	2016/03/17
0.3	郭鎧瑋、劉至峻	Domain Model Add associations Add attributes	2016/03/31
0.4	郭鎧瑋、劉至峻	Logical Architecture Use-Case Realizations with GRASP Patterns Design Class Model	2016/04/27
0.5	郭鎧瑋、劉至峻	Implementation Class Model Programming Unit Testing	2016/05/04

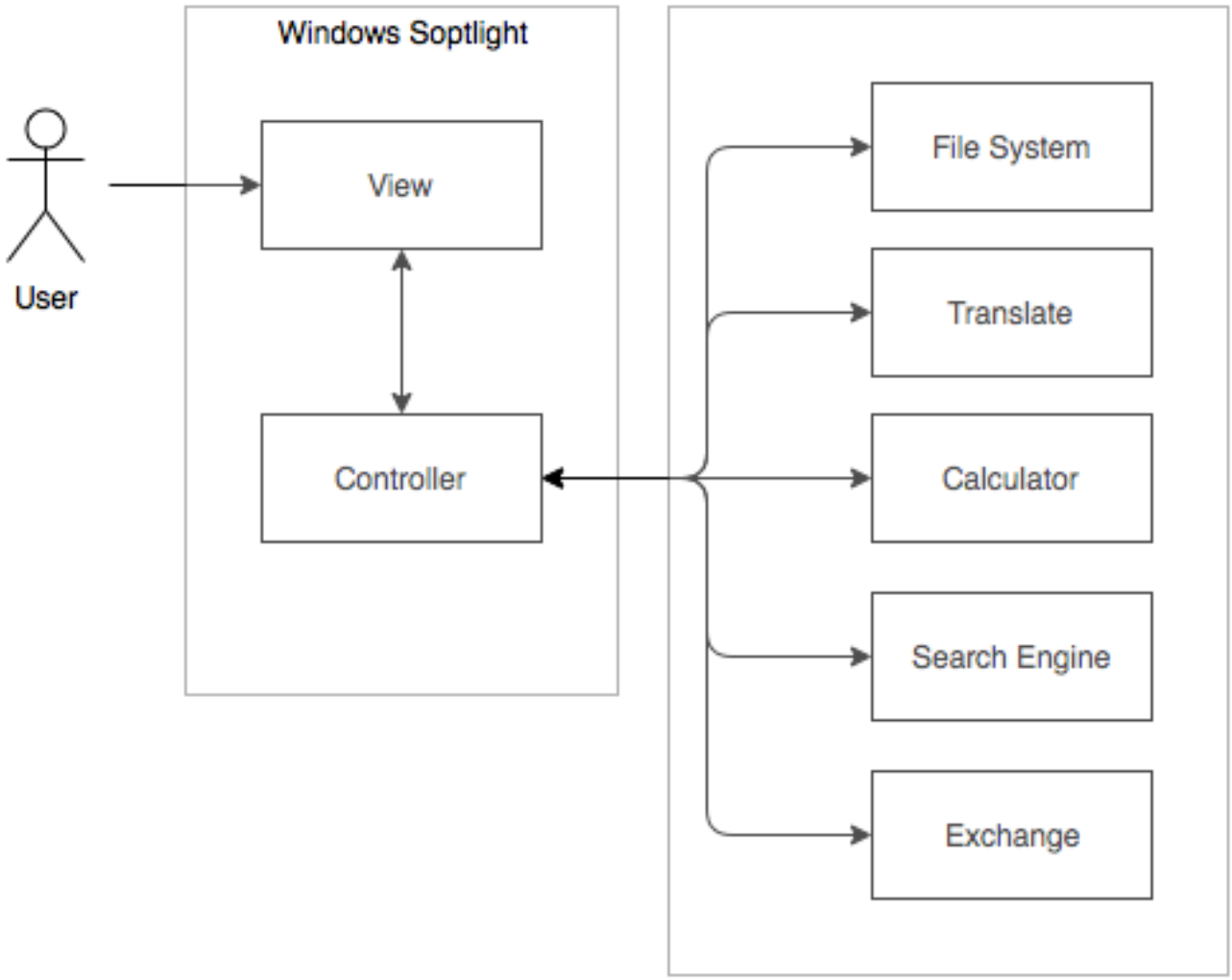
2. Problem Statement

Windows spotlight 的目的是為了提供windows使用者快速的尋找電腦中各式各樣的文件，包括應用程式、資料夾、文件、圖檔等，此外也可以做到匯率轉換、數字運算、字典查詢、搜尋網頁，提供windows使用者如同使用mac spotlight一樣的良好使用者體驗。

Windows在搜尋檔案時需要花費較多的時間，且透過開始快速開啟應用程式時，有些應用程式並不會顯示在上面，導致使用上的不便，因此我們希望提供一個程式可以讓使用者在想開啟應用程式，能用簡易的快速鍵呼叫，輸入關鍵字後即可快速的開啟。除了搜尋檔案的主要功能之外，我們在另外加入數字運算，字典查詢，匯率換算等功能，提供使用者更多的功能，提升使用的便利性。

Windows spotlight主要以文字方塊的方式呈現，使用者可以在文字方塊中輸入關鍵字，spotlight解讀關鍵字後，會在文字方塊下方產生查詢、翻譯或計算結果。

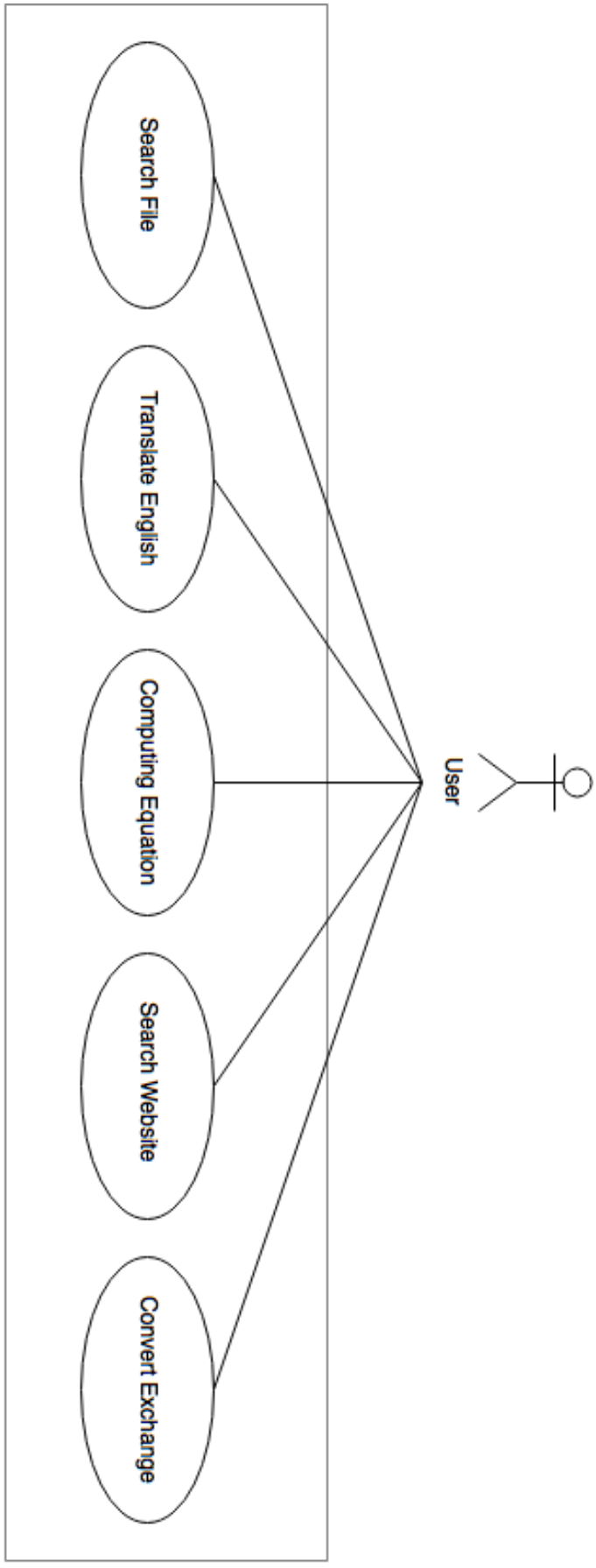
3. System Context Diagram



4. Summary of System Features

Feature ID	Description
FEA-01	提供使用者快速搜尋、開啟檔案的功能
FEA-02	提供使用者翻譯的功能
FEA-03	提供使用者計算機的功能
FEA-04	提供使用者匯率轉換的功能
FEA-05	提供使用者搜尋網頁的功能

5. Use Case Diagram



6. Use Cases

Use Case ID	Use Case Name
UC-01	Search File (搜尋檔案)
UC-02	Translate English (翻譯)
UC-03	Computing Equation (計算公式)
UC-04	Search Website (搜尋網頁)
UC-05	Convert Exchange (轉換匯率)

Use Case ID	UC-01
Use Case Name	Search File (搜尋檔案)
Scope	Windows Spotlight
Level	User goal
Primary Actor	User
Stakeholders & Interests	User: 輸入檔案關鍵字後可以獲得含有關鍵字的檔案列表，並可以開啟檔案
Precondition	1. 使用者開啟Windows Spotlight
Postcondition	列出含有關鍵字的檔案以及應用程式
Main Success Scenario	1. 使用者透過快捷鍵呼叫系統 2. 系統顯示輸入框，且游標停在輸入框 3. 使用者輸入檔案名稱 4. 系統依據使用者的輸入，將找到的檔案，依照符合程度，由高到低列出 5. 使用者可從清單中選擇開啟哪個檔案
Extensions	4a. 若找不到使用者輸入的檔案 1. 顯示沒有結果 4b. 若有複數檔案關聯度相同 1. 依照使用者開啟檔案的次數決定優先順序 5a. 開啟檔案: 1. 關聯度最高的檔案會自動被選擇 2. 使用者可使用上下鍵在清單中選擇欲開啟的檔案 3. 使用者按下Enter後，將開啟選擇的檔案
Speciel Requirement	1. 操作介面必須要很容易操作 2. 搜尋的速度必須要快
Technology And Data Variations List	None

Frequency of Occurrence	經常發生(當使用者需要開啟檔案時)
Open Issues	None

Use Case ID	UC-02
Use Case Name	Translate English (翻譯英文)
Scope	Windows Spotlight
Level	User goal
Primary Actor	User
Stakeholders & Interests	User: 輸入英文後 得到 中文翻譯
Precondition	1. 使用者開啟Windows Spotlight 2. 輸入語言須為英文 3. 需連上網路
Postcondition	顯示中文翻譯
Main Success Scenario	1. 使用者開啟Windows Spotlight 2. 開啟後游標停在輸入筐 3. 使用者輸入英文 4. 清單中出現翻譯選項 5. 使用者選擇翻譯選項後顯示翻譯
Extensions	1a. 無法開啟Windows Spotlight 1. 無任何反應 2a. 游標未停在輸入框 1. 使用者自行點擊到輸入筐 3a. 輸入語言非英文 1. 之後回傳內容無法翻譯成正確中文 5a. 翻譯失敗時 1. 告知使用者翻譯失敗 5b. 未連線到網路時 1. 清單中翻譯選項出現未連到網路的提醒 5c. 翻譯系統無回應時 1. 告知使用者系統無回應，請稍候再試
Speciel Requirement	1. 原本未連到網路，翻譯選項後又重新連到網路 ，當使用者選擇翻譯選項，會再傳送翻譯需求到網路上的翻譯系統並顯示回傳結果。
Technology And Data Variations List	None

