CHƯƠNG 3

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG VỚI C#

Nội dung

1. Lập trình hướng đối tượng
2. Lập trình hướng đối tượng với C#.NET

1. Lập trình hướng đối tượng

1.1. Giới thiệu
1.2. Các khái niệm trong lập trình hướng đối tượng
1.3. Các tính chất của lập trình hướng đối tượng

1.1. Giới thiệu

Lập trình có cấu trúc/ thủ tục

- Tổ chức chương trình thành các chương trình con (hàm | thủ tục)
- Mỗi chương trình con đảm nhận xử lý một công việc nhỏ trong toàn bộ hệ thống
- Mỗi chương trình con này lại có thể chia thành các chương trình con nhỏ hơn

1.1. Giới thiệu

Lập trình có cấu trúc/ thủ tục

- ***Ưu điểm:***
  - Chương trình dễ hiểu, dễ bảo trì
  - Dễ dàng tạo ra các thư viện phần mềm
- ***Nhược điểm***
  - Dữ liệu và mã xử lý tách rời
  - Người lập trình phải biết cấu trúc dữ liệu

1.1. Giới thiệu

Lập trình có cấu trúc/ thủ tục

- ***Nhược điểm . . .***
  - Khi thay đổi cấu trúc dữ liệu thì thuật toán phải thay đổi theo
  - Không khởi tạo hay giải phóng dữ liệu tự động
  - Không mô tả đầy đủ, trung thực hệ thống trong thực tế

1.1. Giới thiệu

Lập trình OOP

- OOP hoàn toàn dựa vào mô hình thế giới thực
- Xây dựng chương trình xung quanh các đối tượng được định nghĩa
- Ứng dụng chỉ bao gồm 3 thành phần: class (lớp), object (đối tượng), method (phương thức)

7

1.2. Các khái niệm trong lập trình hướng đối tượng

Class (lớp)

- Class dùng để phân loại các đối tượng
- Là bản thiết kế của đối tượng, gồm các thành phần:
  - Thuộc tính của đối tượng
  - Phương thức của đối tượng

8

1.2. Các khái niệm trong lập trình hướng đối tượng

Class – ví dụ: lớp car

- Phân biệt với các phương tiện giao thông khác
- Thuộc tính (đặc điểm) của đối tượng
  - Số bánh xe
  - Số chỗ ngồi
  - Dung tích xy lanh . . .
- Phương thức (hành động) của đối tượng
  - Khởi động
  - Tăng tốc …

9

1.2. Các khái niệm trong lập trình hướng đối tượng

Object (đối tượng)

- Đối tượng biểu diễn một thực thể trong thế giới thực
- Các đối tượng đều chứa:
  - Dữ liệu (giá trị các thuộc tính)
  - Một tập hợp cách hành vi quy định (phương thức)
- Có định danh duy nhất
- Là một "hộp đen" nhận thông điệp

10

1.2. Các khái niệm trong lập trình hướng đối tượng

Object – ví dụ

My car

- Số bánh xe: 4
- Dung tích: 2.7L
- Biển số xe: 603311

Your car

- Số bánh xe: 4
- Dung tích: 2.7L
- Biển số xe: 602145

11

1.2. Các khái niệm trong lập trình hướng đối tượng

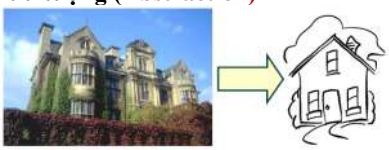
Thông điệp và Phương thức

- Các object tương tác (giao tiếp) với nhau qua thông điệp (message)
- Các hành vi (phương thức) là hành động hoặc sự phản ứng của đối tượng khi trạng thái của nó bị thay đổi và (hoặc) có thông điệp được gửi tới

12

1.3. Các tính chất của lập trình hướng đối tượng

Tính trừu tượng (Abstraction)



Quá trình chuyển từ phức tạp về đơn giản được coi là sự trừu tượng hóa

→ chỉ quan tâm tới những thông tin quan trọng, cần thiết để giải quyết bài toán

1.3. Các tính chất của lập trình hướng đối tượng

Tính trừu tượng – ví dụ

- Đối tượng: Sport Car
  - Dữ liệu:
    - Số bánh xe: 4 bánh
    - Màu xe: cam
    - Số cửa: 2 cửa
    - Có cửa trên: Yes
    - Số chỗ: 2 chỗ ngồi
    - Dung tích cylinder :2 .1L
  - Hành vi:
    - Khởi động
    - Rẽ trái, rẽ phải

