C# (Cont…)

Main Content

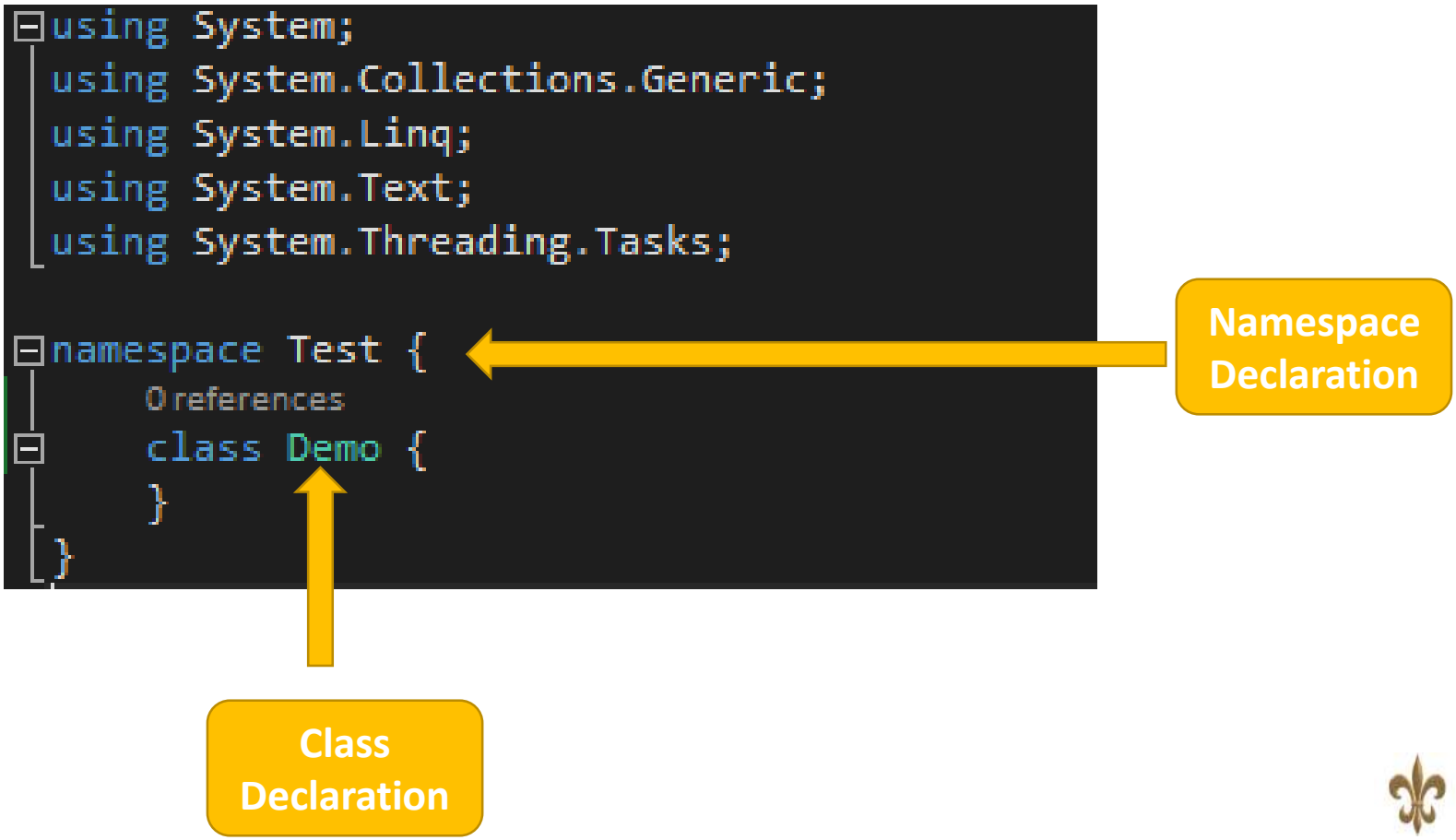
Part 1
- Some objects in C#

Part 2
- Arrays in C#

Part 3
- String and StringBuilder in C#

Part 4
- Some mid-term seminar topics

Basics of objects in C#



Basics of objects in C#

- Class Declaration

```
namespace namespace1 {
    class ClassName {
        //variable

        //constructor

        // getter setter

        //method for class
    }
}
```

Declare variables, getters and setters

- Syntax:
  - [access modifier] + [data type] +[variable name];

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

namespace Test {
    0 references
    class Demo {
        public String name;
    }
}
```

- Declare variables with getters and setters
- Syntax :
  - [access modifier] + [data type] +[variable name]{get;set;}

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

namespace Test {
    0 references
    class Demo {
        0 references
        public String name {set;get;}
    }
}
```

