Bài 3

Xây dựng lớp và các thành phần của lớp

Mục tiêu

- Lớp và khai báo lớp
- Tạo đối tượng
- Thuộc tính
- Phương thức
- Truy xuất các thành phần bên trong đối tượng
- Biến tĩnh và phương thức tĩnh
- Truyền tham số cho phương thức
- Phương thức khởi tạo và từ khóa “this”
- Phương thức hủy

Lớp (class) và khai báo lớp

- Class là một khái niệm trong lập trình hướng đối tượng mô tả cho những thực thể có chung tính chất và hành vi.
- Class định nghĩa những thuộc tính và hành vi dùng chung cho những đối tượng của lớp đó.
- Khai báo lớp

```
<phạm_vi> class <tên_lớp>
{
        //khai báo các trường
        //Khai báo các thuộc tính
        //Khai báo các phương thức
}
```

Lớp (class) và khai báo lớp

DEVMASTER.VN

Ví dụ

```
public class Category
{
    //khai báo trường
    private int id;
    private string name;
    //khai báo các thuộc tính
    public int Id
    {
        get { return id; }
        set { id = value; }
    }
    public string Name
    {
        get { return name; }
        set { name = value; }
    }
    //khai bao phương thức
    public void Display()
    {
        Console.WriteLine("Id:" + id);
        Console.WriteLine("Name:" + name);
    }
}
```

Lớp (class) và khai báo lớp

DEVMASTER.VN

Một số lưu ý khi khai báo lớp

- Phạm vi của lớp có thể là public(sử dụng ở bất kỳ đâu) hoặc để mặc định là internal (sử dụng trong cùng một assembly)
- Tên các lớp phải là danh từ
- Ký tự đầu của mỗi từ phải là chữ hoa
- Tên phải đơn giản và có ý nghĩa
- Tên không được trùng với từ khóa của C#
- Tên không có khoảng trắng
- Tên không thể bắt đầu với ký tự số, tuy nhiên có thể bắt đầu với “@” hoặc “_”

Đối tượng(Object) và cách tạo

- Đối tượng là một thực thể trong thế giới thực, nó là thể hiện của một lớp.
- Tạo đối tượng

  <tên_lớp> tên_đối_tượng = new <tên_class>();
- Ví dụ

```
Category c = new Category();
```

Thuộc tính (properties) 1-5

- Trong C# thuộc tính là thành phần được sử dụng để truy xuất đến các trường private được khai báo bên trong lớp
- Mỗi thuộc tính truy xuất đến một biến thành viên duy nhất
- **Khai báo lớp thuộc tính**

```
class <Tên_lớp>
{
        //khai bao thuộc tính
        [public | private] Kiểu_dữ_liệu Tên_thuộc_tính
        {
                get{ return tên_trường;}
                set{ tên_trường=value;}
        }
        //khai báo tiếp
}
```

