

Chương 5

Object - Oriented Programming

5.1 Một số khái niệm cơ bản
5.2 Tiếp cận lập trình hướng đối tượng
5.3 Câu hỏi

1

5.1 Một số khái niệm

- Lập trình truyền thống
- Lập trình hướng đối tượng
- So sánh

2

Lập trình truyền thống



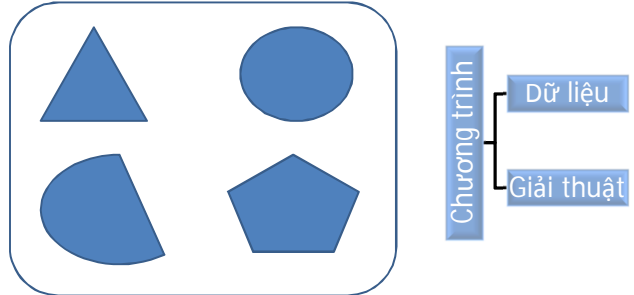
- Ngôn ngữ (tiếng, chữ viết, cử chỉ)
- Hình ảnh



- Một tập từ ngữ
- Một tập các ký hiệu

3

Chương trình



4

Ngôn ngữ lập trình

- Có rất nhiều ngôn ngữ trợ giúp lập trình:
 - Pascal
 - C++
- Đặc điểm chung:
 - Tính đơn thể
 - Cấu trúc điều khiển và tính vào/ra đơn

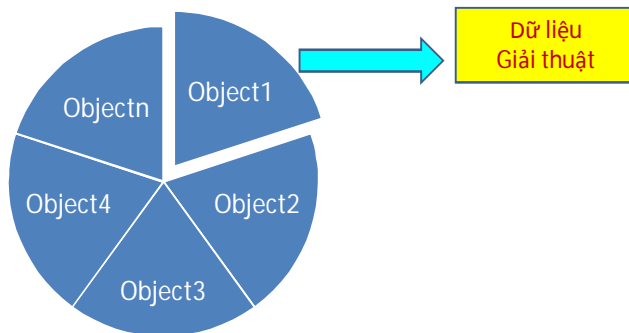
5

Ưu điểm – Nhược điểm

- Ưu điểm:
 - Dễ viết, dễ đọc, dễ hiểu, dễ kiểm lỗi và dễ hiệu chỉnh
 - Tư duy giải thuật rõ ràng
- Nhược điểm:
 - Khi thay cấu trúc dữ liệu => thay đổi chương trình
 - Không dùng lại
 - Không theo kịp sự phát triển

6

Lập trình hướng đối tượng

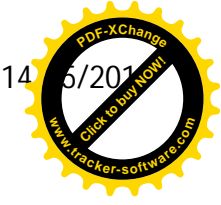


7

Ưu điểm – Nhược điểm

- Ưu điểm:
 - Không trùng lặp dữ liệu
 - Thay đổi Cấu trúc dữ liệu của một đối tượng, không cần thay đổi mã nguồn của các đối tượng khác
 - Có thể sử dụng lại mã nguồn
- Nhược điểm:
 - Mới lạ
 - Thay đổi hệ tư tưởng về lập trình
 - Thay đổi về ngôn ngữ (lập trình; hệ điều hành)
 - Đòi hỏi

8



So sánh

- Ví như xây căn nhà:
 - Móng
 - Khung
 - Trần
 - Trang trí

9

5.2. Tiếp cận lập trình hướng đối tượng

- 2.1 Class - Lớp
- 2.2 Object - Đối tượng
- 2.3 Truyền tham số
- 2.4 Một số vấn đề bổ trợ

10

2.1 Clacss

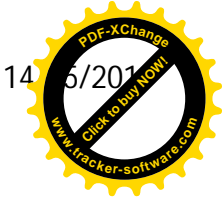
- Khái niệm
- Khai báo
- Thuộc tính truy cập
- Tham số
- Truyền tham số

