(T4)比較 RAM 的 Stack、Heap。比較 Struct、Class。比較 ValueType、ReferenceType

CourseGUID: 29f1196a-1950-41a4-b9c1-dd13a9e92d92

==

(T4)比較 RAM 的 Stack、Heap。比較 Struct、Class。比較 ValueType、ReferenceType

==

==

0. Summary

0.

Struct V.S. Class
RAM: Stack V.S. Heap
Value Type
Reference Type

1.
Struct V.S. Class
Value type V.S. Reference type
Reference:
http://net-informations.com/faq/net/stack-heap.htm

1.1.
Both Stack and Heap are in the RAM.
Constructor is a special method and will be called when the object is created.
Destructor is a special method and will be called when the object is destroyed.
Sealed types means it can not has sub-class.

1.2.
Value Type
// E.g. int i = 2;
Value type stores its value directly on stack
which is used for static memory allocation
when the program is compiled.
Value types are destroyed immediately after the scope is lost.

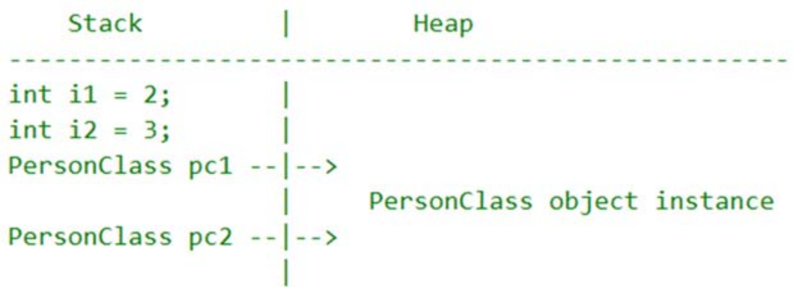
1.3.
Reference Type
// E.g. ClassA a1 = new ClassA()
RHS: Use ClassA as template to create its instance object, this is the "Value".
LHS: declare a1 as "Object Reference Variable" with the type ClassA.
Reference type stores its "Object Reference Variable" in Stack which is static memory allocation,
and stores its "value" in heap at run time which is for dynamic memory allocation.
Thus, heap is a bit slower than stack.
The heap size is only limited by the size of virtual memory.
The "Object Reference Variable" in LHS is destroyed after the scope is lost.
The object "value" in RHS is later destroyed by garbage collector.

1.4.
A struct is a value type.
When copy struct1 into struct2.
The changes in struct2 will not affect struct1,

because the value of struct1 and struct2 are
stored in different memory location in Stack.
struct can't have destructor.
struct can have constructor with parameters,
but can not have constructor without any parameter.
Struct can not extend other Class, but can implement Interface.
Struct is sealed type can not be extended by other Class or other Struct.
E.g.
int (System.Int32), double(System.Double) etc.

1.5.
A class is a reference type.
When copy Class1 into Class2.
The changes in Class2 will reflect back Class1,
because both "Object Reference Variable" (LHS) of Class1 and Class2 are stored in Stack,
but both "Object Reference Variable" are pointing to the same heap location
which stored its "Object Value" (RHS)
Class can have destructor.
Class can have any kind of constructor.
Class can extend other Class, also can implement Interface.
Sealed keyword can prevent a class from being inherited.