- Declare variables and intervene in getter setter
- Syntax:

```
private + [data type] +_[variable name];
public  + [data type] + [variable name]{
    get{
            return +  _[variable name];
    }
    set{
            this. +  _[variable name]  = value;
    }
}
```

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

namespace Test {
    0 references
    class Demo {
        private String _name;
        0 references
        public String name {
            set {
                this._name = value;
            }
            get {
                return this._name;
            }
        }
    }
}
```

Constructor and Method

Constructor

- Syntax:

```
[access modifier] + [class name]([list parameter]){
        //code of constructor
}
```

Method

- Syntax :

```
[access modifier] +[return type]+ [method name] ([list parameter]){{
        //code of method
}
```

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

namespace Test {
    3 references
    class Demo {
        2 references
        public String name {set;get;}
        1 reference
        public Demo(String name) {
            this.name = name;
        }
        1 reference
        public bool isSameName(String name) {
            return this.name.Equals(name);
        }
    }
}
```

Namespace

- **Namespace** is used to identify classes and group classes together. This makes it easier for programmers to manage code, avoiding confusion.
- **Namespace** helps reduce the complexity of inheriting code from another application.
- To use C# **namespace**, use the keyword **"using"** to declare.

❖Syntax for creating **namespace**

```
namespace Name{
    class A {

    }
    class B {

    }
}
```

❖Syntax for creating **namespace**

```
namespace name.spacea{
    class A {

    }
    class B {

    }
}
```

```
namespace name {
    namespace spacea {
        class A {

        }
        class B {

        }
    }
}
```

Declare inner class

❖Syntax :

```
class Container
{
    class Nested
    {
        // Add code here.
    }
}
```

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

namespace Test {
    2 references
    class Demo {
        1 reference
        class inner {
            1 reference
            public int age { set; get; }
            0 references
            public inner(int age) {
                this.age = age;
            }
        }
    }
}
```

Inherit in C#

- Syntax:
  - class DerivedClass:BaseClass{}

```
namespace Test {
    2 references
    class Demo:super {

    }
    1 reference
    abstract class super {

    }
}
```

Method

- Abstract method:
  - Syntax:[modifier] abstract [return type] [name]();
- Override method:
  - Syntax :[modifier] override [return type] [name](){}

```
namespace Test {
    2 references
    class Demo:super {
        1 reference
        public override int abc() {
            return 1;
        }
    }
    1 reference
    abstract class super {
        1 reference
        public abstract int abc();
    }
}
```

Static member

- There are 4 main types of static members:
  - Static variable.
  - Static method.
  - static class.
  - Static constructor.

Static Variable

❖Static variable

❖ Syntax:[modifier] **static** [data type] [variable name]= [initialization value];

❖Ex:

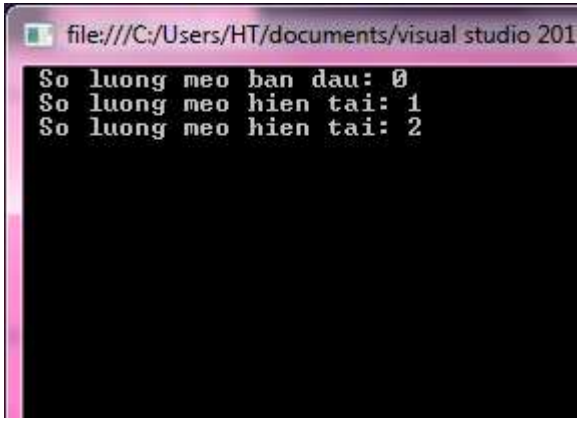
- We want to manage the number of cats we have (assuming 1 object created is 1 cat)

```
public static int Count = 0;
public Cat()        {
weight = 20;
height = 500;
/* Vì constructor chỉ được gọi khi có đối tượng được tạo ra nên ta sẽ tăng Count ở đây */
Count++;
}
}
```

Static Variable

❖Testing: In main() Method:

```
Console.WriteLine(" So luong meo ban dau: " + Cat.Count);
Cat BlackCat = new Cat(); // Tạo ra con mèo đầu tiên
Console.WriteLine(" So luong meo hien tai: " + Cat.Count);
Cat WhiteCat = new Cat(); // Tạo ra con mèo thứ
Console.WriteLine(" So luong meo hien tai: " + Cat.Count);
}
}
```



Static method

❖Static method

- Syntax: [modifier] **static** [return type] [method name](){}