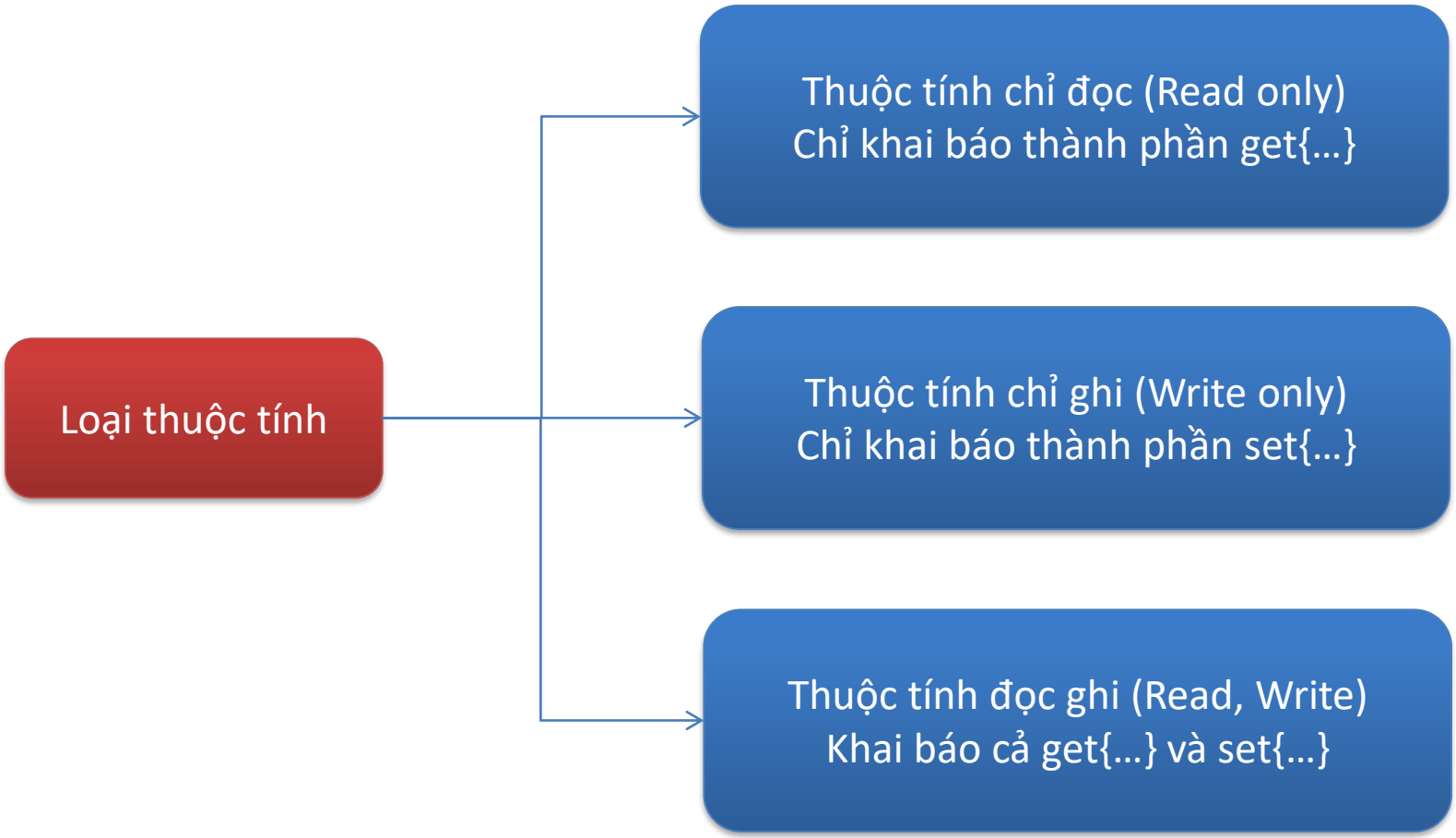
Thuộc tính (properties) 2-5

DEVMASTER.VN

Ví dụ

```
class Employee
{
    //khai báo trường private
    private string id;
    private string fullName;
    private int salary;
    //khai báo các thuộc tính
    public string Id
    {
        get { return id; }
        set { id = value; }
    }
    public string FullName
    {
        get { return fullName; }
        set { fullName = value; }
    }
    public int Salary
    {
        get { return salary; }
        set { salary = value; }
    }
}
```

Thuộc tính (properties) 3-5



Thuộc tính (properties) 4-5

Ví dụ

```
class Employee
{
    //khai báo trường private
    private string id;
    private string fullName;
    private int salary;
    //khai báo các thuộc tính đọc ghi
    public string Id
    {
        get { return id; }
        set { id = value; }
    }
    //thuộc tính chỉ đọc
    public string FullName
    {
        get { return fullName; }
    }
    //thuộc tính chỉ ghi
    public int Salary
    {
        set { salary = value; }
    }
}
```

Thuộc tính (properties) 5-5

- **Khai báo thuộc tính tự động**

```
class <tên_lớp>
{
    public   kiểu_dữ_liệu   tên_thuộc_tính { get; set; }
}
```

Ví dụ

```
class Product
{
    public int Id { get; set; }
    public string Name { get; set; }
    public int Quantity  { get; private set; } //read only
    public int Price { private get; set; } //write onley
}
```

Phương thức (method) 1-2

- Phương thức là các chức năng khai báo bên trong lớp để thực hiện 1 chức cụ thể
- Phương thức có thể có hoặc không có tham số
- phương thức có thể có hoặc không có giá trị trả về

Quy ước đặt tên phương thức

- Không thể là từ khóa trong C#
- Không chứa khoảng trắng
- Không bắt đầu là số
- Có thể bắt đầu với ký tự _ hoặc @