11

Khái niệm

- Class là : Tập hợp các **đối tượng** có **chung các thuộc tính và hành động**
- Tất cả các thể hiện của Class được gọi là đối tượng, sẽ có chung các trạng thái và hành vi
- **Chú ý:**
 - Các thuộc tính và hành động có thể được gán chỉ sau khi một đối tượng được tạo ra
 - Khi một đối tượng được tạo ra, lúc đó ta mới có một đại diện thật sự của một thực thể

12



Ví dụ

Lớp	Đối tượng
Xe moto	Dream II Dung tích 97 cm ³ Màu nho
Xe moto	Click Dung tích 97 cm ³ Màu đen
Sinh viên	Trần Hoàng Thảo Nữ 20.07.1991 Diễn Châu
Sinh viên	Nguyễn Văn Hùng Nam 12.02.1990 TP Vinh

13

Khai báo lớp

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

14

Thuộc tính truy cập

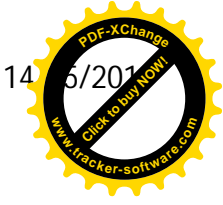
Thuộc tính	Giới hạn truy cập
public	Không hạn chế. Những thành viên được đánh dấu public có thể được dùng bởi bất kì các phương thức của lớp bao gồm những lớp khác.
private	Thành viên trong một lớp A được đánh dấu là private thì chỉ được truy cập bởi các phương thức của lớp A.
protected	Thành viên trong lớp A được đánh dấu là protected thì chỉ được các phương thức bên trong lớp A và những phương thức dẫn xuất từ lớp A truy cập.
internal	Thành viên trong lớp A được đánh dấu là internal thì được truy cập bởi những phương thức của bất cứ lớp nào trong cùng khối hợp ngữ với A.
protected internal	Thành viên trong lớp A được đánh dấu là protected internal được truy cập bởi các phương thức của lớp A, các phương thức của lớp dẫn xuất của A, và bất cứ lớp nào trong cùng khối hợp ngữ của A.

15

Định nghĩa phương thức

```
public void/int Tênphươngthức()
{
    //Câu lệnh;
}
```

16



```

using System;
public class ThoiGian
{
    public void ThoiGianHienHanh()
    {
        Console.WriteLine("Hien thi thoi gian hien hanh");
    }
    int Nam, Thang, Ngay, Gio, Phut, Giay;
}
public class Tester
{
    static void Main()
    {
        ThoiGian t = new ThoiGian();
        t.ThoiGianHienHanh();
    }
}

```

Tạo thể hiện lớp ThoiGian và gán cho đối tượng t

Hàm Main sử dụng phương thức của t

Tham số của phương thức

```

public void SomeMethod(int p1, float p2)
{
    Console.WriteLine("hai tham so: {0} va {1}", p1,p2);
}

```

18

- ## 2.2 Object
- Khái niệm
 - Khai báo
 - Khởi tạo biến thành viên
 - Hàm dựng sao chép
 - Từ khoá **this**
 - Bộ khởi dựng
 - Huỷ đối tượng
- 19

Khái niệm

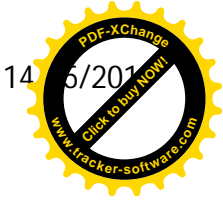






- Những vật hữu hình
- Sờ mó được
- Tồn tại trong thế giới thực
- Mô tả được:
 - Thuộc tính
 - Hành động

20



Ví dụ

	Thuộc tính	Hành động
Xe hơi	Loại Màu Động cơ	Chạy Triển lãm
Chó	Giống Giới tính Màu lông Tuổi	Ăn Sủa Chạy
Hoa	Loại Màu	Nở Toả hương
Nhà	Cấp Vị trí	Ở Cho thuê Bán Để không