```
class TienIch
   {
      /* Khai báo và định nghĩa 1 phương thức tính luỹ thừa 2 số nguyên */
      public static long LuyThua(int CoSo, int luyThua)
      {
         long KetQua = 1;
         for (int i = 0; i < luyThua; i++)
         {
            KetQua *= CoSo;
         }
         return KetQua;
      } }
```

Arrays

❖1-DIMENSIONAL Array IN C#

- **Arrays in C#**
  - Collection of objects with the same data type.
  - Each object in the array is called an element.
  - Elements are distinguished from each other by element index.
  - In C# element indices are non-negative integers and start from 0 1 2 3...
- **Characteristics of the array:**
  - Elements in the array share a common name and are accessed through the element index.
  - An array needs to limit the number of elements it can contain.
  - Memory must be allocated before the array can be used.
  - The memory locations of the elements in the array are allocated adjacent to each other.

Arrays (cont..)

❖1-DIMENSIONAL Array IN C#

- **Declaring a 1-dimensional array C#**

  <data type> [] **<array name>**;

  VD: String[] Animals= **new String**[3];

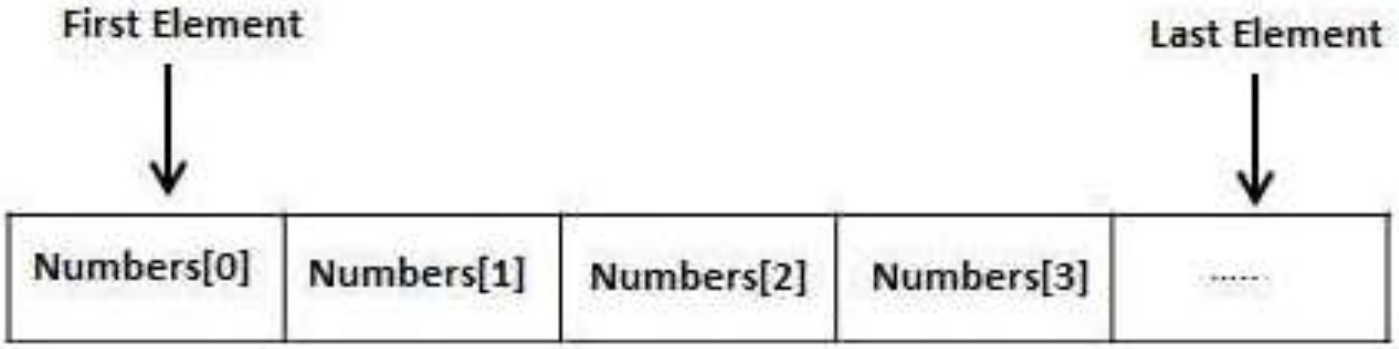
  - **<data type>**[] **<array name>** = **new** <**data type**>**[]** { <value1>, …, <valuen> };
  - **/* Khai báo, cấp phát và khởi tạo giá trị cho mảng */**

  VD: String[] Animals = **new String**[] { "**Dog**", "**Bird**","**Rabbit**" };

Arrays (cont..)

❖1-DIMENSIONAL Array IN C#

- Index of 1-dimensional array



After the array is allocated memory, the elements in the array will have default values:

For integers it is 0
For real numbers it is 0.0
For character type is '' (empty character)
For reference types it is null

Arrays (cont..)

❖1-DIMENSIONAL Array IN C# (properties and methods)

Attribute and method names	Meaning
Length	Thuộc tính: Số phần tử tối đa của mảng (int)
GetLength(<Số chiều>)	Số phần tử trong chiều đã xác định (Chiều là các số từ 0 trở đi VD: mảng 1 chiều thì chiều là 0
Sort()	Phương thức thực hiện sắp xếp mảng theo thứ tự
Clear()	Phương thức xoá hết giá trị trong mảng và đưa nó về giá trị mặc định của kiểu. Lưu ý: Chỉ xoá giá trị, vùng nhớ vẫn còn đó, có thể tiếp tục sử dụng
Copy()	Phương thức thực hiện việc copy giá trị của mảng ra một vùng nhớ mới
Reverse()	Phương thức thực hiện việc đảo ngược thứ tự của mảng một chiều

2-Dimensional array C#

- **Characteristics of 2-dimensional array::**
  - Elements in a 2-dimensional array are accessed through 2 element indices (called row index and column index).
  - Rows and columns are numbered from 0 and increasing. Each element is the intersection of the corresponding row and column, and we use that row and column index to access the element of the 2-dimensional array.
  - **For example:** A[1, 2] is the way to access the element in the 2nd row of the 3rd column (because the index is numbered from 0).

	0	1	2	3	4	5
0	A[0,0]	A[0,1]	A[0,2]	A[0,3]	A[0,4]	A[0,5]
1	A[1,0]	A[1,1]	A[1,2]	A[1,3]	A[1,4]	A[1,5]
2	A[2,0]	A[2,1]	A[2,2]	A[2,3]	A[2,4]	A[2,5]
3	A[3,0]	A[3,1]	A[3,2]	A[3,3]	A[3,4]	A[3,5]
4	A[4,0]	A[4,1]	A[4,2]	A[4,3]	A[4,4]	A[4,5]

2-Dimensional array C# (cont..)

2-Dimensional array C#

- **Declaring a 2-dimensional array C#**

<data type>[,] <array name> = **new** <data type>[]

{

{ <value col 1>, …, <value col n> },

{ <value row 1>, …, <value row n> }

};

- Consider each line as a 1-dimensional array and initialize as a 1-dimensional array.
- Initial values are enclosed in parentheses {} and separated by commas.

2-Dimensional array C# (cont..)

2-Dimensional array C#

- **Declaring a 2-dimensional array C#**
- Ex:

```
int[,] IntArray =
{{1, 2},
{3, 4},
{5, 6}};
```

- Ex:
  - String[,] arrayS = **new** String[2, 3];
- Ex:
  - String[,] iTNongLam = **new** String[,]
  - {{ "**.NET**", "**Java**","**C#**" },
  - { "**Android**", "**IOS**" }};

2-Dimensional array C# (cont..)

❖2-Dimensional array C#

- Some typical properties and methods of 2-dimensional arrays:

Tên thuộc tính hoặc phương thức	Ý nghĩa
Length	Thuộc tính trả về kiểu số nguyên là số phần tử tối đa của mảng (tích số dòng và số cột của mảng)
getLength(<số chiều>)	Trả về số nguyên (số phần tử trong chiều đã xác định, Lưu ý: chiều của mảng là số nguyên và được đánh dấu từ số 0
Rank	Thuộc tính trả về số nguyên đại diện cho số chiều của mảng
Clone()	Thực hiện copy giá trị của mảng ra một vùng nhớ mới

2-Dimensional array C# (cont..)

❖ 2-Dimensional array C#

For example: Let's try to consider a simple example: write a program that allows you to enter an integer value for any 2-dimensional array and then print the entered array to the screen along with the total of all values in the array.

```
Console.Write(" Moi ban nhap so dong cua mang: ");
int Rows = int.Parse(Console.ReadLine());
Console.Write(" Moi ban nhap so cot cua mang: ");
int Columns = int.Parse(Console.ReadLine());
//Tạo 1 mảng 2 chiều với số dòng và số cột đã nhập
int[,] IntArray = new int[Rows, Columns];
/** Duyệt mảng để nhập giá trị cho các phần tử
* Chủ yếu minh họa cách sử dụng mảng nên bỏ qua các
bước kiểm tra dữ liệu mà ép kiểu trực tiếp
for (int i = 0; i < IntArray.GetLength(0); i++){
for (int j = 0; j < IntArray.GetLength(1); j++){
Console.Write(" Moi ban nhap phan tu IntArray[{0}, {1}] = ", i, j);
IntArray[i, j] = int.Parse(Console.ReadLine());
}}
```

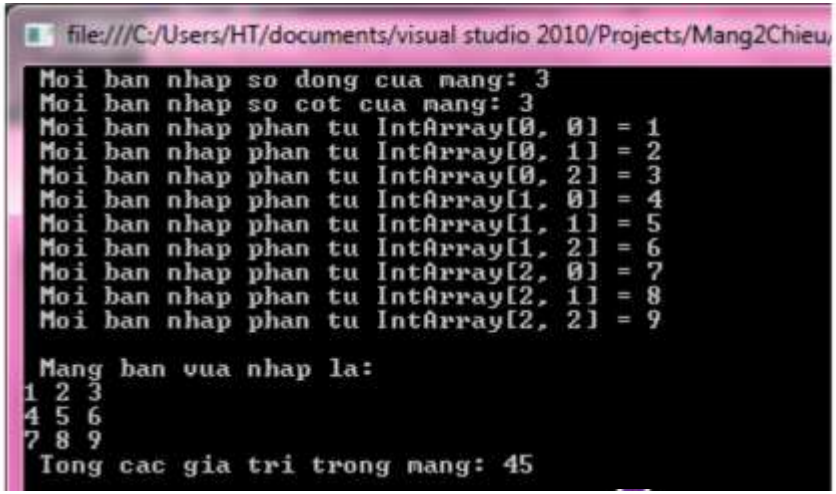
2-Dimensional array C# (cont..)

❖2-Dimensional array C#