Frequency of Occurrence	時常發生 (當使用者要翻譯時)
Open Issues	None

Use Case ID	UC-03
Use Case Name	Computing Equation (計算公式)
Scope	Windows Spotlight
Level	User goal
Primary Actor	User
Stakeholders & Interests	User: 輸入數字及運算符號後 得到結果
Precondition	1. 使用者開啟Windows Spotlight 2. 需輸入數學公式
Postcondition	顯示運算的結果
Main Success Scenario	1. 使用者開啟Windows Spotlight 2. 開啟後游標停在輸入框 3. 使用者輸入數學式子 4. 顯示運算結果
Extensions	1a. 無法開啟Windows Spotlight 1. 無任何反應 2a. 游標未停在輸入框 1. 使用者自行點擊到輸入筐 3a. 輸入非四則運算式 1. 不做四則運算 4a. 運算出結果 1. 將結果顯示出來
Speciel Requirement	1. 可以判斷sqrt(), sin(), cos(), tan() 等數學函式
Technology And Data Variations List	None
Frequency of Occurrence	時常發生 (當使用者要計算算式)
Open Issues	None

Use Case ID	UC-04
Use Case Name	Search Website (搜尋網頁)
Scope	Windows Spotlight

Level	User goal
Primary Actor	User
Stakeholders & Interests	User: 輸入文字
Precondition	1. 使用者開啟Windows Spotlight 2. 需連線至網路
Postcondition	開啟瀏覽器並連線至網頁
Main Success Scenario	1. 使用者透過快捷鍵呼叫系統 2. 系統顯示輸入框，且游標停在輸入框 3. 使用者輸入網站名稱 4. 依照使用者輸入的網站名稱搜尋，並顯示於清單 5. 開啟網站
Extensions	4a. 若找不到使用者輸入的網頁 1. 顯示沒有結果 4b. 若未連至網路 1. 提醒使用者需連線至網路 5a. 開啟網站: 1. 開啟瀏覽器並自動連線至該網站
Speciel Requirement	None
Technology And Data Variations List	None
Frequency of Occurrence	偶爾發生 (當使用者想透過此應用程式開啟網頁時)
Open Issues	None

Use Case ID	UC-05
Use Case Name	Convert Exchange (轉換匯率)
Scope	Windows Spotlight
Level	User goal
Primary Actor	User
Stakeholders & Interests	User: 輸入數字及貨幣代碼
Precondition	1. 使用者開啟Windows Spotlight 2. 需輸入數字及貨幣代碼 3. 需連上網路
Postcondition	換算各國匯率

Main Success Scenario	1. 使用者開啟Windows Spotlight 2. 開啟後游標停在輸入框 3. 使用者輸入要轉換的數字及貨幣代碼 4. 系統至網路上取得今天的匯率並做運算 5. 顯示轉換結果
Extensions	1a. 無法開啟Windows Spotlight 1. 無任何反應 2a. 游標未停在輸入框 1. 使用者自行點擊到輸入框 3a. 輸入非數字及貨幣代碼 1. 不做匯率轉換 3b. 只輸入數字及貨幣代碼，未輸入指定要轉換的幣別代碼 1.預設轉換為新台幣 3c. 有輸入要轉換的幣別代碼 1.轉換為指定幣別 4a. 為連接到網路 1.提示尚未連接到網路 5a. 運算出結果 1. 將結果顯示出來
Speciel Requirement	None
Technology And Data Variations List	None
Frequency of Occurrence	偶爾發生 (當使用者要運算匯率轉換時)
Open Issues	1. 顯示當下轉換的匯率 2. 顯示其他主要貨幣的匯率

7. Non-functional Requirements and Constraints

NFR ID	Category	Description
NFR-01	Usability	提供使用者方便操作的介面
Scenario: 使用者在使用時只需要輸入文字，系統會自動判定使用者想要使用的是什麼功能		
NFR-02	Performance	搜尋檔案的時間不超過三秒
Scenario: 使用者在搜尋檔案時，依照使用者目前輸入的資訊找到檔案的時間不能超過三秒		
NFR-03	Usability	提供使用者利用快捷鍵呼叫
Scenario: 使用者可以透過快捷鍵呼叫，游標會自動移動到輸入框，讓使用者不用浪費時間使用滑鼠		

NFR ID	Category	Description
NFR-04	Usability	使用者可透過上下鍵選擇欲執行的功能
Scenario: 使用者輸入完關鍵字後，使用者可以利用上下鍵來選擇要執行的功能		

8. Glossary

Item	Definition or Description
View	使用者介面
Controller	依照關鍵字來決定要執行的功能的模組
File System	當搜尋檔案時使用的系統
Translate	當使用翻譯時使用的系統
Calculator	當計算數字時使用的系統
Search Engine	使用搜尋引擎來搜尋網頁的系統
Exchange	計算匯率時使用的系統

9. Software Environments

Environments: win10
Development language: C#

Domain model

1. Domain class diagram showing only concepts

先找出在use case中出現的concepts：

User	DisplayPanel	Adapter	SearchFile
FileSystem	ExchangeResult	FileName	File
ItemList	Translator	Translate	Word
Calculate	Calculator	Expression	CalculatorResult
FindWebsite	SearchEngine	Website	ExchangeCurrency
Exchange	Currency	Cash	

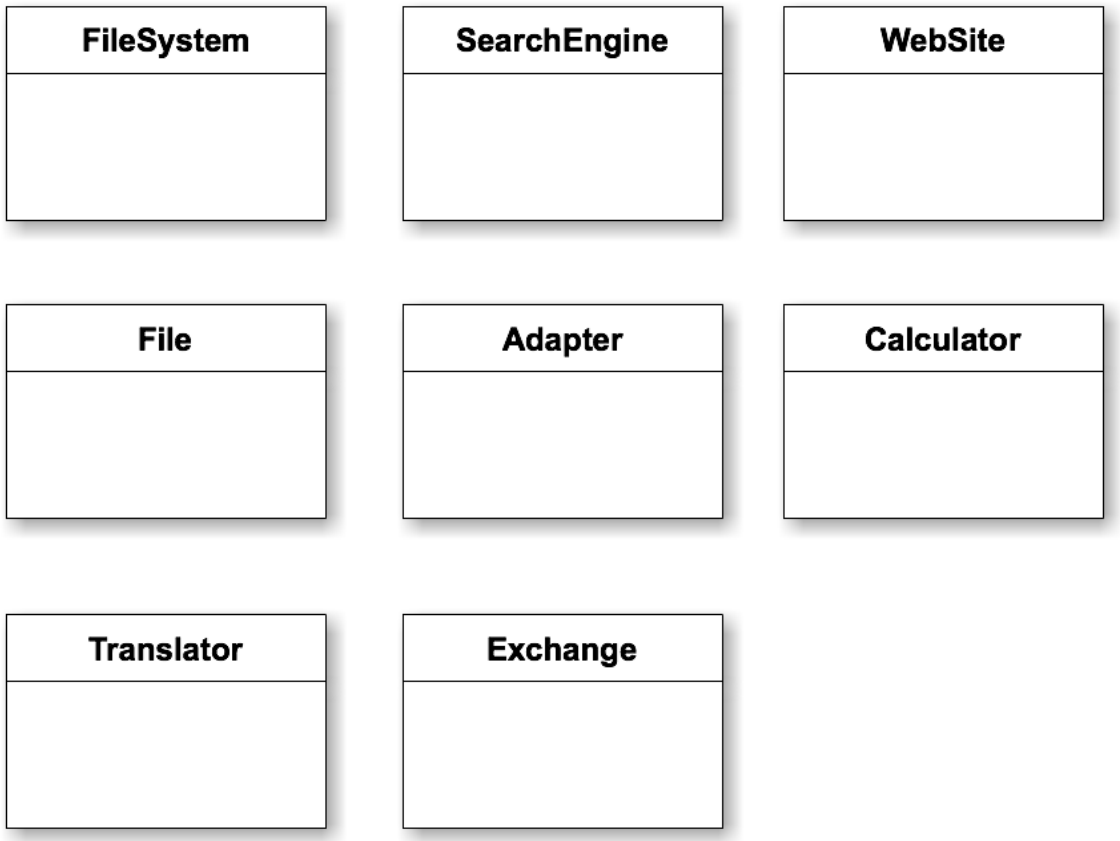
再從concepts中找出不好的classes：

Attributes	Operation	Roles	Implementation Construction	Redundant
FileName	SearchFile	User	ExchangeResult	DisplayPanel
Expression	Translate		CalculatorResult	
Currency	Calaulate		Word	
Cash	ExchangeCurrency			
ItemList	FindWebsite			

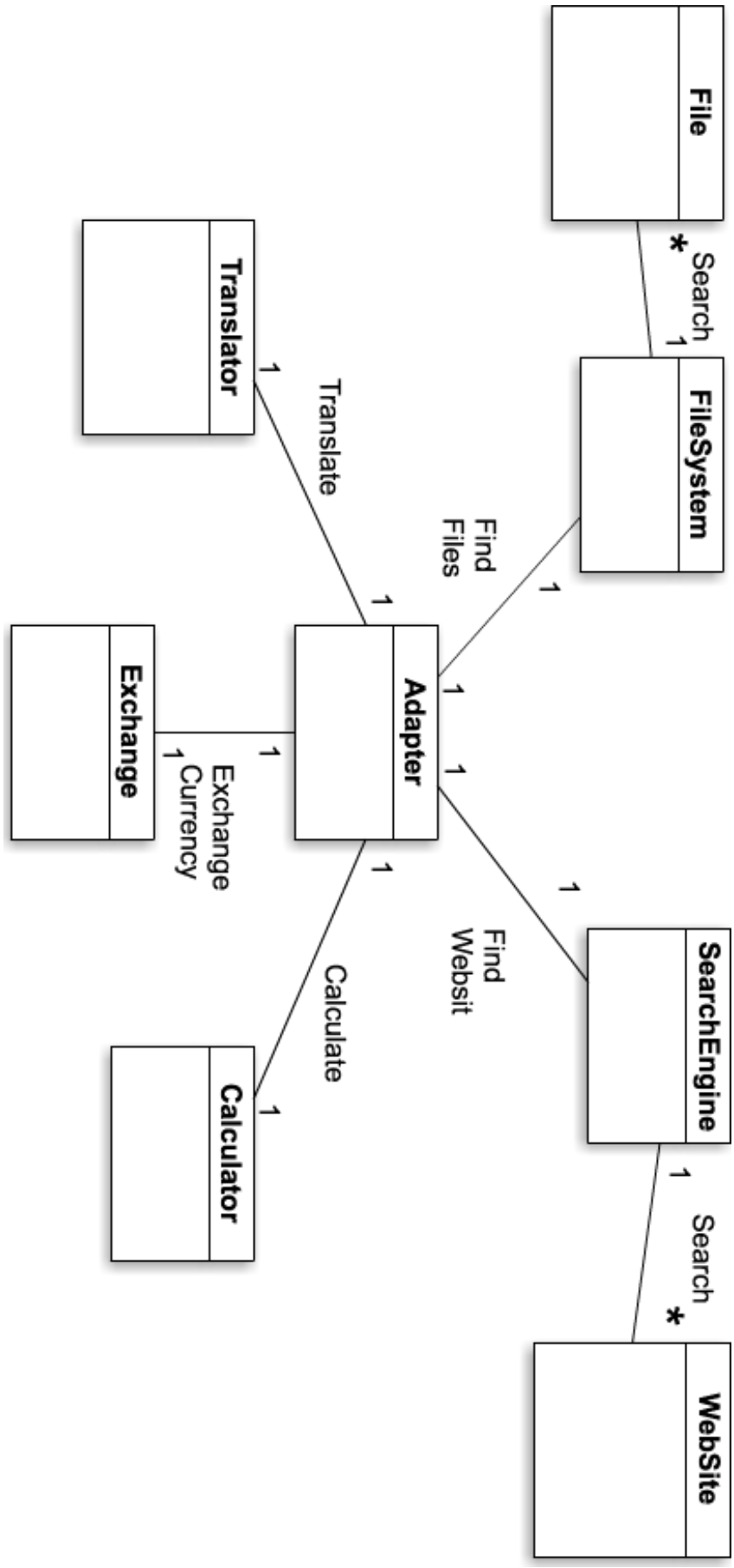
將不好的classes分為五種

- Attributes：可用attributes來代替class
- Operation：是操作步驟中的一部分
- Roles：角色
  - User是指使用者，不適合在Domain Model中呈現
- Implementation Construction：實作產生的結構
  - ExchangeResult是Exchange回傳給Adapter的資訊
  - CalculatorResult是Calculator回傳給Adapter的資訊
  - Word是Translator回傳給Adapter的資訊
- Redundant：多餘的Classes
  - DisplayPanel(操作介面)，不適合在Domain Model中呈現