Tăng tốc, giảm tốc
Dừng xe

1.3. Các tính chất của lập trình hướng đối tượng

Tính trừu tượng – ví du

- Car được mô tả bởi:
  - Số bánh xe
  - Màu xe
  - Số cửa
  - Có cửa trên hay không
  - Số chỗ ngồi
  - Khởi động
  - Tăng | giảm tốc
  - Rẽ trái | phải
  - Dừng xe

Mỗi chiếc xe có dữ liệu của chính nó và các hành động

1.3. Các tính chất của lập trình hướng đối tượng

Tính đóng gói (Encapsulation)

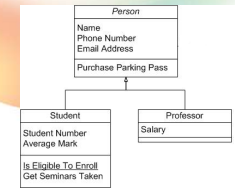
- Chỉ cho thấy các phương thức quan trọng của đối tượng
- Ẩn đi các thông tin chi tiết
  - Che giấu dữ liệu
  - Che dấu sự triển khai
  - Chỉ truy cập dữ liệu thông qua các phương thức được cài đặt

1.3. Các tính chất của lập trình hướng đối tượng

Tính kế thừa (inheritance)

Là khả năng tạo ra các lớp mới dựa trên các lớp đã có

- Base class – Super class: class cung cấp sự thực thi
- Derived class – Subclass: class kế thừa sự thực thi



1.3. Các tính chất của lập trình hướng đối tượng

Tính đa hình (polymorphism)

- Khả năng của các đối tượng có các thi hành khác nhau từ cùng một giao diện
- Khả năng của các đối tượng khác nhau hồi đáp cùng một thông điệp.
- Đa hình được triển khai bằng cách:
  - Nạp chồng phương thức & toán tử,
  - Ghi đè dữ liệu và phương thức

2. Lập trình hướng đối tượng với C#.NET

2.1. Lớp
2.2. Phương thức
2.3. Thuộc tính
2.4. Sử dụng các thành viên tĩnh
2.5. Thừa kế

19

2.1. Lớp

Định nghĩa lớp

```
[Thuộc tính] [Bổ sung truy cập] class <Định danh lớp>
                                        [: Lớp cơ sở]
{
    // Phần thân của lớp: bao gồm định nghĩa các
    // thuộc tính và phương thức hành động
}
```

20

2.1. Lớp

Định nghĩa lớp . . .

Bổ sung truy cập

Quyết định khả năng các phương thức của lớp bao gồm việc các phương thức của lớp khác có thể nhìn thấy và sử dụng các biến thành viên hay những phương thức bên trong lớp.

21

2.1. Lớp

Bổ sung truy cập

Thuộc tính	Giới hạn truy cập
public	Không hạn chế, được dùng bởi bất kỳ các phương thức của lớp, bao gồm những lớp khác
private	Chỉ được truy cập bởi các phương thức trong cùng lớp
protected	Chỉ được các phương thức trong cùng lớp và các phương thức của lớp dẫn xuất truy cập

22

2.1. Lớp

Bổ sung truy cập

Thuộc tính	Giới hạn truy cập
internal	Được truy cập bởi những phương thức của bất cứ lớp nào trong cùng assembly
protected internal	Được truy cập bởi các phương thức trong cùng lớp; các phương thức của lớp dẫn xuất và bất cứ lớp nào trong cùng assembly

23

2.2. Thuộc tính

Biến thành viên:

- Các biến thành viên được khai báo trong class để lưu giá trị của các thuộc tính
- Sử dụng *Bổ sung truy cập* khi khai báo biến

24

2.2. Thuộc tính

Thuộc tính:

- Là cách truy cập an toàn các biến thành viên trong class

```
private kiểu_dữ_liệu   tên_biến;
public kiểu_dữ_liệu   tên_thuộc_tính
{
    get
    {
        return tên_biến;
    }
    set
    {
        tên_biến = value;
    }
}
```

- để trả về một đối tượng có kiểu dữ liệu của thuộc tính
- thiết lập một giá trị mới cho thuộc tính
- từ khóa value đại diện cho tham số được truyền vào và được lưu trữ bởi thuộc tính

25

2.3. Phương thức

- Định nghĩa hành vi của đối tượng bằng phương thức
- Sử dụng *Bổ sung truy cập* khi định nghĩa phương thức
- **Khai báo đối tượng của lớp**

  tên_lớp tên_đối_tượng;
- **Khởi tạo đối tượng**

  tên_đối_tượng = new tên_lớp();

26

2.3. Phương thức

Nạp chồng phương thức

- Viết các phương thức cùng tên nhưng khác signature (ký hiệu)

Constructor (bộ khởi dựng) và Destructor (bộ hủy)

- ***Constructor***: là phương thức tự động thực thi khi một đối tượng được khởi tạo
- ***Destructor***: là phương thức tự động thực khi một đối tượng bị hủy
- Được định nghĩa khi xây dựng lớp, nếu không thì CLR sẽ tạo một cách mặc định.