```
int Sum = 0;
Console.WriteLine(" \n Mang ban vua nhap la: ");
for (int i = 0; i < IntArray.GetLength(0); i++) {
    for (int j = 0; j < IntArray.GetLength(1); j++){
    Console.Write(IntArray[i, j] + " ");
    Sum = Sum + IntArray[i, j];
    }
Console.WriteLine();}
Console.WriteLine(" Tong cac gia tri trong mang: " +
Sum);
```

2-Dimensional array C# (cont..)

- 2-Dimensional array C#
- **Results after running the program:**

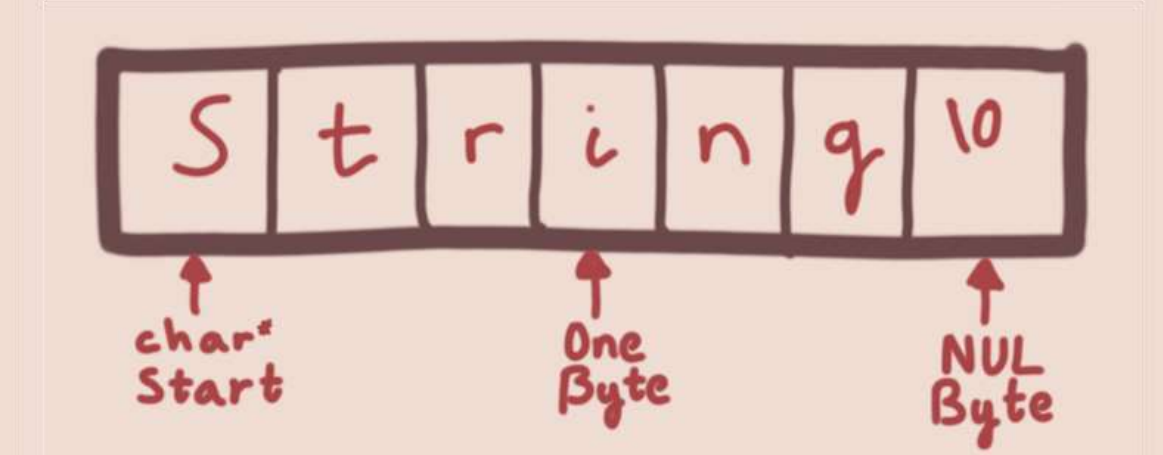


C# String

❖String:

- String class in C#
- Apply the String class to handle strings.
- StringBuilder class in C#

String is a reference data type used to store strings of characters.



C# String (Cont..)

❖Some important properties and methods in the String Class

Properties/ Method	Mean	Note
Length	Trả về độ dài của chuỗi (int)	Thuộc tính của String
String.Compare(String A, String B)	So sánh A và B nếu A=B--> 0, A>B --> 1, A<B -->-1	Có thể sử dụng <tên biến>.CompareTo(String B)
String.Concatenate(String A, String B)	Nối 2 chuỗi A, B thành một chuỗi	Tương tự như cộng chuỗi bằng toán tử +
IndexOf(char value)	Trả về vị trí xuất hiện đầu tiên của char value	Nếu không tìm thấy → -1
Insert(int startIndex, String value)	Trả về chuỗi mới đã chèn thêm value tại vị trí startIndex	