本專案適合的Classes

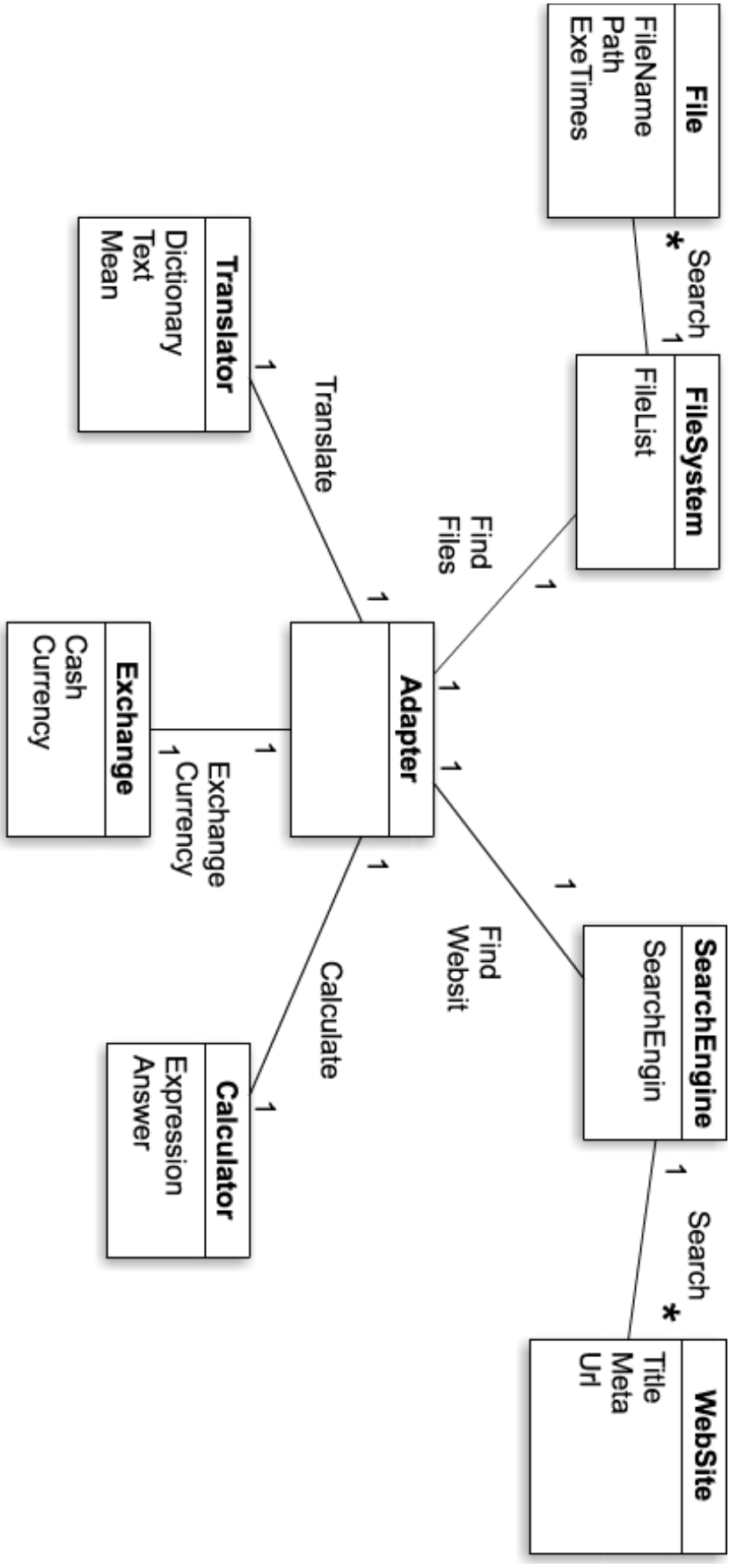


2. Add associations



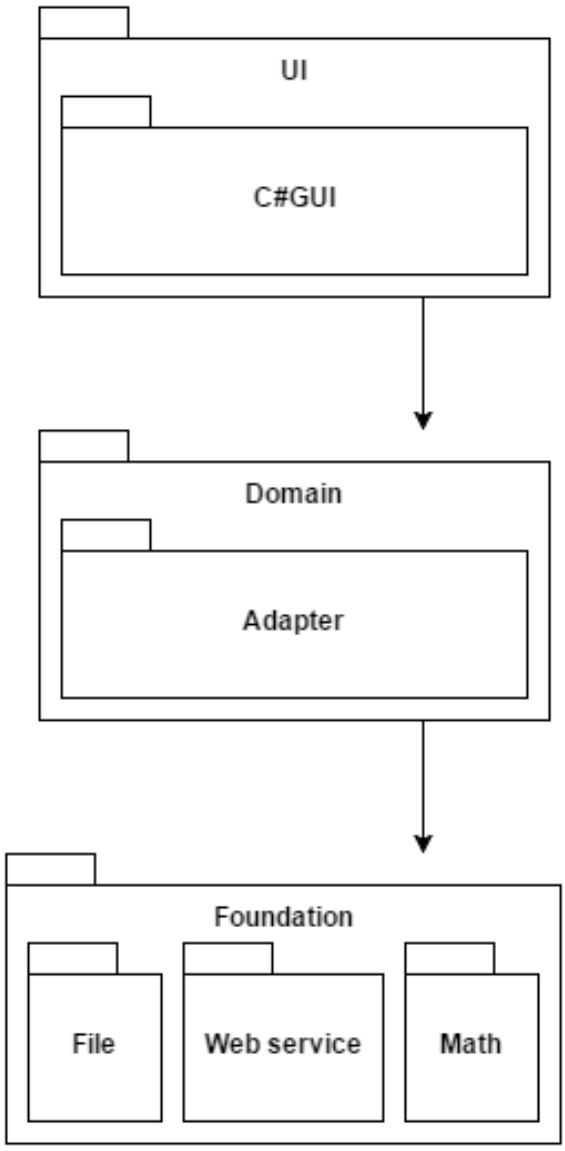
Relationship
Adapter分配搜尋檔案的工作給FileSystem
FileSystem搜尋檔案
Adapter將翻譯的工作分配給Translator
Adapter將搜尋網站的工作分配給SearchEngine
SearchEngine搜尋網站
Adapter分配計算運算式的工作給Calculator
Adapter分配計算匯率的工作給Exchange

3. Add attributes



Design

1. Logical Architecture

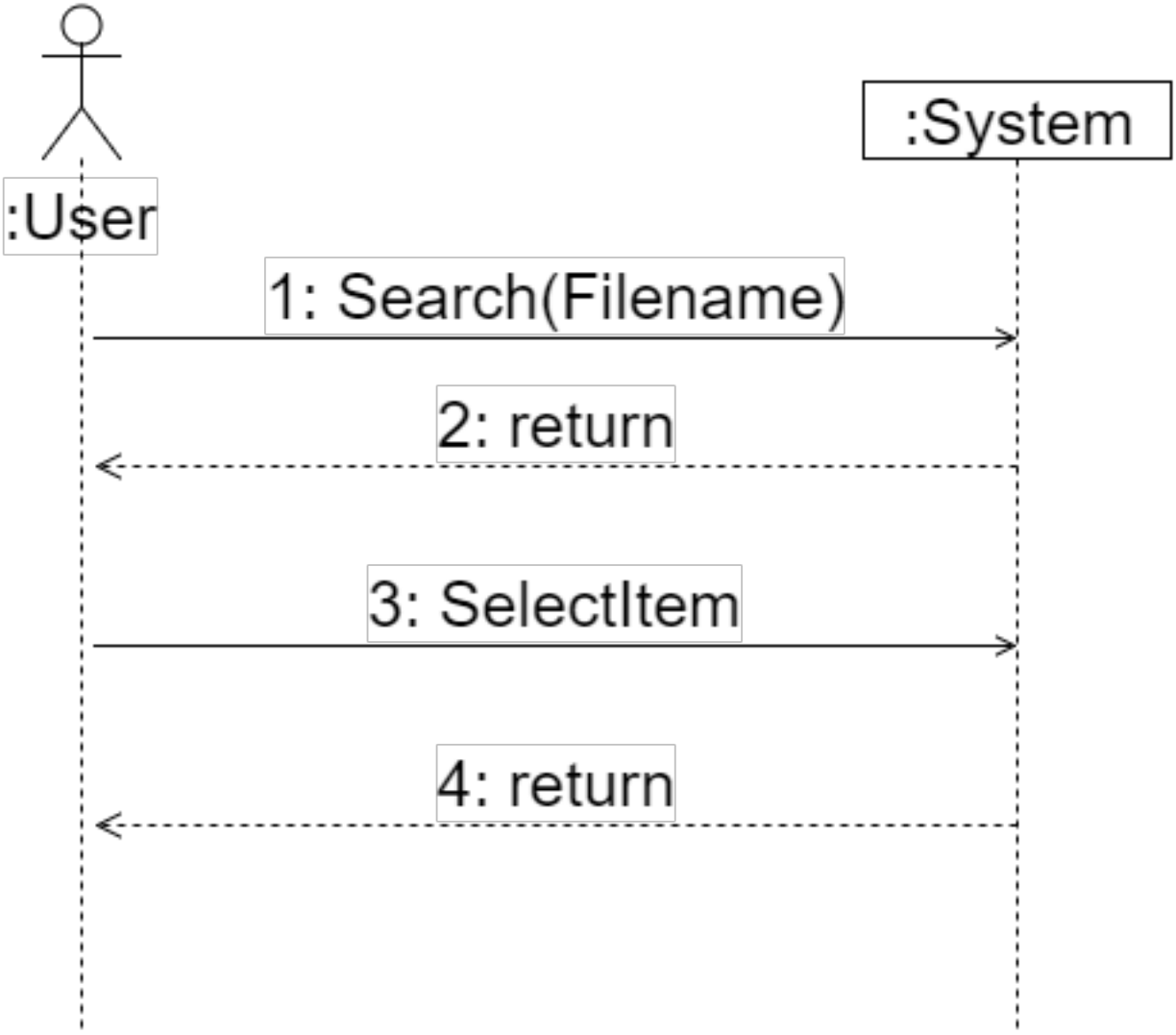


C#GUI：Windows Spotlight UI的相關類別
Adapter：分配工作的相關類別
File：檔案存取搜尋的相關類別
Web service：網路服務相關的類別
Math：數學運算相關的類別