21

Nhận xét

- Trong phần mềm, mô tả các đối tượng :
 - Trạng thái (thuộc tính)
 - Hành vi (hành động)
- Sự thuận lợi khi sử dụng đối tượng (object) :
 - Nó giúp chúng ta hiểu hơn về thế giới thực
 - Nó ánh xạ các thuộc tính và các hành động của các đối tượng trong thế giới thực thành trạng thái và hành vi của các đối tượng phần mềm

22

Khai báo

- Tạo một đối tượng cho lớp nào đó tương tự như việc gọi thực hiện một phương thức của lớp đó

```
ThoiGian t = new ThoiGian();
```

Tạo thể hiện lớp ThoiGian và gán cho đối tượng **t**
 Hoặc
 Tạo đối tượng **t** là thể hiện của lớp ThoiGian

23

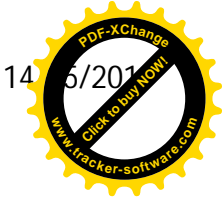
Ví dụ

```
using System;
public class ThoiGian
{
    public void ThoiGianHienHanh()
    {
        Console.WriteLine("Hien thi thoi gian hien hanh");
    }
    int Nam, Thang, Ngay, Gio, Phut, Giay;
}
public class Tester
{
    static void Main()
    {
        ThoiGian t = new ThoiGian();
        t.ThoiGianHienHanh();
    }
}
```

Tạo thể hiện lớp ThoiGian và gán cho đối tượng **t**

Hàm Main sử dụng phương thức của t

24



Khởi tạo biến thành viên

- Khởi tạo các biến thành viên:
 - Trực tiếp khi khai báo
 - Khởi tạo trong bộ khởi dựng
- Để thực hiện việc khởi tạo ta chỉ việc sử dụng phép gán giá trị cho một biến:

```
private int Giay = 30;
```

25

Hàm dựng sao chép

- Là thực hiện việc tạo một đối tượng mới bằng cách sao chép tất cả các biến từ một đối tượng đã có và cùng một kiểu dữ liệu
- Ví dụ :
 - Chúng ta muốn đưa một đối tượng ThoiGian vào bộ khởi dựng lớp ThoiGian để tạo một đối tượng ThoiGian mới có cùng giá trị với đối tượng ThoiGian cũ
 - Hai đối tượng này hoàn toàn khác nhau và chỉ giống nhau ở giá trị biến thành viên sao khi khởi dựng.

26

Ví dụ

```
public ThoiGian( ThoiGian tg)
{
    Nam = tg.Nam;
    Thang = tg.Thang;
    Ngay = tg.Ngay;
    Gio = tg.Gio;
    Phut = tg.Phut;
    Giay = tg.Giay;
}
```

```
ThoiGian t2 = new ThoiGian( t1 );
```

Trong đó t1 là đối tượng ThoiGian đã tồn tại, sau khi lệnh trên thực hiện xong thì đối tượng t2 được tạo ra như bản sao của đối tượng t1

27

Từ khóa this

- Từ khóa this :
 - Được dùng để tham chiếu đến thể hiện hiện hành của một đối tượng
 - Được dùng tham chiếu đến những phương thức khác và các biến thành viên
- Tham chiếu this này được sử dụng thường xuyên theo ba cách:
 - Sử dụng khi các biến thành viên bị che lấp bởi tham số đưa vào
 - Để truyền đối tượng hiện hành vào một tham số của một phương thức của đối tượng khác
 - Sử dụng tham chiếu this là mảng chỉ mục (indexer)

28

Code

```
public void SetYear( int Nam)
{
    this.Nam = Nam;
}

public void Method1( OtherClass otherObject )
{
    // truyền tham số là bản
    // thân đối tượng đang thực hiện.
    otherObject.SetObject( this );
}
```

29

Bộ khởi dựng

- Khái niệm – Chức năng
- Tạo bộ khởi dựng
- Gán giá trị
- Sử dụng các thành viên tĩnh
- Sử dụng bộ khởi tạo tĩnh
- Sử dụng bộ khởi dựng private
- Sử dụng các thuộc tính tĩnh