String.isNullOrEmpty(String A)	Kiểm tra chuỗi A có null hoặc rỗng	
LastIndexOf(char a)	Trả về vị trí xuất hiện cuối cùng của a	
toCharArray()	Trả về mảng các ký tự trong chuỗi ban đầu	

C# String (tt)

❖Some important properties and methods in the String Class

Properties/ Methods	Mean	Note
ToLower()	Đổi chuỗi input thành chữ thường	
ToUpper	Đổi chuỗi input thành chữ hoa	
Trim()	Trả về một chuỗi mới đã loại bỏ khoảng trắng ở đầu và cuối chuỗi.	
Remove(int startIndex, int count)	Trả về chuỗi mới đã được gỡ bỏ **count** ký tự từ vị trí startIndex	
Replace(char oldValue, char newValue)	Trả về chuỗi mới đã thay thế ký tự oldValue bằng newValue	
Split(char value)	Trả về mảng các chuỗi được cắt ra dựa trên ký tự value	
SubString(int startIndex, int Length)	Trả về chuỗi mới được cắt từ startIndex với số ký tự cắt là Length	

C# StringBuilder

- The StringBuilder class built in by .NET helps us manipulate the original string directly and saves more memory than the String class.
- Able to automatically expand memory area when needed.
- Do not allow other classes to inherit

C# StringBuilder (tt)

❖Initialize object in StringBuilder

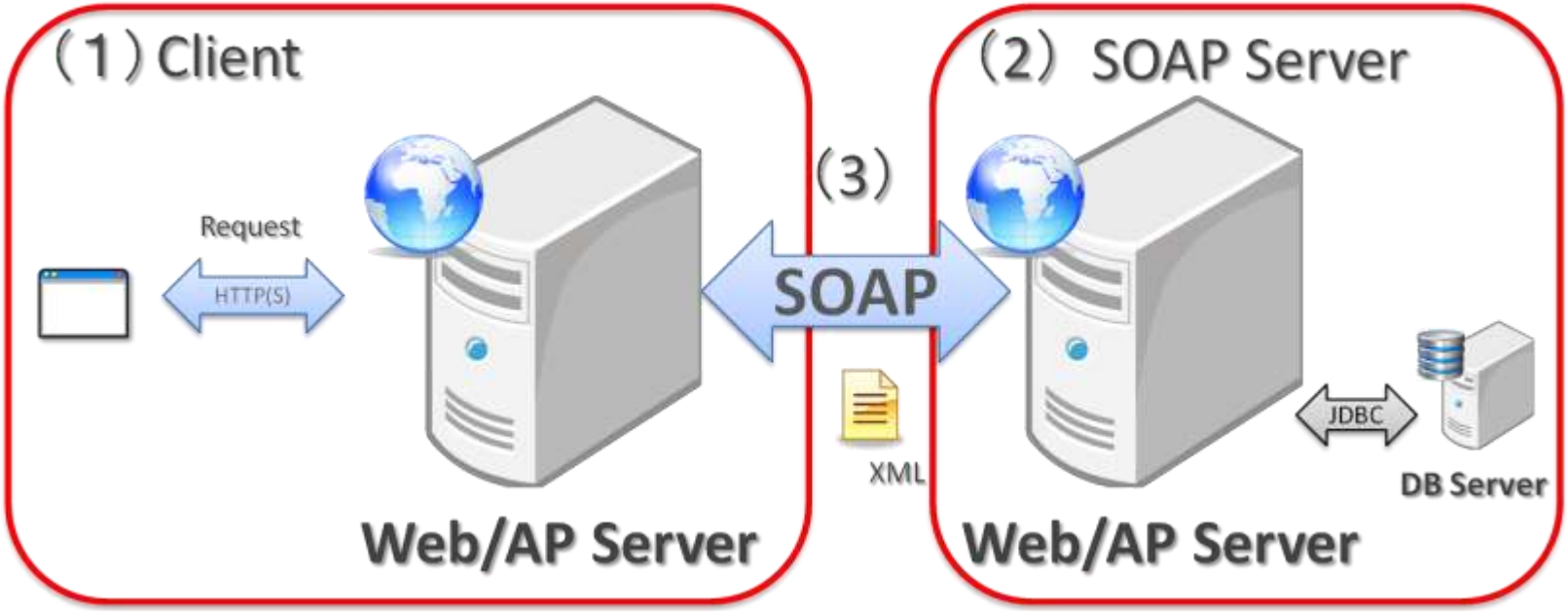
- StringBuilder <variable name> = **new** StringBuilder();
- StringBuilder <variable name> = **new** StringBuilder(<**valueString**>);