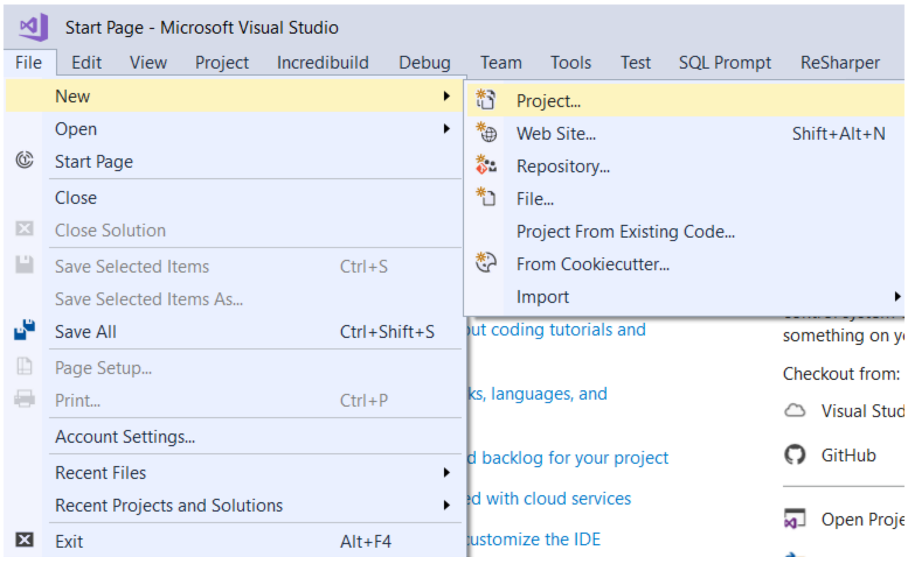
2.
Value Type V.S. Reference Type
E.g.
```
//int i1 = 2;
//int i2 = i1;
//i2++;
...
//PersonClass pc1 = new PersonClass();
//pc1.Id = 1;
//pc1.Name = "Name01";
//PersonClass pc2 = pc1;
//pc2.Id = 2;
//pc2.Name = "Name02";
```

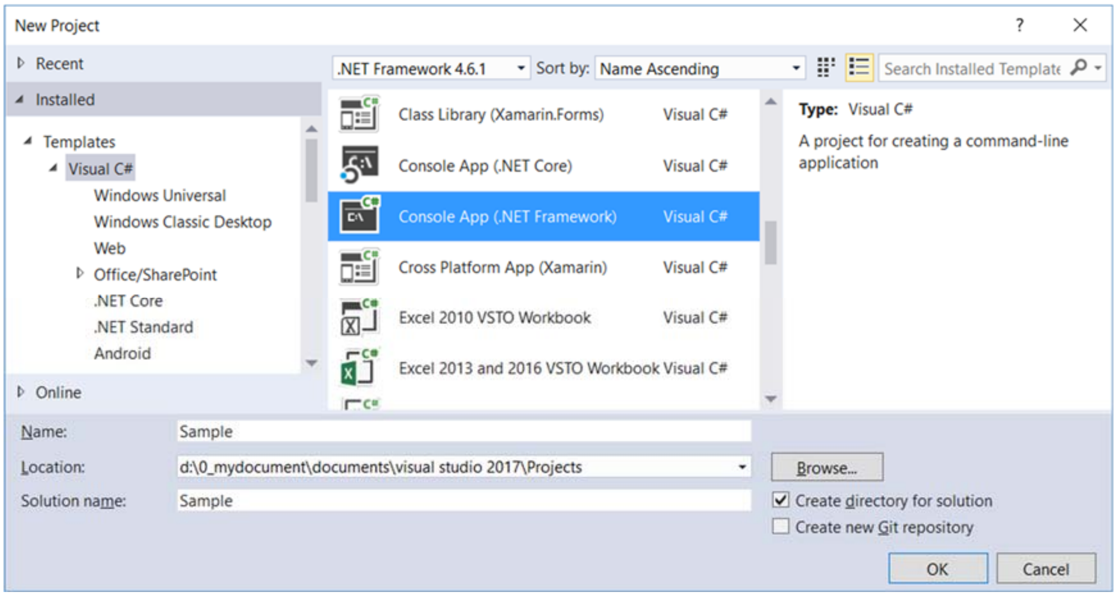

```
    Stack           |        Heap
-----------------------------------------------------
int i1 = 2;         |
int i2 = 3;         |
PersonClass pc1 --|-->
                    |    PersonClass object instance
PersonClass pc2 --|-->
                    |
```

==

1. Create New Project

File --> New --> Project... -->
Visual C# --> **Console App (.Net Framework)** -->
Name: **Sample**




New Project
Recent
Installed
Templates
Visual C#
Windows Universal
Windows Classic Desktop
Web
Office/SharePoint
.NET Core
.NET Standard
Android
Online
.NET Framework 4.6.1
Sort by: Name Ascending
Search Installed Template
Class Library (Xamarin.Forms) Visual C#
Console App (.NET Core) Visual C#
Console App (.NET Framework) Visual C#
Cross Platform App (Xamarin) Visual C#
Excel 2010 VSTO Workbook Visual C#
Excel 2013 and 2016 VSTO Workbook Visual C#
Browse...
OK
Cancel