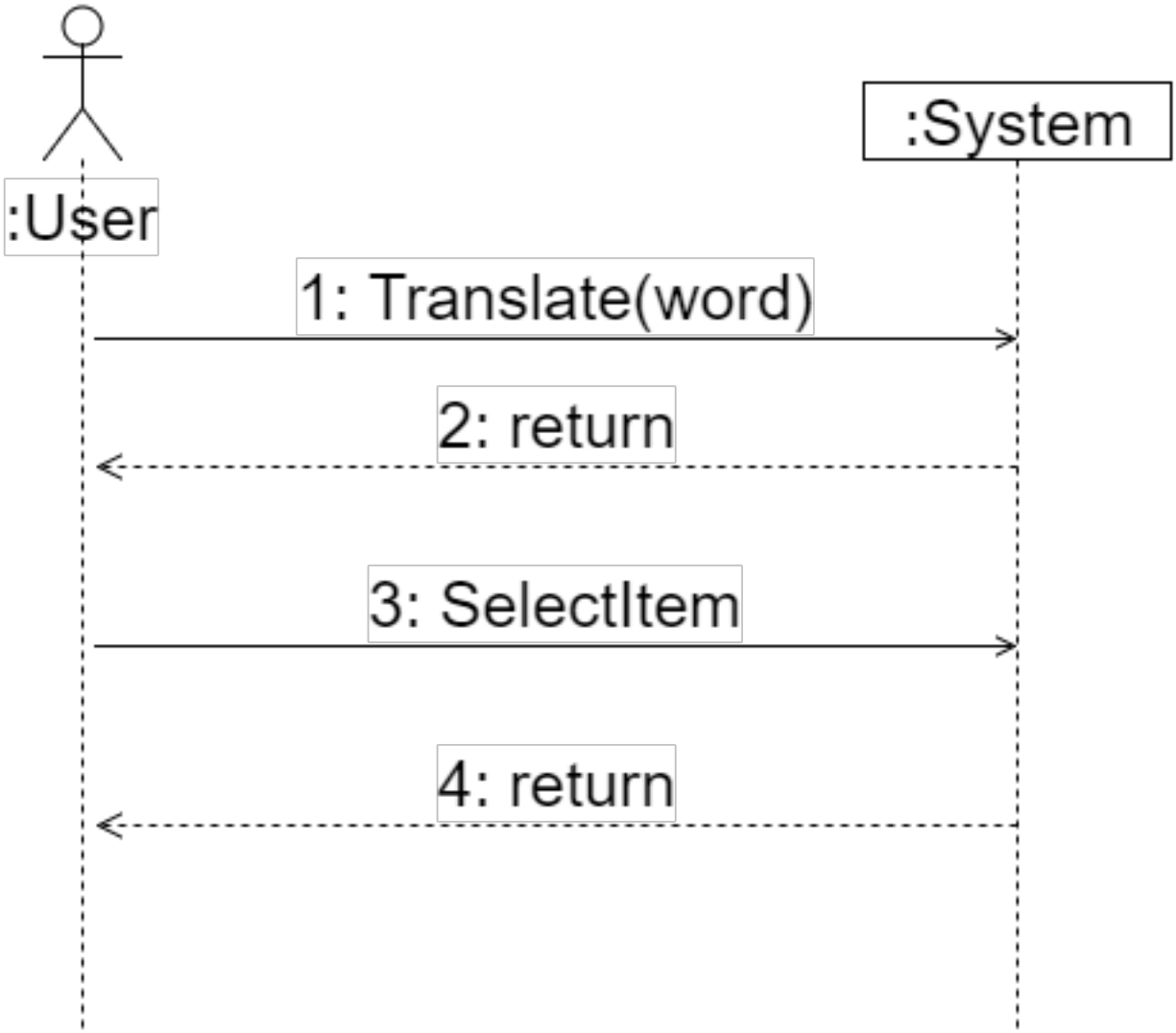
2. Use-Case Realizations with GRASP Patterns

1. SSD

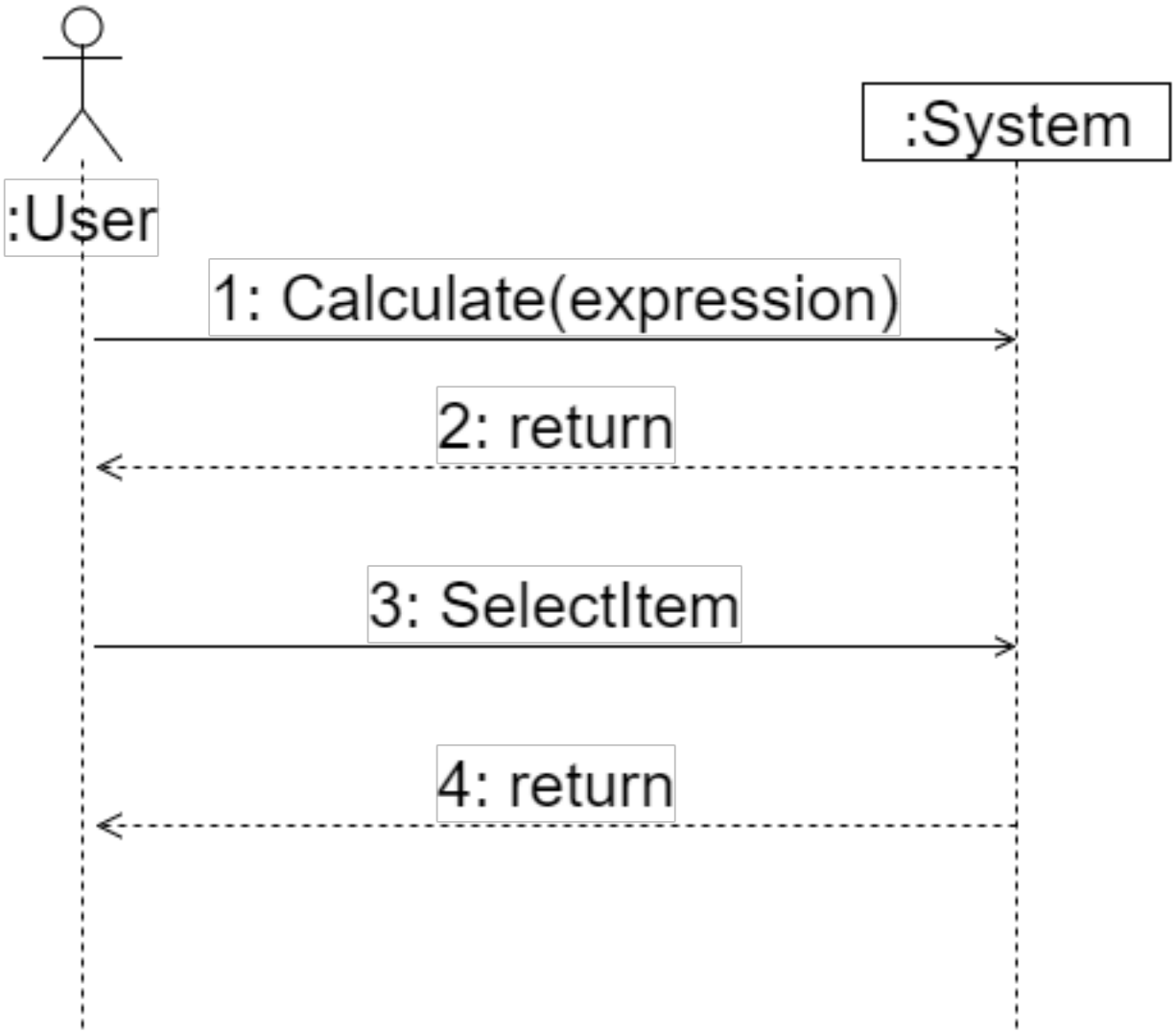
Search File(搜尋檔案)



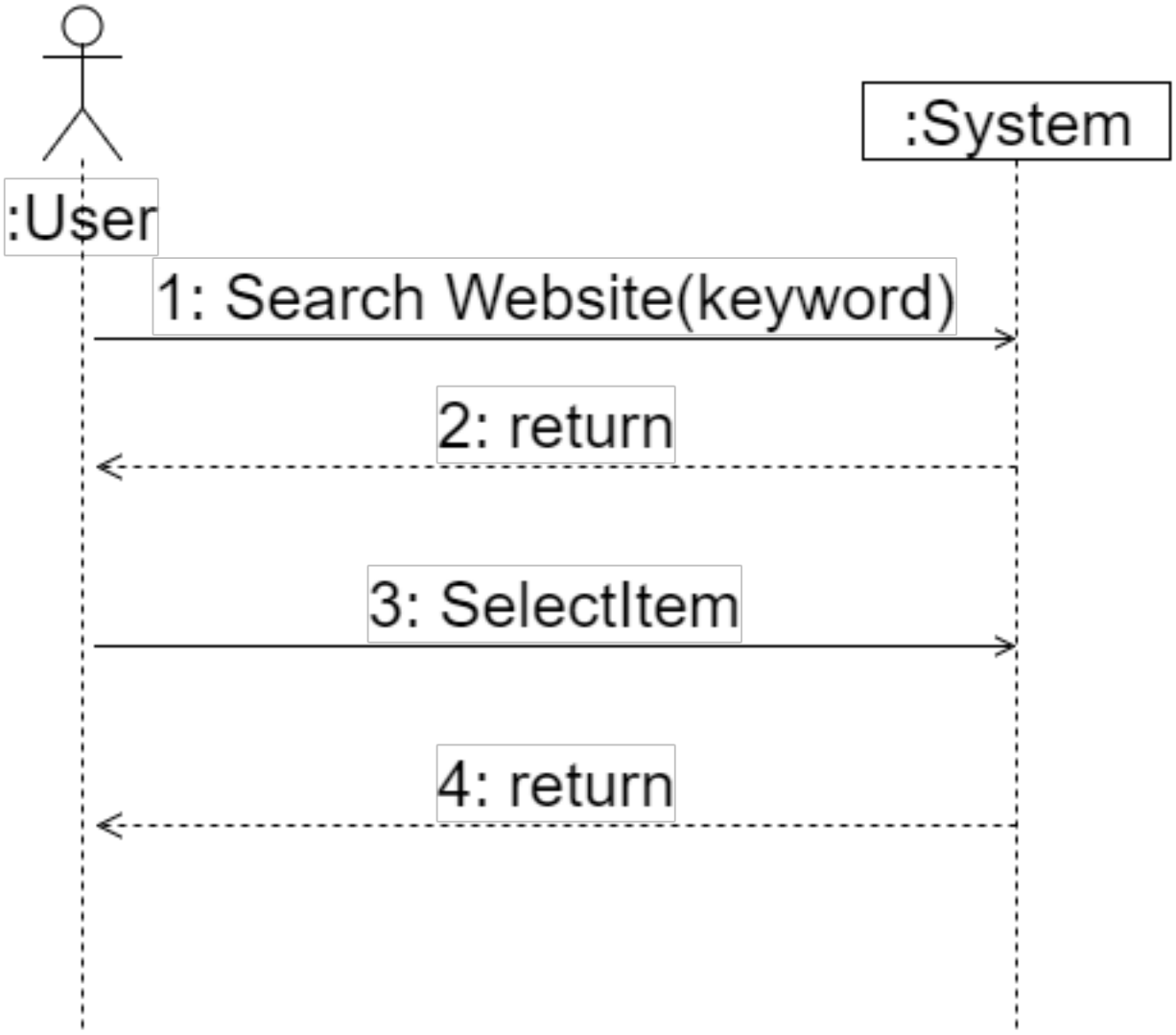
Translate English(翻譯)