- In class StringBuilder have some methods: **Remove**, **Insert**, **Replace** is used exactly the same as class **String**.

Method name	Mean
Append(String value)	Nối chuỗi vào sau chuỗi ban đầu
Clear()	Xoá bỏ toàn bộ nội dung của đối tượng (Lưu ý: vẫn giữ lại vùng nhớ)
ToString()	Chuyển đổi đối tượng StringBuilder sang kiểu String

Các chủ đề seminar giữa kỳ

- Tìm hiểu **Webservice** (Demo) SOAP trên nền tảng .NET
- Tìm hiểu **Xamarin**(Demo) trên nền tảng .NET ngôn ngữ C#
- Tìm hiểu **Windows Presentation Foundation** hay gọi tắt là **WPF** (Demo) C#
- Tìm hiểu **MAUI**(Demo) trên nền tảng .NET ngôn ngữ C#

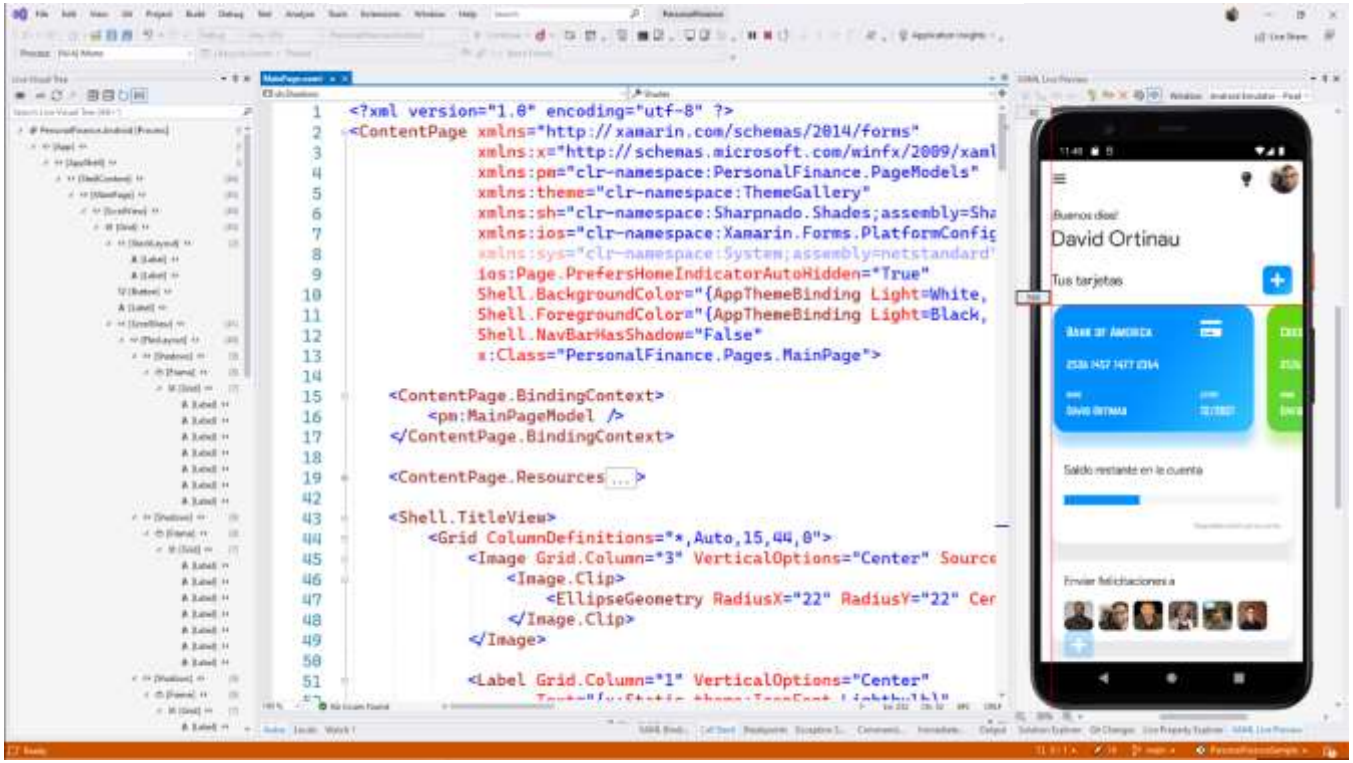
Webservices SOAP



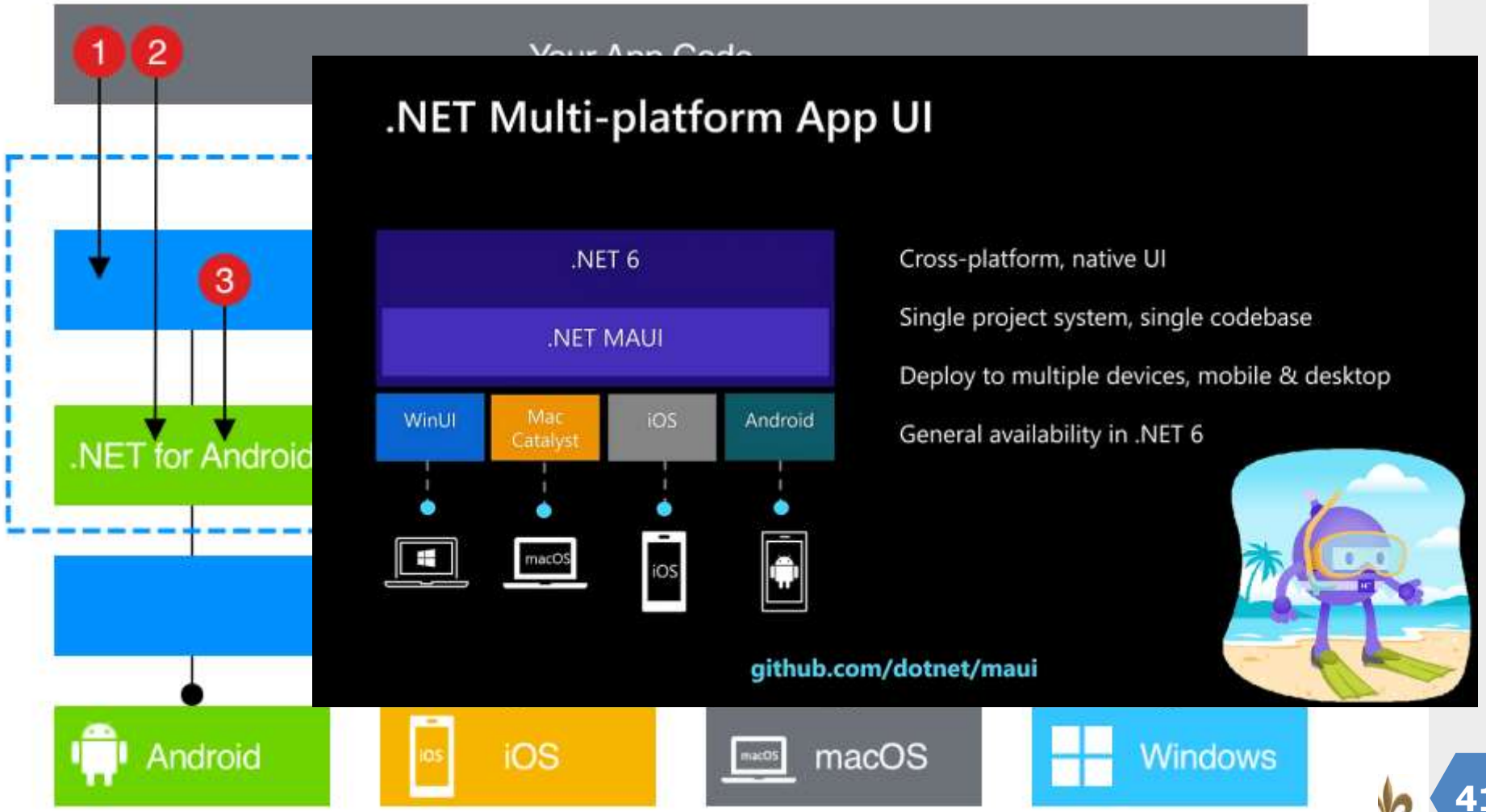
REST FUL API → JSON, GSON

XAMARIN


Xamarin


1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
3 xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml
4 xmlns:pm="clr-namespace:PersonalFinance.PageModels"
5 xmlns:theme="clr-namespace:ThemeGallery"
6 xmlns:sh="clr-namespace:Sharpnado.Shades;assembly=Sha
7 xmlns:ios="clr-namespace:Xamarin.Forms.PlatformConfig
8 xmlns:sys="clr-namespace:System;assembly=netstandard
9 ios:Page.PrefersHomeIndicatorAutoHidden="True"
10 Shell.BackgroundColor="{AppThemeBinding Light=White,
11 Shell.ForegroundColor="{AppThemeBinding Light=Black,