30

Khái niệm – Chức năng

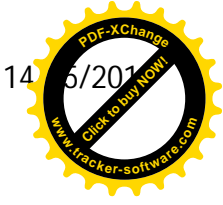
- Một phương thức sẽ được gọi thực hiện khi chúng ta tạo một đối tượng. Phương thức này được gọi là **bộ khởi dựng** (constructor):
 - Được định nghĩa khi xây dựng lớp
 - Nếu ta không tạo ra thì CLR sẽ tạo phương thức khởi dựng một cách mặc định
- Chức năng của bộ khởi dựng:
 - Là tạo ra đối tượng được xác định bởi một lớp và đặt trạng thái này là hợp lệ
 - Sau khi bộ khởi dựng thực hiện hoàn thành thì bộ nhớ sẽ lưu giữ một thể hiện hợp lệ của lớp vừa khai báo

31

Chưa có bộ khởi dựng

```
public class ThoiGian
{
    public void ThoiGianHienHanh()
    {
        Console.WriteLine("Hien thi thoi gian hien hanh");
    }
    int Nam, Thang, Ngay, Gio, Phut, Giay;
}
```

32



Tạo bộ khởi dựng

```

using System;
public class ThoiGian
{
    public void ThoiGianHienHanh()
    {
        Console.WriteLine("Thoi gian hien hanh la : {0}/{1}/{2} {3}:{4}:{5}", Ngay, Thang, Nam, Gio, Phut, Giay);
    }
    public ThoiGian( System.DateTime dt )
    {
        Nam = dt.Year; Thang = dt.Month; Ngay = dt.Day;
        Gio = dt.Hour; Phut = dt.Minute; Giay = dt.Second;
    }
    int Nam, Thang, Ngay, Gio, Phut, Giay;
    public class Tester
    {
        static void Main()
        {
            System.DateTime currentTime = System.DateTime.Now;
            ThoiGian t = new ThoiGian( currentTime );
            t.ThoiGianHienHanh();
        }
    }
}

```

Định nghĩa bộ khởi dựng riêng
 Nghĩa là
 Định nghĩa một phương thức có tên giống tên lớp đã khai báo

Phương thức khởi dựng lấy một đối tượng DateTime và khởi tạo tất cả các biến thành viên dựa trên giá trị của đối tượng này. Khi phương thức này thực hiện xong, một đối tượng ThoiGian được tạo ra và các biến của đối tượng cũng đã được khởi tạo. Hàm ThoiGianHienHanh được gọi trong hàm Main() sẽ hiển thị giá trị thời gian lúc đối tượng được tạo ra

Giá trị mặc định

- Các biến trong bộ khởi dựng mặc định sẽ nhận các giá trị mặc định như sau

Kiểu dữ liệu	Giá trị mặc định
int, long, byte,...	0
bool	false
char	'\0' (null)
enum	0
reference	null

- Người sử dụng có thể gán giá trị khác cho các biến này

Sử dụng các thành viên tĩnh (static member)

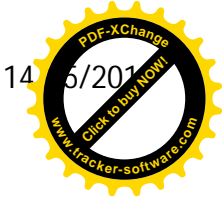
- Những thuộc tính và phương thức trong một lớp có thể là những thành viên :
 - Thể hiện (instance members) : thành viên của đối tượng liên quan đến thể hiện của một kiểu dữ liệu
 - Tĩnh (static members) : được xem như một phần của lớp
- Chúng ta có thể truy cập đến thành viên tĩnh của một lớp thông qua tên lớp đã được khai báo mà không cần tạo một thể hiện lớp

```

using System;
public class Class1
{
    public void SomeMethod(int p1, float p2)
    {
        Console.WriteLine("Ham nhan duoc hai tham so: {0} va {1}", p1,p2);
    }
}
public class Tester
{
    static void Main()
    {
        int var1 = 5;
        float var2 = 10.5f;
        Class1 c = new Class1();
        c.SomeMethod( var1, var2 );
    }
}