Type: Visual C#

A project for creating a command-line application

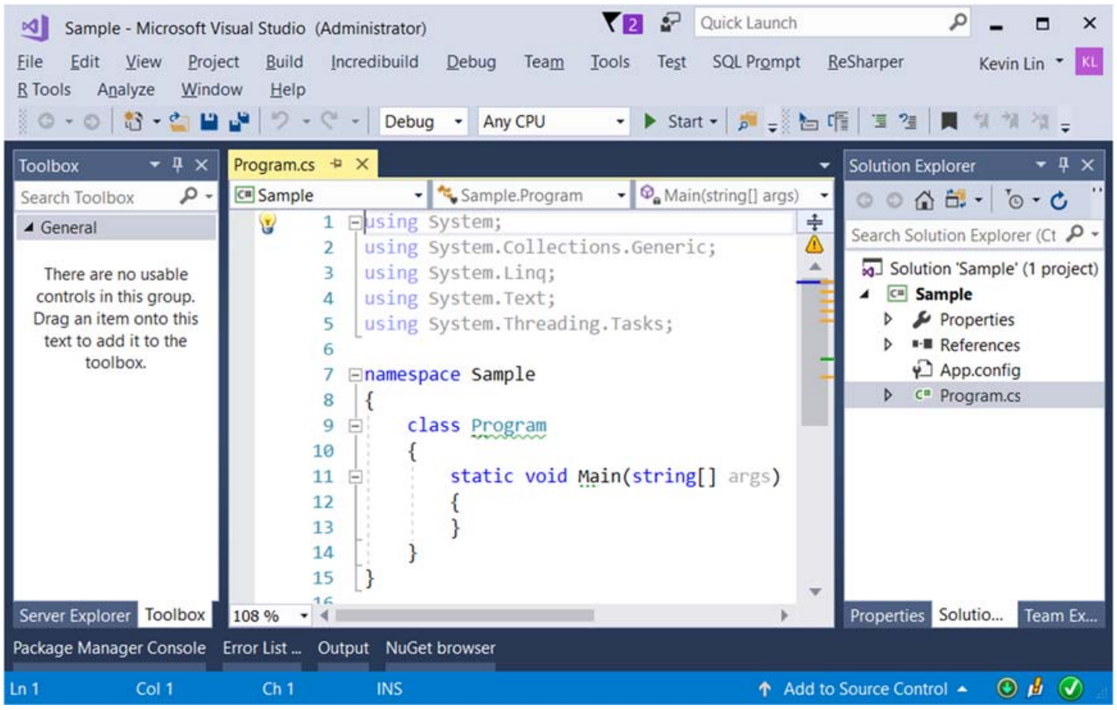
Name: Sample

Location: d:\0_mydocument\documents\visual studio 2017\Projects

Solution name: Sample

☑ Create directory for solution

☐ Create new Git repository


Sample - Microsoft Visual Studio (Administrator)
Quick Launch
File Edit View Project Build Incredibuild Debug Team Tools Test SQL Prompt ReSharper
Kevin Lin
R Tools Analyze Window Help
Debug
Any CPU
Start
Toolbox
Search Toolbox
General
There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.
Program.cs
Sample
Sample.Program
Main(string[] args)
Solution Explorer
Search Solution Explorer (Ct
Solution 'Sample' (1 project)
Sample
Properties
References
App.config
Program.cs
Server Explorer Toolbox
108 %
Properties Solutio... Team Ex...
Package Manager Console Error List ... Output NuGet browser
Ln 1 Col 1 Ch 1 INS
Add to Source Control