Phương thức (method) 2-2

DEVMASTER.VN

Ví dụ

```
class Student
{
    private string id;
    private string name;
    private double mark1, mark2, mark3;
    //phương thức không tham số và không trả về giá trị
    public void Display()
    {
        Console.WriteLine("Id:" + id);
        Console.WriteLine("Name:" + id);
        Console.WriteLine(" Makr1 {0} \n Mark2 {1} \n Mark3 {3}", mark1, mark2, mark3);
    }
    //phương thức không tham số và trả về giá trị kiểu double
    public double Average()
    {
        double avg = (mark1 + mark2 + mark3) / 3;
        return avg;
    }
    //phương thức có tham số và trả về giá trị kiểu số nguyên
    public int Add(int a, int b)
    {
        return (a + b);
    }
}
```

Truy xuất vào thành phần bên trong đối tượng 1-2

- **Truy xuất vào trường**

  <tên_đối_tượng>.<tên_trường>;

- **Truy xuất vào thuộc tính**

  <tên_đối_tượng>.<tên_thuộc_tính>;

- **Truy xuất vào phương thức**

  <tên_đối_tượng>.<tên_phương_thức>([gt1,gt2,….]);

Truy xuất vào thành phần bên trong đối tượng 2-2

DEVMASTER.VN

Ví dụ

```
class Student
{
    private string id;
    private string name;
    private double mark1, mark2, mark3;
    //khai bao thuộc tính Id
    public string Id...
    //phương thức không tham số và không trả về giá trị
    public void Display()...
    //phương thức không tham số và trả về giá trị kiểu double
    public double Average()...
    //phương thức có tham số và trả về giá trị kiểu số nguyên
    public int Add(int a, int b)...
    public static void Main(string[] args)
    {
        Student st = new Student();
        //truy xuất vào thuộc tính
        st.Id = "S001";
        //truy xuất vào các trường
        st.name = "Tran Tuan Kiet";
        st.mark1 = 10;
        st.mark2 = 9.5;
        st.mark3 = 8;
        //gọi phương thức
        st.Display();
        double avg=st.Average();
        int sum = st.Add(10, 5);
    }
}
```

Phạm vi truy xuất

Cấp độ

Bổ tử truy xuất	Sử dụng ở bất kỳ đâu trong ứng dụng	Sử dụng trong lớp hiện tại	Sử dụng trong lớp dẫn xuất
public	✓	✓	✓
private	x	✓	x
protected	x	✓	✓
internal	x	✓	✓

Biến tĩnh

- Biến tĩnh là một biến đặc biệt được truy cập thông qua tên lớp mà không cần sử dụng đối tượng của lớp. Biến tĩnh được khai báo với từ khóa static, khi biến tĩnh được tạo, nó tự động khởi tạo trước khi được sử dụng, tất cả các đối tượng của lớp đều chia sẻ cùng một biến tĩnh. Đối tượng của lớp không thể truy cập vào biến tĩnh.

Ví dụ

```
class Program
{
    static double rate = 10.4;
    static string name = "Bkap";

    0 references
    static void Main(string[] args)
    {
        Console.WriteLine("Rate {0}", rate);
        Console.WriteLine("Name {1}", Program.name);
    }
}
```

Phương thức tĩnh 1-2

- Mặc định một phương thức được gọi bằng cách sử dụng tên đối tượng của lớp (phương thức thể hiện)
- Tuy nhiên bạn có thể gọi phương thức mà không cần tạo bất kỳ đối tượng nào, đó là phương thức tĩnh.
- Khi khai báo phương sử dụng từ khóa **static**
- Phương thức tĩnh có thể truy cập trực tiếp tới các biến tĩnh
- Tuy nhiên chúng không thể truy cập trực tiếp tới biến không tĩnh.

Phương thức tĩnh 2-2

DEVMASTER.VN

Ví dụ

```
class Calculator
{
    static int Add(int a, int b)
    {
        return (a + b);
    }
    static int Sub(int a, int b)
    {
        return (a - b);
    }
    int Mul(int a, int b)
    {
        return a * b;
    }
    static void Main(string[] args)
    {
        //cách gọi phương thức tĩnh
        int sum = Add(4, 5);
        int sub = Calculator.Sub(4, 5);
        //cách gọi phương thức không tĩnh
        Calculator cal = new Calculator();
        int Mul = cal.Mul(4, 5);

    }
}
```

Truyền tham số cho phương thức 1-2

- **Truyền tham trị**: là cách truyền bản sao của tham số thực(giá trị) cho tham số hình thức, mọi thay đổi của tham số hình thức trong phương thức sẽ không ảnh hưởng tới tham số thực
- **Truyền tham chiếu**: sử dụng từ khóa ref hoặc out, mọi thay đổi của tham số hình thức sẽ ảnh hưởng tới tham số thực
  - Với từ khóa ref thì giá trị của tham số hình thức sẽ giữ lại khi kết thúc phương thức
  - Với từ khóa out giống từ khóa ref nhưng tham số thực sự không cần khởi tạo giá trị ban đầu.