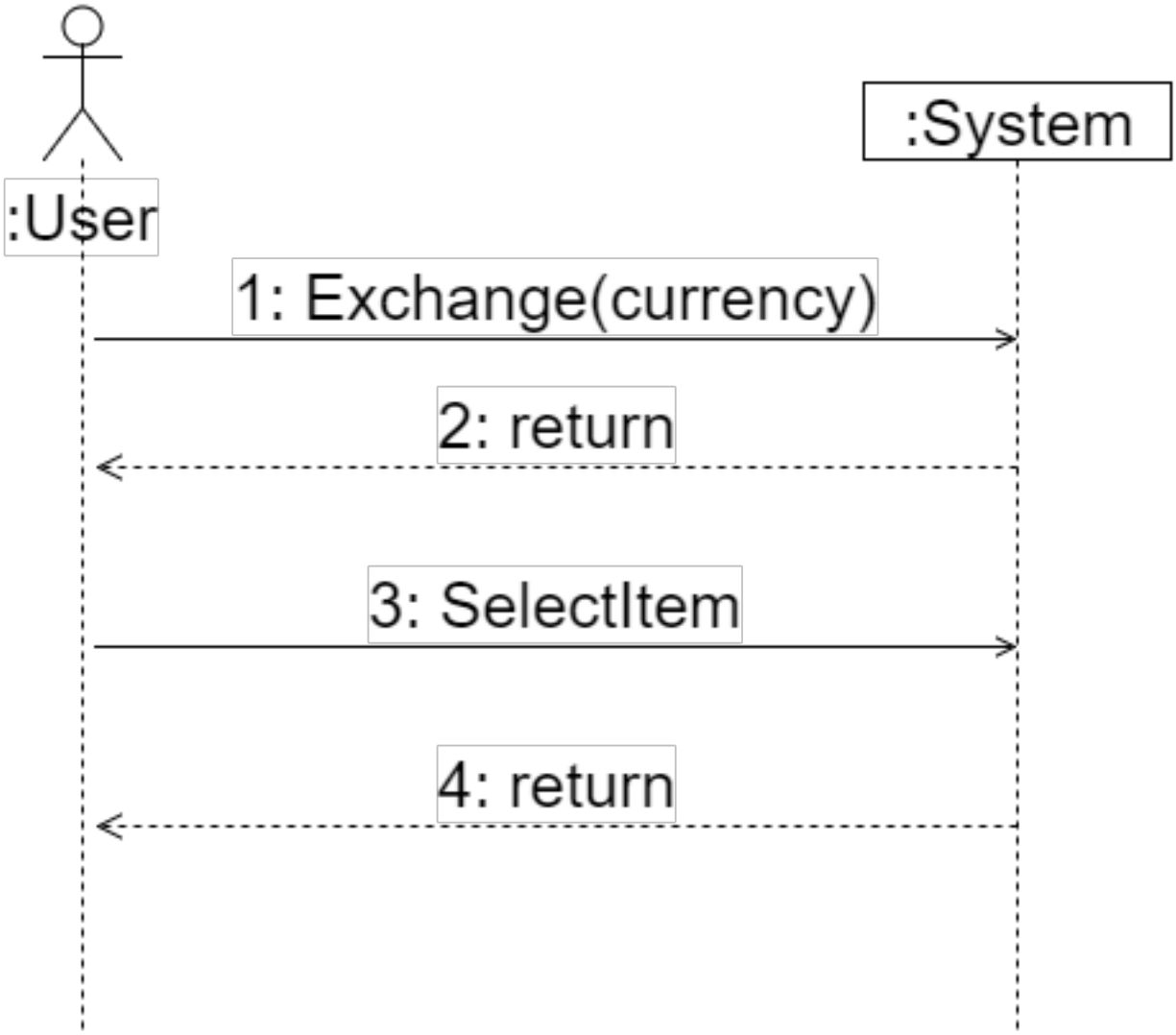
Computing Equation(計算公式)



Search Website(搜尋網頁)



Convert Exchange(轉換匯率)



2. Contract

Contract01 - Search(Filename):

Operation:	Search(filename)
Cross Reference:	Use Case: UC-01
Precondition:	Windows Spotlight is opened.
Postconditions:	回傳檔案的搜尋結果，並顯示於UI上

Contract02 - Translate(word):

Operation:	Translate(word)
Cross Reference:	Use Case: UC-02
Precondition:	Windows Spotlight is opened.
Postconditions:	回傳單字的翻譯結果，並顯示於UI上

Contract03 - Calculate(expression):

Operation:	Calculate(expression)
Cross Reference:	Use Case: UC-03
Precondition:	Windows Spotlight is opened.
Postconditions:	回傳算式的計算結果，並顯示於UI上

Contract04 -SearchWebsite(keyword):

Operation:	SearchWebsite(keyword)
Cross Reference:	Use Case: UC-04
Precondition:	Windows Spotlight is opened.
Postconditions:	回傳網頁的搜尋結果，並顯示於UI上

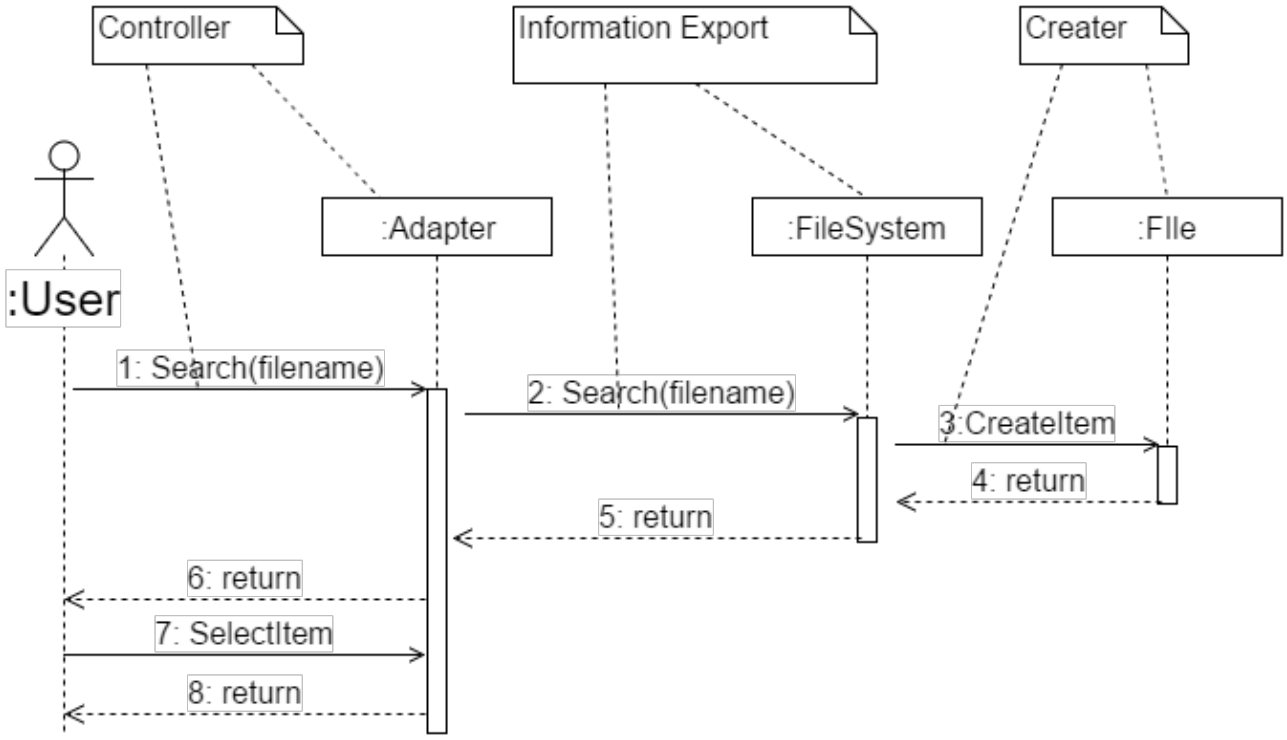
Contract05 -Exchange(currency):

Operation:	Exchange(currency)
Cross Reference:	Use Case: UC-05
Precondition:	Windows Spotlight is opened.
Postconditions:	回傳匯率的轉換結果，並顯示於UI上

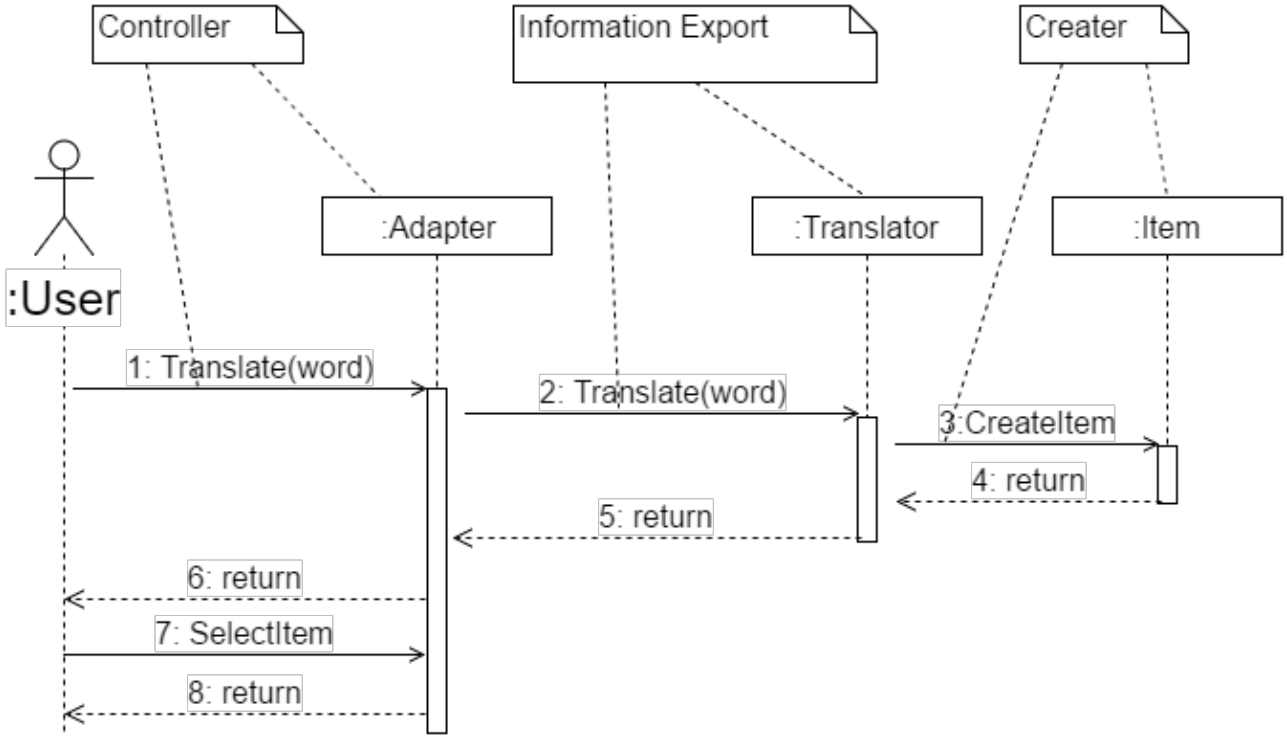
Contract06 - SelectItem:

Operation:	SelectItem
Cross Reference:	Use Case: UC-01, UC-02, UC-03, UC-04, UC-05
Precondition:	UI上已顯示各功能回傳的結果
Postconditions:	執行使用者所選擇的功能

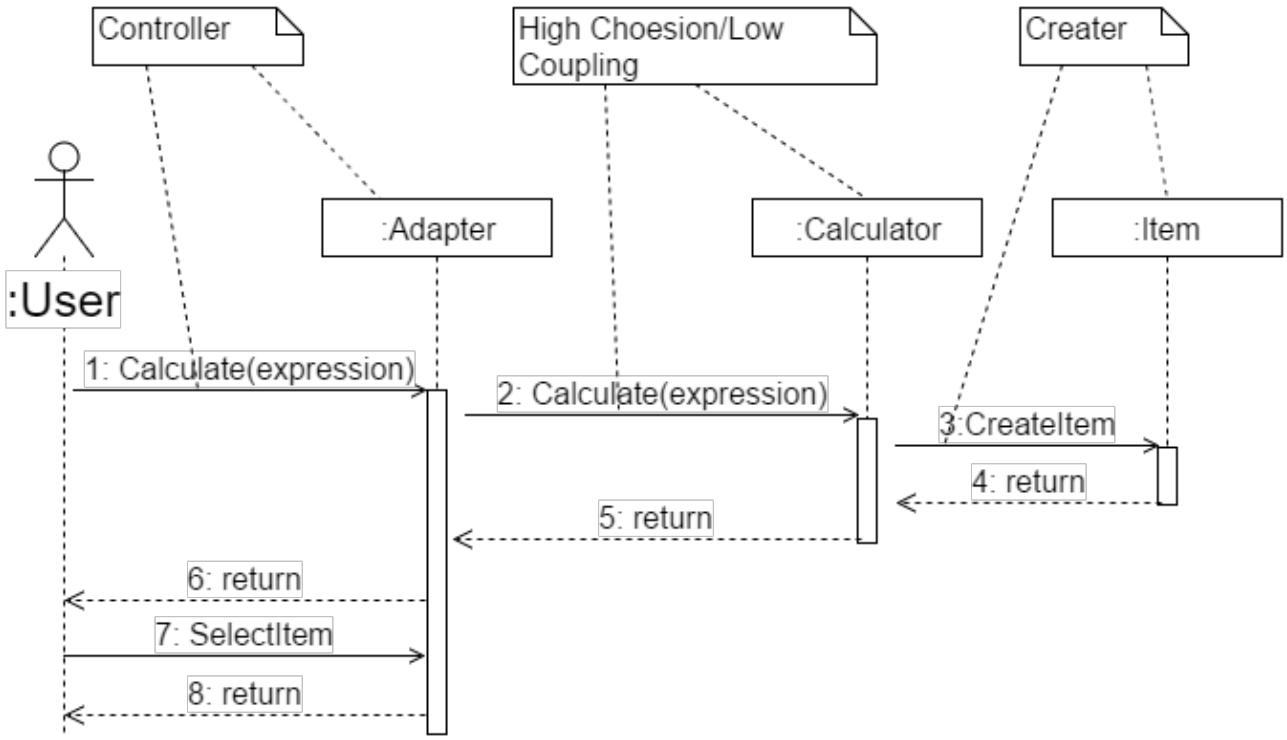
Search(filename)