```

- SomeMethod là phương thức không tĩnh của lớp Class1
- Main() là một phương thức tĩnh
- Phương thức tĩnh không thể truy cập trực tiếp đến các thành viên không có tính chất tĩnh (nonstatic)
- Giải quyết :**
- Để truy cập được phương thức này, ta phải tạo một đối tượng c là thể hiện của lớp Class1



Sử dụng bộ khởi dựng tĩnh

```

6 namespace handungtinh
7 {
8     public class ThoiGian
9     {
10        public void ThoiGianHienHanh()
11        {
12            System.Console.WriteLine(" Ten: {0}", ten);
13            Console.WriteLine("Thoi gian hien hanh la : {0}/{1}/{2} {3}:{4}:{5}", Ngay, Thang, Nam, Gio, Phut, Giay);
14        }
15        public ThoiGian( System.DateTime dt )
16        {
17            Nam = dt.Year; Thang = dt.Month; Ngay = dt.Day;
18            Gio = dt.Hour; Phut = dt.Minute; Giay = dt.Second; }
19        static ThoiGian()
20        {
21            ten = "Thoi gian"; }
22        int Nam, Thang, Ngay, Gio, Phut, Giay;
23        private static string ten;
24    }
25    public class Tester
26    {
27        static void Main(string[] args)
28        {
29            System.DateTime currentTime = System.DateTime.Now;
30            ThoiGian t = new ThoiGian( currentTime );
31            t.ThoiGianHienHanh();
32            System.Console.ReadLine();
33        }
34    }
35 }

```

3. Dòng lệnh sử dụng biến thành viên **ten** trong phương thức không tĩnh

1. Tạo bộ khởi dựng tĩnh

2. Khai báo biến thành viên **ten** dạng tĩnh

37

Sử dụng bộ khởi dựng private

- Ngôn ngữ C# không có phương thức toàn cục và hằng số toàn cục
- Vậy :
 - Ta có thể tạo ra những lớp tiện ích nhỏ chỉ để chứa các phương thức tĩnh
 - Khi tạo một lớp mà không cho phép tạo bất kỳ thể hiện nào của lớp thì ta sử dụng bộ khởi dựng private

38

Sử dụng các thuộc tính tĩnh

- Ta không thể tạo được biến toàn cục để làm công việc đếm số thể hiện của một lớp. Vậy làm sao kiểm soát được số thể hiện của một lớp được tạo ra khi thực hiện chương trình
- Thông thường các biến thành viên tĩnh được dùng để đếm số thể hiện đã được tạo ra của một lớp

39

```

6 namespace demsothien
7 {
8     public class Cat
9     {
10        public Cat()
11        {
12            instance++; }
13        public static void HowManyCats()
14        { Console.WriteLine("{0} cats", instance); }
15        private static int instance = 0;
16    }
17    public class Tester
18    {
19        static void Main()
20        {
21            Cat.HowManyCats();
22            Cat cat1 = new Cat();
23            Cat.HowManyCats();
24            Cat cat2 = new Cat();
25            Cat cat3 = new Cat();
26            Cat.HowManyCats();
27            System.Console.ReadLine();
28        }
29    }
30 }

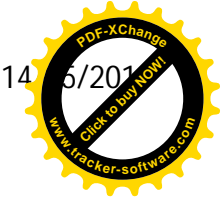
```

Bộ khởi dựng tĩnh

Biến thành viên tĩnh

0 cats
1 cats
3 cats

40



Hủy đối tượng

- Bộ hủy của C#
- Phương thức Dispose
- Phương thức Close
- Câu lệnh using

41

Bộ hủy của C#

```
-Class1()  
{  
    // Thực hiện một số công việc  
}
```

Cũng tương tự như viết :

```
Class1.Finalize()  
{  
    // Thực hiện một số công việc  
    base.Finalize();  
}
```

