Ôn tập: Lập trình hướng đối tượng

Thanh Le, Ph.D.
Khoa Công nghệ Thông tin Kinh doanh

July 17, 2018

Nội dung

- Xem lại:
  - Danh sách, từ điển
- Lập trình hướng đối tượng

Danh List và Dictionary

- List là tập hợp các phần tử cùng kiểu
- Dictionary là tập hợp các phần tử cùng kiểu; có thể truy cập các phần tử thông qua mã phần tử (khóa)
- System.Collections.Generic

  Là thư viện .Net cho List và Dictionary

3

UEH

List các phần tử chuỗi

```
using System;
using System.Collections.Generic;

public class Program
{
    public static void Main()
    {
        List<string> list = new List<string>(){ "mot", "hai", "ba" };
        foreach (string s in list)
            Console.WriteLine(s);
    }
}
```

4

UEH

Dictionary: khóa là số nguyên

```
using System;
using System.Collections.Generic;

public class Program
{
    public static void Main()
    {
        Dictionary<int, string> dict = new Dictionary<int, string>()
        {
            {1, "Một"}, {2, "Hai"}, {3, "Ba"}
        };
        dict.Add(4,"số bốn");

        foreach (int key in dict.Keys) {
            Console.Write("\n{0} => {1}", key, dict[key]);
        }
    }
}
```


UEH

Dictionary: khóa là chuỗi

```
using System;
using System.Collections.Generic;

public class Program
{
    public static void Main()
    {
        Dictionary<string, string> dict = new Dictionary<string, string>()
        {
            {"one", "Một"}, {"two", "Hai"}, {"three", "Ba"}
        };
        dict.Add("four","số bốn");

        foreach (string key in dict.Keys) {
            Console.Write("\n{0} => {1}", key, dict[key]);
        }
    }
}
```


UEH

Hàm List và Dictionary

- List
  - Add(phần-tử-mới)
  - Remove(giá-trị-xóa)
  - Insert(vị-trí, giá-trị);
- Dictionaty
  - Add(khóa-mới, gia-trị-mới)
  - Remove(khóa-phần-tử-xóa)

7

UEH

Cập nhật List

```
using System;
using System.Collections.Generic;

public class Program
{
    public static void Main()
    {
        List<string> list = new List<string>(){ "mot", "hai", "ba" };

        list.Add("bon");
        foreach (string s in list)
            Console.WriteLine(s);

        Console.WriteLine();
        list.Remove("ba");
        foreach (string s in list)
            Console.WriteLine(s);
    }
}
```

8

UEH

Chèn List

```
/*
 * Tạo 1 List các kiểu string và thêm 2 phần tử vào List.
 */
List<string> MyList4 = new List<string>();
MyList4.Add("Free");
MyList4.Add("Education");

// In giá trị các phần tử trong List
Console.WriteLine(" List ban dau: ");
Console.WriteLine(" So luong phan tu trong List la: {0}", MyList4.Count);
foreach (string item in MyList4)
{
    Console.Write(" " + item);
}
Console.WriteLine();

// Chèn 1 phần tử vào đầu List.
MyList4.Insert(0, "HowKteam");
```


UEH

Cập nhật Dictionary

```
using System;
using System.Collections.Generic;

public class Program
{
    public static void Main()
    {
        Dictionary<string, string> dict = new Dictionary<string, string>()
        {
            {"one", "Một"}, {"two", "Hai"}, {"three", "Ba"}
        };

        dict.Add("four","số bốn");
        foreach (string key in dict.Keys) {
            Console.Write("\n{0} => {1}", key, dict[key]);
        }


        Console.WriteLine();
        dict.Remove("three");
        foreach (string key in dict.Keys) {
            Console.Write("\n{0} => {1}", key, dict[key]);
        }
    }
}
```


UEH

Ví dụ: xử lý mảng 1 chiều

```
public static void Main()
{
    int []  n = new int[10]; /* n la mot mang gom 10 so nguyen */
    int i,j;
    /* khoi tao cac phan tu cua mang n */
    for (i = 0; i < 10; i++)
    {
        n[i] = i + 100;
    }

    /* hien thi gia tri cac phan tu cua mang n */
    for (j = 0; j < 10; j++)
    {
        Console.WriteLine("Phan tu [{0}] = {1}", j, n[j]);
    }
}
```


UEH

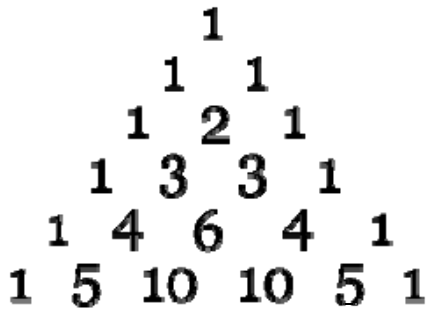
Ví dụ: xử lý mảng 1 chiều

```
public static void Main()
{
    int []  n = new int[5]; /* n la mot mang gom 10 so nguyen */
    int i,j;
    /* khoi tao cac phan tu cua mang n */
    for (i = 0; i < 5; i++)
    {
        n[i] = i + 100;
    }

    /* hien thi gia tri cac phan tu cua mang n */
    for (j = 0; j < 5; j++)
    {
        Console.WriteLine("Phan tu [{0}] = {1}", j, j+n[j]);
    }
}
```