12 Shell.NavBarHasShadow="False"
13 x:Class="PersonalFinance.Pages.MainPage">
14
15 <ContentPage.BindingContext>
16 <pm:MainPageModel />
17 </ContentPage.BindingContext>
18
19 <ContentPage.Resources ...>
42
43 <Shell.TitleView>
44 <Grid ColumnDefinitions="*,Auto,15,44,8">
45 <Image Grid.Column="3" VerticalOptions="Center" Source
46 <Image.Clip>
47 <EllipseGeometry RadiusX="22" RadiusY="22" Cer
48 </Image.Clip>
49 </Image>
50
51 <Label Grid.Column="1" VerticalOptions="Center"
Buenos días!
David Ortinau
Tus tarjetas
Saldo restante en la cuenta
Enviar felicitaciones a

MAUI .NET


1
2
3
.NET for Android
Android
iOS
macOS
Windows
.NET Multi-platform App UI
.NET 6
.NET MAUI
WinUI
Mac Catalyst
iOS
Android
macOS
iOS
Cross-platform, native UI
Single project system, single codebase
Deploy to multiple devices, mobile & desktop
General availability in .NET 6
github.com/dotnet/maui

WPF

WPF, viết tắt của Windows Presentation Foundation, là hệ thống API mới hỗ trợ việc xây dựng giao diện đồ hoạ trên nền Windows.

	Windows Forms	PDF	Windows Forms/ GDI+	Windows Media Player	Direct3D	WPF
Giao diện đồ họa (form và các control)	x					x
On-screen văn bản	x					x
Fixed-format văn bản		x				x
Hình ảnh			x			x
Video và âm thanh				x		x
Đồ họa 2 chiều			x			x
Đồ họa 3 chiều					x	x

Q & A