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

namespace Sample
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

==

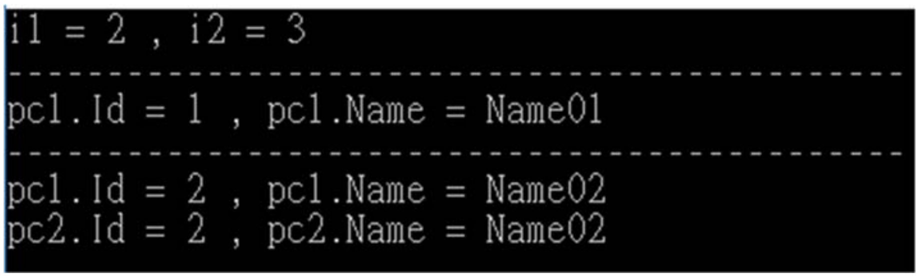
2. Program

```
using System;
namespace Sample
{
    class Program
    {
        static void Main(string[] args)
        {
            int i1 = 2;
            int i2 = i1;
            i2++;
            Console.WriteLine("i1 = {0} , i2 = {1}", i1, i2);
            // i1 = 2 , i2 = 3
            Console.WriteLine("---------------------------------------------");
            PersonClass pc1 = new PersonClass();
            pc1.Id = 1;
            pc1.Name = "Name01";
            Console.WriteLine("pc1.Id = {0} , pc1.Name = {1}", pc1.Id, pc1.Name);
            // pc1.Id = 1 , pc1.Name = Name01
            Console.WriteLine("---------------------------------------------");
            PersonClass pc2 = pc1;
            pc2.Id = 2;
            pc2.Name = "Name02";
            Console.WriteLine("pc1.Id = {0} , pc1.Name = {1}", pc1.Id, pc1.Name);
            Console.WriteLine("pc2.Id = {0} , pc2.Name = {1}", pc2.Id, pc2.Name);
            //pc1.Id = 2 , pc1.Name = Name02
            //pc2.Id = 2 , pc2.Name = Name02
            Console.ReadLine();
        }
    }
}

public interface IPersonA
{
}
public interface IPersonB
{
}
public class ClassA
{
}
public class PersonClass : ClassA, IPersonA, IPersonB
{
    // Private Fields ------------------------
    private int _id;
    private string _name;
    // Constructor ------------------------
    public PersonClass(int id, string name)
    {
        _id = id;
        this._name = name;
    }
    public PersonClass()
    {
        _id = 0;
        _name = "";
    }
    // Destructor ------------------------
    ~PersonClass()
    {
        _id = 0;
        _name = "";
    }
    // Public Properties ------------------------
    public int Id
    {
        get { return _id;}
        set { _id = value;}
    }
    public string Name
    {
        get { return _name; }
        set { _name = value; }
    }
    // Method ------------------------
    public void PrintName()
    {
        Console.WriteLine("Id = {0} && Name = {1}", _id, _name);
    }
}

////Struct can not extend other Class, but can implement Interface.
//public struct PersonStruct : ClassA, IPersonA, IPersonB   //compile error
public struct PersonStruct : IPersonA, IPersonB
{
    // Private Fields ------------------------
    private int _id;
    private string _name;
    // Constructor ------------------------
    //struct can have constructor with parameters,
    //but can not have constructor without any parameter.
    public PersonStruct(int id, string name)
    {
        _id = id;
        _name = name;
    }
    //public PersonStruct()
    //{
    //    _id = 0;
    //    _name = "";
    //}
    // Destructor ------------------------
    //// struct can't have destructor.
    //~PersonStruct()
    //{
    //    _id = 0;
    //    _name = "";
    //}
    // Public Properties ------------------------
    public int Id
    {
        get { return _id; }
        set { _id = value; }
    }
    public string Name
    {
        get { return _name; }
        set { _name = value; }
    }
    // Method ------------------------
    public void PrintName()
    {
        Console.WriteLine("Id = {0} && Name = {1}", _id, _name);
    }
}

////Struct is sealed type can not be extended by other Class or other Struct.
//public class ClassB : PersonStruct
//{
//}
public sealed class ClassB : PersonClass
{
}
////Sealed keyword can prevent a class from being inherited.
//public class ClassC : ClassB
//{
//}
```



```
i1 = 2 , i2 = 3
----------------------------------------------
pc1.Id = 1 , pc1.Name = Name01
----------------------------------------------
pc1.Id = 2 , pc1.Name = Name02
pc2.Id = 2 , pc2.Name = Name02
```