UEH

List và đệ qui: tam giác pascal



UEH

Tam giác pascal đệ qui

```
using System;
using System.Collections.Generic;

public class Program
{
    public static void Main()
    {
        // dòng thứ 1 trên tam giác pascal
        List<int> ps = pascal(1);

        // dòng thứ 5 trên tam giác pascal
        ps = pascal(6);
        foreach (int m in ps)
            Console.Write("{0} ", m);
    }

    static List<int> pascal(int N) {
        if (N == 1) return new List<int>(){1};
        List<int> pascalN_1 = pascal(N-1);
        List<int> pascalN = new List<int>();
        pascalN.Add(1);
        for (int i=1; i<N-1; i++)
            pascalN.Add( pascalN_1[i-1]+pascalN_1[i] );
        pascalN.Add(1);
        return pascalN;
    }
}
```


UEH

Lập trình hướng đối tượng

- Lập trình hướng đối tượng (Object-Oriented Programming - OOP) là phương pháp lập trình dựa trên sự phối hợp các đối tượng khách quan trong không gian bài toán; chúng hoạt động và tương tác lẫn nhau để đưa bài toán về trạng thái mong muốn.
- Phương pháp tiếp cận này cho phép tách không gian bài toán thành các tập hợp các đối tượng. Đối tượng có tính độc lập, và duy nhất. Từ một công việc lớn phức tạp ta có thể phân chia thành nhiều công việc nhỏ đơn giản và dễ thực hiện hơn, đồng thời việc điều chỉnh bổ sung tính năng cho mỗi đối tượng cũng không ảnh hưởng đến hoạt động của các đối tượng khác. Nhờ đó, khi có nhu cầu phát triển bài toán thì ta không phải xây dựng lại từ đầu mà có thể dựa trên những nội dung đã có. Đây chính là một ưu điểm lớn so với lập trình cấu trúc.

15

Lớp

- Lớp là một tập hợp các đối tượng có cùng một số tính chất khảo sát. Các tính chất này có thể là trạng thái (thuộc tính) hay hành vi của đối tượng.
- Ví dụ: Lớp cá là tập hợp bao gồm những động vật có cùng đặc điểm: vây, mang, máu lạnh; có các hành vi: sống dưới nước, thở bằng mang, đẻ trứng.

16

Đối tượng

- Đối tượng là một thể hiện cụ thể của lớp. Một đối tượng thuộc lớp nào sẽ có những thuộc tính, hành vi của lớp đó.
- Sự khác nhau về thuộc tính, cách thể hiện hành vi của mỗi đối tượng là cơ sở phân biệt nó với các đối tượng khác trong cùng một lớp.

Đặc tính lớp đối tượng

- **Thuộc tính**: Thuộc tính là giá trị phản ánh trạng thái của đối tượng tại một thời điểm xác định. Đối tượng có một hoặc nhiều thuộc tính.
- **Hành vi**: là cách ứng xử của đối tượng với môi trường xung quanh, làm thay đổi thuộc tính của bản thân đối tượng và các đối tượng liên quan.
  - Hoạt động sống của con người là quá trình vận động thay đổi bản thân con người và tác động (cũng như bị tác động) đến môi trường xung quanh.
- Đối tượng = Thuộc tính + Hành vi

Chương trình OOP

- Theo tiếp cập OOP, chương trình là chuỗi thao tác phối hợp của một hay nhiều đối tượng, theo một kịch bản thiết kế trước, nhằm đạt đến một kết quả mong muốn.
- Lập trình cấu trúc (Niklaus Wirth, 1976):
  - Chương trình = Cấu trúc DL + Giải Thuật
- Lập trình hướng đối tượng (*Le, 2002)
  - Chương trình = Các đối tượng + Kịch bản

*Lập trình về Windows với MFC Microsoft Visual C++ 6.0

19

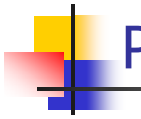
UEH

Đặc điểm OOP

- **Tính khách quan (Objective)**: Lớp đối tượng hình thành từ một chủ thể khách quan. Kịch bản của chương trình vận dụng những yếu tố khách quan này một cách hợp lý để đạt được mục đích của chương trình.
- **Tính đóng gói (Encapsulation)**: Việc bao hàm các thuộc tính và hành vi trong đối tượng giúp đối tượng có thể hoạt động độc lập cũng như phối hợp với các lớp đối tượng khác.
- **Tính kế thừa (Inheritance)**: Việc định nghĩa một lớp đối tượng dựa trên các lớp đối tượng đã có gọi là sự kế thừa. Các lớp đã có gọi là lớp cơ sở (based class), lớp được định nghĩa gọi là lớp kế thừa (derived class). Bên cạnh việc kế thừa, lớp kế thừa có thể bổ sung những hành vi và thuộc tính cần thiết để phục vụ cho nhu cầu mới phát sinh. Cơ chế đó đã tạo sự phát triển cho ứng dụng mà đối tượng tham gia.
- **Tính đa hình (Polymorphism)**: Cơ chế triển khai nhiều cách thể hiện khác nhau cho một hành vi theo nguyên tắc xây dựng lớp cơ sở chứa hành vi cần triển khai và các lớp kế thừa. Mỗi lớp kế thừa sẽ phát triển hành vi đó theo một cách khác nhau.