27

2.3. Phương thức

Khai báo constructor:

```
public  tên_lớp([danh_sách_tham_số)
{
   // [các câu lệnh gán giá trị cho các thuộc tính]
}
```

28

2.4. Sử dụng các thành viên tĩnh

- Thuộc tính và phương thức trong một lớp có thể là **thành viên thể hiện** (instance members) hay **thành viên tĩnh** (static members).
- Thành viên thể hiện liên quan đến mỗi thể hiện của lớp
- Thành viên tĩnh được xem như một phần của lớp.

29

2.4. Sử dụng các thành viên tĩnh

- Khai báo thành viên tĩnh sử dụng từ khóa **static**
- Truy cập đến thành viên tĩnh của một lớp thông qua tên lớp

30

2.4. Sử dụng các thành viên tĩnh

Từ khóa this

- Dùng để tham chiếu đến thể hiện hiện hành của một đối tượng.
- Tham chiếu this này được xem là con trỏ ẩn đến tất các phương thức không có thuộc tính tĩnh trong một lớp.
- Mỗi phương thức có thể tham chiếu đến những phương thức khác và các biến thành viên thông qua tham chiếu this này.

31

2.5. Kế thừa

Kế thừa thuộc tính và phương thức

- Lớp con có thể tham chiếu đến tất cả các thành viên public và protected của lớp cha
- Nếu lớp con có thực thi khác với phương thức của lớp cha thì phải nạp đè (override) phương thức

32

2.5. Kế thừa

Nạp đè phương thức

- Khai báo phương thức đầu tiên trong lớp cha với từ khóa **virtual** hoặc **abstract**
- Khai báo phương thức mới trong lớp con với từ khóa **override**.

33

2.5. Kế thừa

Lớp abstract (lớp trừu tượng)

- Lớp chỉ để kế thừa
- Sử dụng từ khóa abstract khi khai báo
- Mỗi phương thức trong lớp cha có thuộc tính abstract không có code thực thi và phải được nạp đè trong lớp con

34

2.5. Kế thừa

Lớp abstract (lớp trừu tượng) . . .

```
public abstract class lớp_cha
{
    public abstract kiểu_dữ_liệu tên_phương_thức();
}

public class lớp_con:lớp_cha
{
    public override kiểu_dữ_liệu tên_phương_thức()
    {
        // code thực thi
    }
}
```

35

2.5. Kế thừa

Lớp sealed

- Lớp không được kế thừa
- Sử dụng từ khóa sealed khi khai báo

36

2.5. KẾ THỪA

Interface

- Interface là ràng buộc đảm bảo cho các lớp hay các cấu trúc phải thực hiện một điều gì đó.
- Khi thực thi một giao diện, lớp phải thực thi tất cả các phương thức của giao diện.

37

2.5. KẾ THỪA

Cú pháp:

[bổ sung truy cập] **interface** <tên giao diện>
[: danh sách giao diện cơ sở]
{
<phần thân giao diện>
}

- Giao diện chỉ chứa: phương thức, thuộc tính, sự kiện, chỉ mục.
- Định nghĩa các phương thức của giao diện không có phần bổ sung truy cập (ngầm định là public)

38

2.5. KẾ THỪA

Interface IComparable

- Phương thức Sort() của List<> chỉ hoạt động được đối với những class cài đặt interface IComparable
- Phương thức CompareTo() có 1 tham số kiểu object hoặc kiểu dữ liệu của lớp cài đặt trong trường hợp sử dụng interface IComparable<>
- Phương thức CompareTo() trả về
  - Giá trị < 0: đối tượng hiện tại nhỏ hơn đối tượng so sánh
  - Giá trị = 0: đối tượng hiện tại bằng đối tượng so sánh
  - Giá trị > 0: đối tượng hiện tại lớn hơn đối tượng so sánh

39

2.5. KẾ THỪA

Interface IEquatable

- Phương thức Equals() có 1 tham số kiểu object hoặc kiểu dữ liệu của lớp cài đặt trong trường hợp sử dụng interface IEquatable <>
- Phương thức Equals() trả về
  - true: đối tượng hiện tại bằng đối tượng so sánh
  - false: đối tượng hiện tại khác đối tượng so sánh

40