Truyền tham số cho phương thức 2-2

DEVMASTER.VN

Ví dụ

```
class Number
{
    public static void Swap(int a, int b)
    {
        int tg = a;
        a = b;
        b = tg;
    }
    public static void Swap(ref int a, ref int b)
    {
        int tg = a;
        a = b;
        b = tg;
    }
    public static void Cal(out double area, double r)
    {
        area = 2*3.14 * r;
    }
    public static void Main(string[] args)
    {
        int a=10, b=20;
        double area;
        //truyền tham trị
        Swap(a, b);
        Console.WriteLine("a={0},b={1}", a, b);
        //truyền tham chiếu sử dụng từ khóa ref
        Swap(ref a, ref b);
        Console.WriteLine("a={0},b={1}", a, b);
        //truyền tham chiếu sử dụng từ khóa out
        Cal(out area, 5);
        Console.WriteLine("area={0}", area);
    }
}
```

Phương thức khởi tạo 1-3

- Phương thức khởi tạo (Constructor) là phương thức đặc biệt có cùng tên với tên lớp, nó có vai trò khởi tạo các thành viên dữ liệu của đối tượng khi nó được tạo.
- Constructor được gọi ngay sau khi khởi tạo đối tượng bằng lệnh new, và tương ứng với mỗi đối tượng nó chỉ được gọi một lần duy nhất. Có 2 loại constructor, không tham số và có tham số.
- Constructor không tham số dùng cho việc khởi tạo các giá trị mặc định cho các biến khi đối tượng được tạo.
- Constructor có tham số dùng cho việc khởi tạo các giá trị khác nhau cho các biến khi đối tượng được tạo

Phương thức khởi tạo 2-3

DEVMASTER.VN

Ví dụ

```
class Employee
{
    //khai báo trường private
    private string id;
    private string fullName;
    private int salary;


    //contructor không tham số
    public Employee()
    {
        id = "";
        fullName = "";
        salary = 0;
    }
    //constructor có tham số
    public Employee(string id, string fullName, int salary)
    {
        this.id = id;
        this.fullName = fullName;
        this.salary = salary;
    }
}
```

Phương thức khởi tạo 2-3

- **Gọi constructor**

Ví dụ

```
static void Main(string[] args)
{
    Employee emp1 = new Employee();
    Employee emp2 = new Employee("E001", "Nguyen Trong Tung", 10000);
}
```

Từ khóa “this”

- Từ khóa this được sử dụng để đại diện cho đối tượng hiện tại, nó dùng để truy cập vào các thành viên trùng tên trong cùng phạm vi (xem ví dụ trang trước)
- Từ khóa this không thể truy cập vào các biến tĩnh và phương thức tĩnh

Phương thức khởi tạo tĩnh

- Phương thức khởi tạo tĩnh được sử dụng để khởi tạo các biến tĩnh hoặc gọi các phương thức tĩnh.
- Trong một lớp chỉ có duy nhất một constructor tĩnh. Constructor tĩnh không có bất kỳ tham số nào
  - ✓ Không có phạm vi truy cập, nó được gọi trực tiếp bởi CLR thay vì đối tượng,
  - ✓ Không thể truy cập tới các thành viên không tĩnh của lớp

Ví dụ

```
class Table
{
    string name;
    static int leg;
    static Table()
    {
        leg = 5;
    }
}
```

Phương thức hủy

- Phương thức hủy(Destructor) là phương thức đặc biệt có cùng tên với tên lớp nhưng khi tạo phương thức các bạn thêm ký tự "~" vào trước tên phương thức.
- Phương thức destructor sẽ gọi tự động để giải phóng bộ nhớ khi đối tượng khi đối tượng đó không sử dụng nữa.
- Một số đặc điểm của destructor
  - Destructor không thể ghi đè hoặc kế thừa.
  - Destructor không gọi tường minh
  - Destructor không có phạm vi truy cập và không có tham số

Phương thức hủy

Ví dụ

```
class Employee
{
    //khai báo trường private
    private string id;
    private string fullName;
    private int salary;
    ~Employee()
    {
        //các lệnh dọn dẹp rác
        Console.WriteLine("Doi tuong bi huy");
    }
}
```

Hỏi Đáp