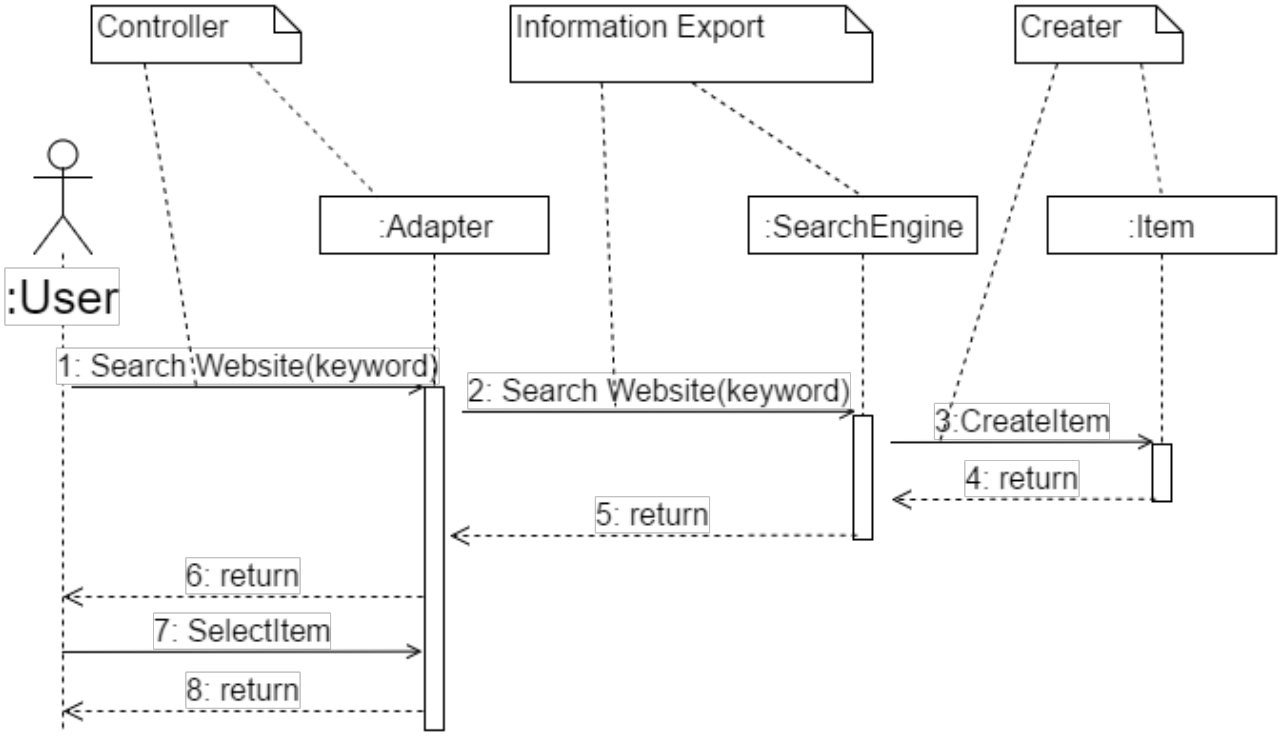
Translate(word)



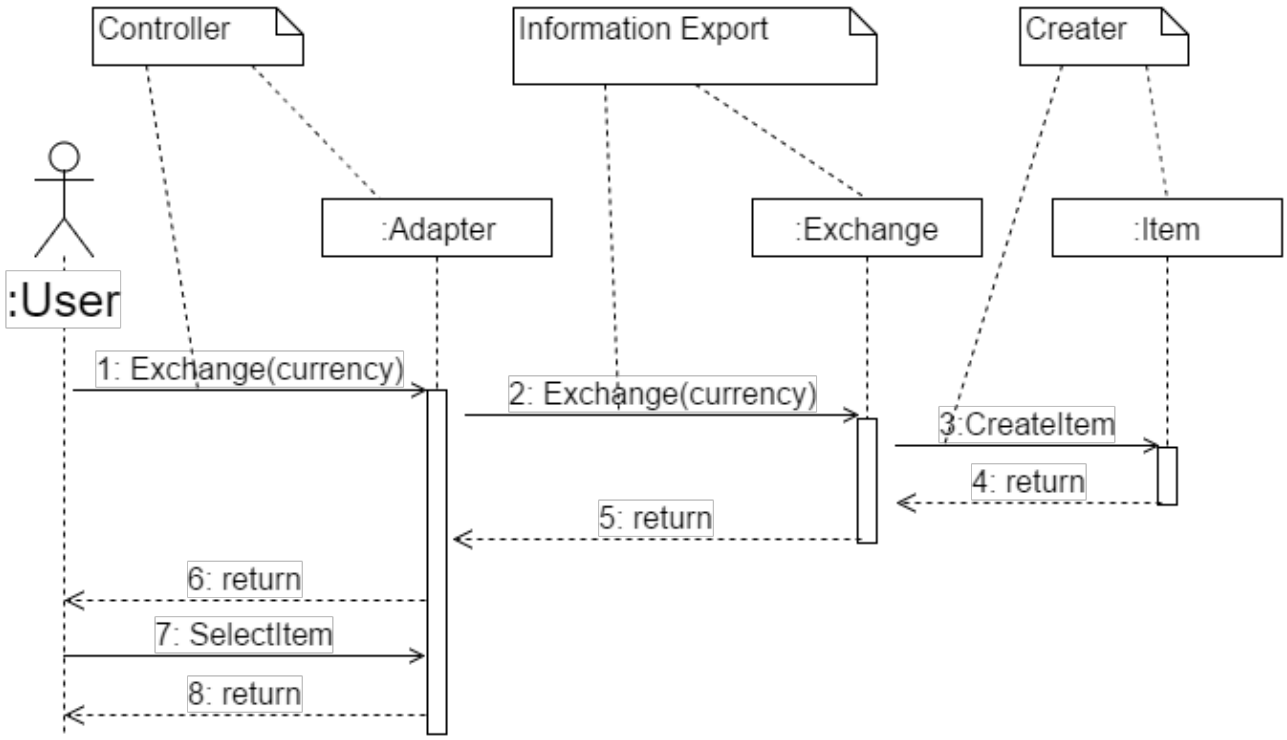
Calculate(expression)



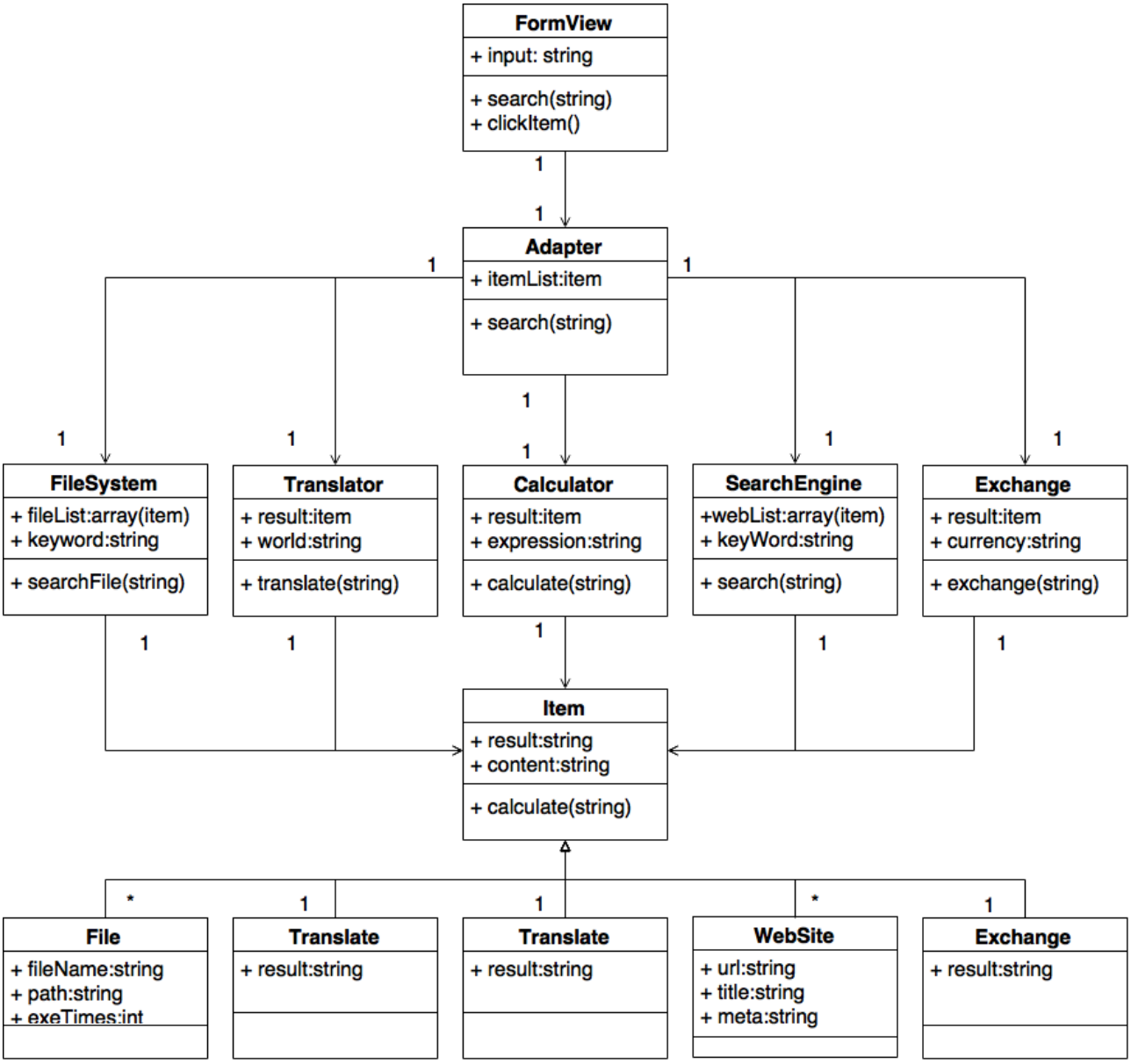
SearchWebsite(keyword)



Exchange(currency)