20

UEH



Phân loại thuộc tính/hành vi

- **Public**: Các thuộc tính, hành vi được đối tượng thể hiện bên ngoài.
- **Protected**: Các thuộc tính, hành vi ẩn chứa bên trong đối tượng, hỗ trợ các hoạt động của đối tượng và có thể truyền lại cho các lớp đối tượng kế thừa.
- **Private**: Các thuộc tính, hành vi ẩn chứa bên trong đối tượng, hỗ trợ các hoạt động của đối tượng và không thể truyền lại cho các lớp đối tượng kế thừa.

Khai báo lớp

```
class  TEN_LOP {

        // khai báo thuộc tính (như khai báo biến)
        [public/protected/private] khai-báo-thuộc-tính;

        // Khai báo hành vi (như khai báo hàm)
         [public/protected/private] khai-báo-hành-vi;

        // hành vi khởi tạo của lớp
        public TEN_LOP([danh sách tham số]) { }
        // hành vi kết thúc của lớp
        ~TEN_LOP([danh sách tham số]) { }

}
```


VD Lớp: duongtron

```
using System;

public class Program
{
    public static void Main()
    {
        duongtron vt = new duongtron(10);
        Console.WriteLine( "Chu vi: {0}", vt.chuvi() );
        vt = new duongtron(5);
        Console.WriteLine( "Chu vi: {0}", vt.chuvi() );
    }

}

class duongtron {
    protected const double pi = 3.14;
    protected double bankinh;
    public duongtron(double initBK) {
        bankinh = initBK;
    }
    ~duongtron() { bankinh  = 0; }
    // tinh dien tich
    public double chuvi() { return 2 * bankinh * pi; }
}
```

23

UEH

VD2: duongtron2 kế thừa

```
public class Program
{
    public static void Main()
    {
        duongtron vt = new duongtron(10);
        Console.WriteLine( "Chu vi: {0}", vt.chuvi() );

        duongtron2 vt2 = new duongtron2(5);
        Console.WriteLine( "Chu vi: {0}", vt2.chuvi() );
        Console.WriteLine( "Dien tich: {0}", vt2.dientich() );
    }

}

class duongtron {
    protected const double pi = 3.14;
    protected double bankinh;
    public duongtron(double initBK) {
        bankinh = initBK;
    }
    ~duongtron() { bankinh  = 0; }
    // tinh dien tich
    public double chuvi() { return 2 * bankinh * pi; }
}

class duongtron2 : duongtron {
    public duongtron2(double bk) : base(bk) {}
    public double dientich() { return bankinh * bankinh * pi; }
}
```

24

UEH

Lỗi do dòng nào?

```
1  using System;
2
3  public class Program
4  {
5      public static void Main()
6      {
7          duongtron vt = new duongtron(10);
8          Console.WriteLine( "Chu vi: {0}", vt.chuvi() );
9
10         duongtron2 vt2 = new duongtron2(5);
11         Console.WriteLine( "Chu vi: {0}", vt2.chuvi() );
12         Console.WriteLine( "Dien tich: {0}", vt2.dientich() );
13     }
14
15 }
16
17 class duongtron {
18     private const double pi = 3.14;
19     protected double bankinh;
20     public duongtron(double initBK) {
21         bankinh = initBK;
22     }
23     ~duongtron() { bankinh  = 0; }
24     // tinh dien tich
25     public double chuvi() { return 2 * bankinh * pi; }
26 }
27
28 class duongtron2 : duongtron {
29     public duongtron2(double bk) : base(bk) {}
30     public double dientich() { return bankinh * bankinh * pi; }
31 }
```


UEH

Lớp timeseries

```
public class Program
{
    public static void Main()
    {
        timeseries ts = new timeseries(new double[5]{1, 2, 3, 4, 5});
        Console.WriteLine("So phan tu: {0}", ts.Length);
        ts.inra();

        Console.WriteLine();
        ts = new timeseries(ts.mTB());
        ts.inra();
    }

}

class timeseries {
    protected double[] data;
    public int Length;
    public timeseries(double[] initData) {
        data = initData;
        Length = initData.Length;
    }
    ~timeseries() { data = null; }
    public void inra() {
        foreach (double item in data) Console.Write("{0:#.00} ", item);
    }
    public double[] mTB() {
        double[] kq = (double[]) data.Clone();
        for (int i=1; i<kq.Length; i++) kq[i] = (data[i-1]+data[i])/2;
        return kq;
    }
}
```


UEH

Xin cảm ơn!