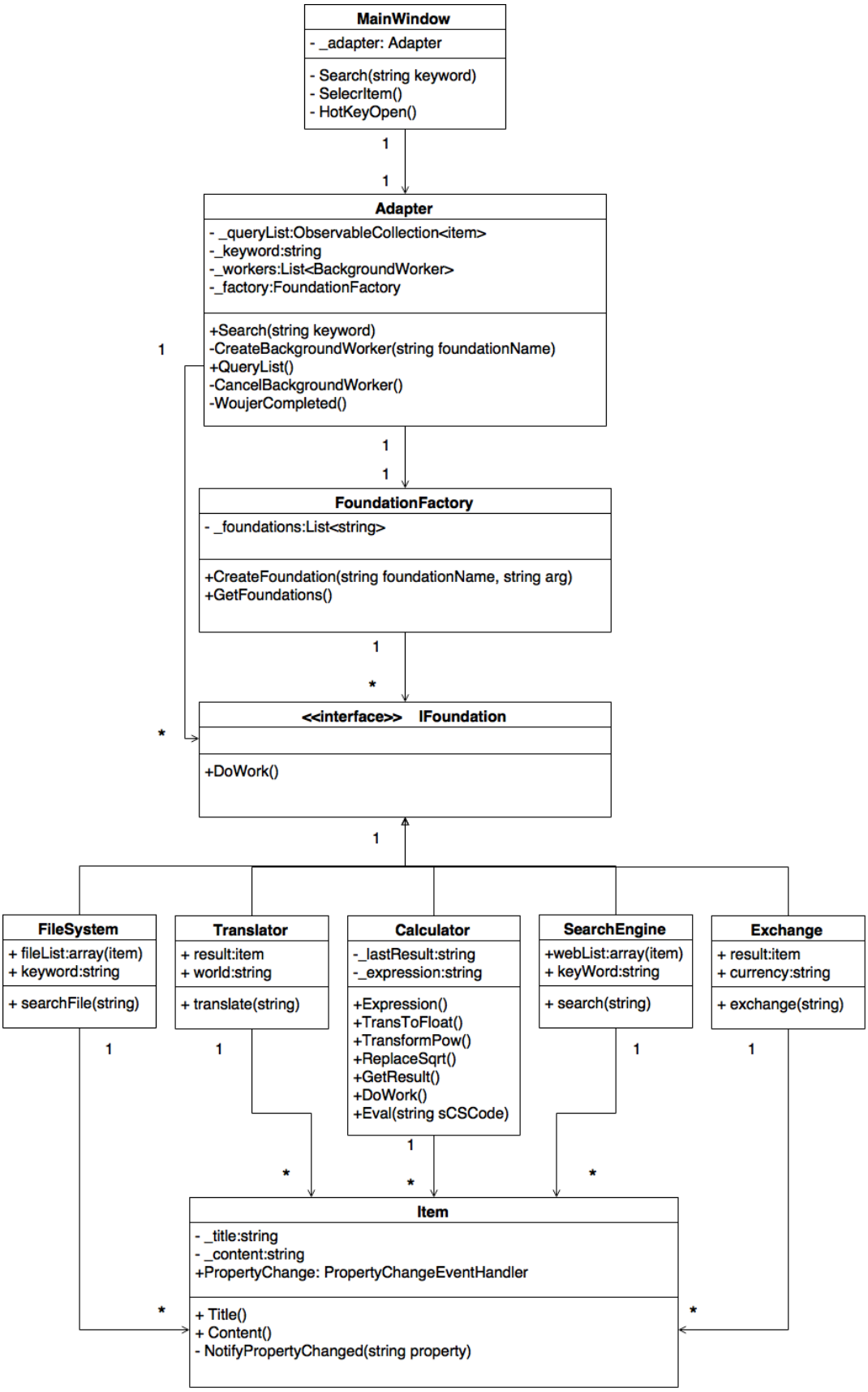


3. Design Class Model(domain class model after class design)



Implementation Class Model

1. Implementation class diagram



2. Difference between implementation class model and design class model

Class	Method	Design	Imp.
MainWindow	Search()	YES	YES
	SelecrItem()	YES	YES
	HotKeyOpen	NO	YES
Adapter	Search()	YES	YES
	CreateBackgroundWorker	NO	YES
	QueryList()	NO	YES
	CancelBackgroundWorker()	NO	YES
	WorkerCompleted()	NO	YES
FoundationFactory (added in imp. phase)	CreateFoundation()	NO	YES
	GetFoundations()	NO	YES
Foundation (added in imp. phase)	DoWork	NO	YES
Calculator	Expression()	NO	YES
	TransToFloat()	NO	YES
	TransformPow()	NO	YES
	ReplaceSqrt()	NO	YES
	GetResult()	NO	YES
	DoWork()	NO	YES
	Eval()	NO	YES
	Calculator()	YES	NO
Item	Title()	NO	YES
	Content()	NO	YES
	NotifyPropertyChanged()	NO	YES
	Calculator()	YES	NO

Table 5.2.1: Comparison with design and implementation class

	Number of added	Number of removed	Number of modified
Class	2	5	4
Method	18	2	1

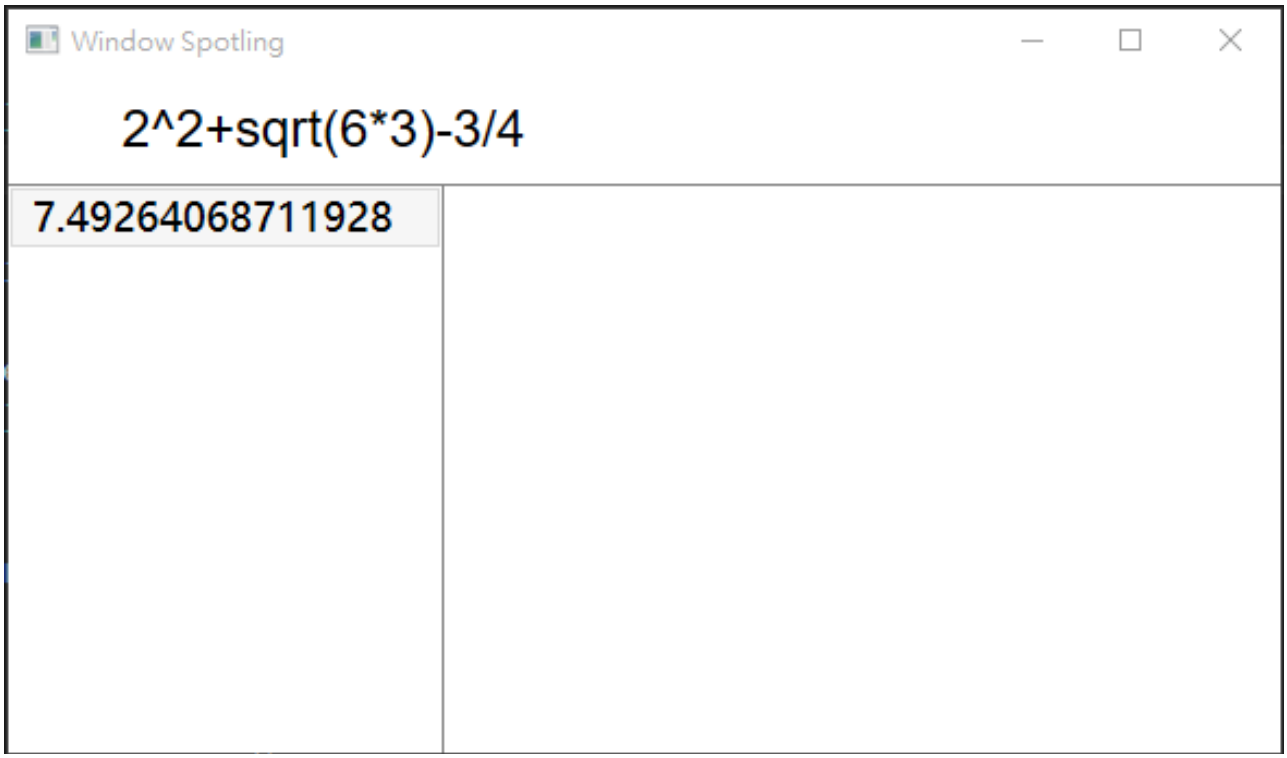
able 5.2.2: Summary of implementation class/method changed

3. Calculate Line of Code

NO	Class Name	Number of Methods	Line of Code in Class (without comment)
1	**MainWindow**	3	55
2	**Adapter**	5	78
3	**FoundationFactory**	2	39
4	**Foundation**	1	14
5	**Calculator**	7	243
6	**Item**	3	44
Sum		21	473

Programming

1. Snapshots of system execution



2. Source Code Listing
    1. MainWindow

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Threading;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;


namespace WPF_Windows_Spotlight
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        private Adapter _adapter;


        public MainWindow()
```

```
        {
            _adapter = new Adapter();
            InitializeComponent();
            QueryList.ItemsSource = _adapter.QueryList;
            this.KeyDown += new KeyEventHandler(HotKeyOpen);
        }

        private void Search(object sender, TextChangedEventArgs e)
        {
            _adapter.Search(Input.Text);
        }

        private void SelectItem(object sender, SelectionChangedEventArgs e)
        {
            ListBox list = (ListBox)sender;
            Item selectedItem = _adapter.QueryList[list.SelectedIndex];

        }

        private void HotKeyOpen(object sender, KeyEventArgs e)
        {
            if ((Keyboard.IsKeyDown(Key.LeftCtrl) && Keyboard.IsKeyDown(Key.A)))
            {
                this.Visibility = Visibility.Visible;
            }
        }
    }

}
```

2. Adapter

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Threading;
using WPF_Windows_Spotlight.Foundation;
using System.Collections.ObjectModel;

namespace WPF_Windows_Spotlight
{
    public class Adapter
    {
        private string _keyword;
        private List<BackgroundWorker> _workers;
        private FoundationFactory _factory;
        private ObservableCollection<Item> _queryList;

        public Adapter()
        {
            _workers = new List<BackgroundWorker>();
            _factory = new FoundationFactory();
            _queryList = new ObservableCollection<Item>();
        }

        public void Search(string keyword)
        {
            _keyword = keyword;
            _queryList.Clear();
            CancelBackgroundWorker();
            List<string> foundations = _factory.GetFoundations();
```

```
            foreach (string foundation in foundations)
            {
                BackgroundWorker worker = CreateBackgroundWorker(foundation);
                worker.RunWorkerAsync();
                _workers.Add(worker);
            }
        }

        private BackgroundWorker CreateBackgroundWorker(string foundationName)
        {
            BackgroundWorker bgworker = new BackgroundWorker();
            IFoundation foundation = _factory.CreateFoundation(foundationName, _keyword);
            bgworker.WorkerSupportsCancellation = true; // 支援取消
            bgworker.RunWorkerCompleted += WorkerCompleted; // 結束時呼叫
            bgworker.DoWork += foundation.DoWork; // start thread時呼叫
            return bgworker;
        }

        public ObservableCollection<Item> QueryList
        {
            get { return _queryList; }
        }

        private void CancelBackgroundWorker()
        {
            foreach (BackgroundWorker worker in _workers)
            {
                if (worker.IsBusy)
                {
                    worker.CancelAsync();
                }
            }
            _workers.Clear();
        }

        private void WorkerCompleted(object sender, RunWorkerCompletedEventArgs e)
        {
            Console.WriteLine("Result = " + e.Result);
            if (e.Result != null)
            {
                //_result = e.Result.ToString();
                _queryList.Add((Item)e.Result);
            }
        }
    }
}
```

3. FoundationFactory

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WPF_Windows_Spotlight.Foundation
{
    public class FoundationFactory
    {
        private List<string> _foundations;

        public FoundationFactory()
        {
```

```
            _foundations = new List<string>();
            _foundations.Add("calculator");
        }

        public IFoundation CreateFoundation(string foundationName, string arg)
        {
            IFoundation foundation;
            switch (foundationName)
                    {
                case "calculator":
                    foundation = new Calculator(arg);
                    break;
                                default:
                    foundation = null;
                    break;
                    }
            return foundation;
        }

        public List<string> GetFoundations()
        {
            return _foundations;
        }
    }
}
```

4. IFoundation

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.ComponentModel;

namespace WPF_Windows_Spotlight.Foundation
{
    public interface IFoundation
    {
        void DoWork(object sender, DoWorkEventArgs e);
    }
}
```

5. Calculator

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.CSharp;
using System.CodeDom;
using System.CodeDom.Compiler;
using System.Reflection;
using System.Text.RegularExpressions;

namespace WPF_Windows_Spotlight.Foundation
{
    public class Calculator : IFoundation
    {
        private string _expression;
        private string _lastResult;
```

```
public Calculator(string expression = "")
{
   _expression = expression.ToLower() + ";";
}

public string Expression
{
   set { _expression = value.ToLower() + ";"; }
}

public void TransToFloat ()
{
   int isFloat = 0;
   int isNumber = 0;
   for (int i = 0; i < _expression.Length; i++)
   {
      if (_expression[i] >= '0' && _expression[i] <= '9')
      {
         isNumber = 1;
      }
      else if (_expression[i] == '.')
      {
         isFloat = 1;
      }
      else
      {
         if (isNumber == 1 && isFloat == 0)
         {
            _expression = _expression.Insert(i, ".0");
            i = i+2;
         }
         isNumber = 0;
         isFloat = 0;
      }
   }

   Console.WriteLine(_expression);
}

public void TransformPow ()
{
   string[] resultString = Regex.Split(_expression, @"\^");
   if (resultString[0] != _expression)
   {
      string splitLeft = resultString[0];
      string splitRight = resultString[1];
      int leftBracketsCount = 0;
      int leftPosition = 0;
      int rightBracketsCount = 0;
      int rightPosition = 0;
      string powLeft = "";
      string powRight = "";
      //指數符號左邊 尋找括號
      if (splitLeft[splitLeft.Length - 1] == ')')
      {
         for (int i = splitLeft.Length - 1; i >= 0; i--)
         {
            if (splitLeft[i] == '(')
            {
               leftBracketsCount--;
            }
            else if (splitLeft[i] == ')')
            {
```

```
                leftBracketsCount++;
            }

            if (leftBracketsCount == 0)
            {
                leftPosition = i;
                break;
            }
        }
    }
    else
    {
        for (int i = splitLeft.Length - 1; i >= 0; i--)
        {
            if (splitLeft[i] == '+' || splitLeft[i] == '-' || splitLeft[i] == '*' || splitLeft[i] == '/' || splitLeft[i] == '^')
            {
                leftPosition = i + 1;
                break;
            }
        }
    }
    //右邊尋找括號
    if (splitRight[0] == '(')
    {
        for (int i = 0; i < splitRight.Length; i++)
        {
            if (splitRight[i] == ')')
            {
                rightBracketsCount--;
            }
            else if (splitRight[i] == '(')
            {
                rightBracketsCount++;
            }

            if (rightBracketsCount == 0)
            {
                rightPosition = i + 1;
                break;
            }
        }
    }
    else
    {
        for (int i = 0; i < splitRight.Length; i++)
        {
            if (splitRight[i] == '+' || splitRight[i] == '-' || splitRight[i] == '*' || splitRight[i] == '/' || splitRight[i] == '^')
            {
                rightPosition = i;
                break;
            }
            rightPosition = i;
        }
    }
    //轉換成 pow
    for (int i = leftPosition; i < splitLeft.Length; i++)
    {
        powLeft = powLeft + splitLeft[i];
    }

    for (int i = 0; i < rightPosition; i++)
    {
```

```
            powRight = powRight + splitRight[i];
        }

        _expression = "";
        for (int i = 0; i < leftPosition; i++)
        {
            _expression = _expression + splitLeft[i];
        }
        _expression = _expression + "Math.Pow(" + powLeft + "," + powRight + ")";
        for (int i = rightPosition; i < splitRight.Length; i++)
        {
            _expression = _expression + splitRight[i];
        }
    }

    Console.WriteLine(_expression);
}

public void ReplaceSqrt ()
{
    _expression = _expression.Replace("sqrt", "Math.Sqrt");
    Console.WriteLine(_expression);
}

public string GetResult()
{
    try
    {
        ToLower();
        TransToFloat();
        ReplaceSqrt();
        TransformPow();
        string result = Eval(_expression).ToString();
        _lastResult = result;
        return _lastResult;
    }
    catch (Exception e)
    {
        return _lastResult;
    }
}

public void DoWork(object sender, DoWorkEventArgs e)
{
    Item item = new Item();
    item.Title = GetResult();
    e.Result = item;
}

private static object Eval(string sCSCode)
{
    CSharpCodeProvider c = new CSharpCodeProvider();
    ICodeCompiler icc = c.CreateCompiler();
    CompilerParameters cp = new CompilerParameters();

    cp.ReferencedAssemblies.Add("system.dll");
    cp.ReferencedAssemblies.Add("system.xml.dll");
    cp.ReferencedAssemblies.Add("system.data.dll");
    cp.ReferencedAssemblies.Add("system.windows.forms.dll");
    cp.ReferencedAssemblies.Add("system.drawing.dll");

    cp.CompilerOptions = "/t:library";
    cp.GenerateInMemory = true;
```

```
            StringBuilder sb = new StringBuilder("");
            sb.Append("using System;\n");
            sb.Append("using System.Xml;\n");
            sb.Append("using System.Data;\n");
            sb.Append("using System.Data.SqlClient;\n");
            sb.Append("using System.Windows.Forms;\n");
            sb.Append("using System.Drawing;\n");

            sb.Append("namespace CSCodeEvaler{ \n");
            sb.Append("public class CSCodeEvaler{ \n");
            sb.Append("public object EvalCode(){\n");
            sb.Append("return " + sCSCode + "; \n");
            sb.Append("} \n");
            sb.Append("} \n");
            sb.Append("}\n");

            CompilerResults cr = icc.CompileAssemblyFromSource(cp, sb.ToString());
            if (cr.Errors.Count > 0)
            {
               Console.WriteLine("ERROR: " + cr.Errors[0].ErrorText,
                 "Error evaluating cs code");
               return null;
            }

            System.Reflection.Assembly a = cr.CompiledAssembly;
            object o = a.CreateInstance("CSCodeEvaler.CSCodeEvaler");

            Type t = o.GetType();
            MethodInfo mi = t.GetMethod("EvalCode");

            object s = mi.Invoke(o, null);
            return s;
         }
      }
}
```

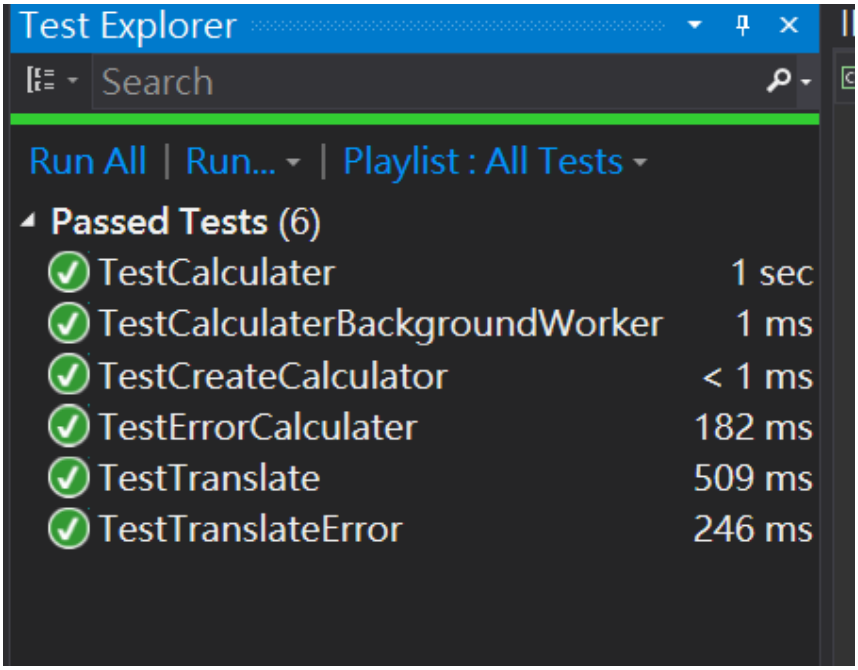
6. Item

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.ComponentModel;

namespace WPF_Windows_Spotlight
{
   public class Item : INotifyPropertyChanged
   {
      private string _title;
      private string _content;

      public string Title
      {
         get { return _title; }
         set {
            _title = value;
            NotifyPropertyChanged("Title");
         }
      }

      public string Content
      {
         get { return _content; }
```

```
            set
            {
                _content = value;
                NotifyPropertyChanged("Content");
            }
        }

        public event PropertyChangedEventHandler PropertyChanged;
        private void NotifyPropertyChanged(string property)
        {
            PropertyChangedEventHandler handler = PropertyChanged;
            if (handler != null)
            {
                handler(this, new PropertyChangedEventArgs(property));
            }
        }
    }
}
```

Unit Test

1. Snapshots of testing result



2. Unit Testing Code Listing
    1. CalculatorTest

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Threading.Tasks;
using System.ComponentModel;
using System.IO;
using HtmlAgilityPack;

namespace WPF_Windows_Spotlight.Foundation
{
   public class Translator : IFoundation
   {
      private string _word;
      private string _url;
      private string _xpath;

      public Translator(string word = "")
      {
         _word = word;
         _url = "https://tw.dictionary.search.yahoo.com/search?p=";
         _xpath = "//div[contains(@class, 'dd algo explain mt-20 lst DictionaryResults')]";
      }

      public string Word
      {
         set { _word = value; }
      }

      public string Translate()
```

```
        {
            HttpWebRequest request = (HttpWebRequest)WebRequest.Create(_url + _word);
            request.Accept = "text/html";
            request.Method = "GET";

            HttpWebResponse response = (HttpWebResponse)request.GetResponse();
            using (StreamReader stream = new StreamReader(response.GetResponseStream()))
            {
                string html = stream.ReadToEnd();
                HtmlDocument dom = new HtmlDocument();
                dom.LoadHtml(html);
                HtmlNode node = dom.DocumentNode.SelectSingleNode(_xpath);
                if (node == null)
                {
                    return "Not Found";
                }
                string result = node.InnerHtml;
                return result;
            }
        }

        public void DoWork(object sender, DoWorkEventArgs e)
        {
            e.Result = Translate();
        }
    }
}
```

Measurement

郭鎧瑋		劉至峻	
HW1			
2016/03/04 13:10 ~ 14:30	Problem statement	2016/03/04 13:10 ~ 14:30	Problem statement
HW2			
2016/03/13 14:08 ~ 15:38	1. Summary of System Features 2. Non-functional Requirements and Constraints 3. Glossary 4. Software Environments	2016/03/13 14:08 ~ 15:38	1. System Context Diagram 2. Use Case Diagram 3. Non-functional Requirement and Constraints 4. Glossary
HW3			
2016/03/13 15:45 ~ 16:40	1. Use Cases: UC-01	2016/03/13 15:45 ~ 16:40	1. Use Case: UC-02
2016/03/17 19:00 ~ 21:40	1. Use Cases: UC-01 2. Use Cases: UC-04	2016/03/15 19:00 ~ 21:00	1. Use Case: UC-03 2. Use Case: UC-05
2016/03/29 19:00 ~ 21:00	Domain Model	2016/03/29 19:00 ~ 21:00	Domain Model
2016/03/30 16:00 ~ 17:00	Associations Attributes	2016/03/30 16:00 ~ 17:00	Associations Attributes
HW4			
2016/04/23 20:40 ~ 22:30	1. Logical Architecture	2016/04/23 20:40 ~ 22:30	1. Logical Architecture
2016/04/24 16:10 ~ 18:00	1. SSD	2016/04/24 16:10 ~ 18:00	1. SSD 2. Class Model
2016/04/26 20:00 ~ 22:00	1. SSD	2016/04/26 20:00 ~ 22:00	1. SSD
2016/04/27 20:40 ~ 22:55	修復助教圈出的錯誤	2016/04/27 20:40 ~ 22:55	修復助教圈出的錯誤
HW5			
2016/05/1 18:00 ~ 21:00	Coding實作	2016/05/2 19:00 ~ 22:00	Coding實作
2016/05/3 20:00 ~ 00:00	Coding實作	2016/05/3 20:00 ~ 00:00	Coding實作
2016/05/4 21:00 ~ 00:30	1. 比較Class Diagram差異 2. 截圖	2016/05/4 21:00 ~ 00:30	1.畫Class Diagram 2. 整理

郭鎧瑋		劉至峻
Total	20.47 hours	20.47 hours