42

Phương thức Dispose

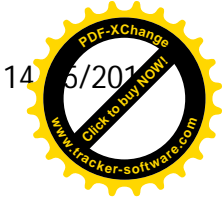
```
public void Dispose()  
{  
    // Thực hiện công việc dọn dẹp  
    // Yêu cầu bộ thu dọn GC trong thực hiện kết thúc  
    GC.SuppressFinalize( this );  
}  
public override void Finalize()  
{  
    Dispose();  
    base.Finalize();  
}
```

43

Phương thức Close

- Phương thức Close() dễ sử dụng hơn phương thức Dispose trong các đối tượng có liên quan đến xử lý tập tin
- Ta có thể xây dựng :
 - Phương thức Dispose() với thuộc tính là private
 - Phương thức Close() với thuộc tính public; Trong Close() gọi thực hiện phương thức Dispose()

44



Câu lệnh using

```
using System.Drawing;
class Tester
{
    public static void Main()
    {
        using ( Font AFont = new Font("Arial",10.0f))
        {
            // Đoạn mã sử dụng AFont
            .....
        } // Trình biên dịch sẽ gọi Dispose để giải phóng AFont
        Font TFont = new Font("Tahoma",12.0f);
        using (TFont)
        {
            // Đoạn mã sử dụng TFont
            .....
        } // Trình biên dịch gọi Dispose để giải phóng TFont
    }
}
```

45

2.3 Truyền tham số cho phương thức

- Truyền tham trị
- Truyền tham chiếu

46

Truyền tham số giá trị

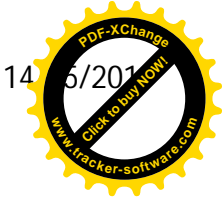
```
public void SomeMethod(int i, float f)
{
    Console.WriteLine("hai tham so: {0} va {1}", i, f);
}
public class Tester
{
    static void Main()
    {
        int var1 = 5; float var2 = 10.5;
        Class1 c = new Class1();
        c.SomeMethod(5, 10.5);
    }
}
```

47

Truyền tham chiếu

- Bổ sung tham số **ref** cho phép truyền các đối tượng giá trị vào trong phương thức theo kiểu tham chiếu
- Bổ sung **out** trong trường hợp muốn truyền dưới dạng tham chiếu mà không cần phải khởi tạo giá trị ban đầu cho tham số truyền
- Bổ sung **params** cho phép phương thức chấp nhận nhiều số lượng các tham số

48



```

6 namespace truyenhamso
7 {
8     public class Time
9     {
10         public void DisplayCurrentTime()
11         { Console.WriteLine("{0}/{1}/{2}/ {3}:{4}:{5}", Date, Month, Year, Hour, Minute, Second); }
12         public int GetHour()
13         { return Hour; }
14         public void GetTime(i 0, i 0, i 0)
15         { h = Hour; m = Minute; s = Second; }
16         public Time( System.DateTime dt)
17         { Year = dt.Year; Month = dt.Month; Date = dt.Day;
18           Hour = dt.Hour; Minute = dt.Minute; Second = dt.Second; }
19         private int Year, Month, Date, Hour, Minute, Second;
20     }
21     public class Tester
22     {
23         static void Main(string[] args)
24         { System.DateTime currentTime = System.DateTime.Now;
25           Time t = new Time( currentTime);
26           t.DisplayCurrentTime();
27           int theHour = 0, theMinute = 0, theSecond = 0;
28           t.GetTime(t.Hour, t.Minute, t.Second);
29           System.Console.WriteLine("Current time: {0}:{1}:{2}", theHour, theMinute, theSecond);
30           System.Console.ReadLine(); }
31     }
32 }

```

25/4/2011/ 20:29:17
Current time: 0:0:0

```

6 namespace truyenhamso
7 {
8     public class Time
9     {
10         public void DisplayCurrentTime()
11         { Console.WriteLine("{0}/{1}/{2}/ {3}:{4}:{5}", Date, Month, Year, Hour, Minute, Second); }
12         public int GetHour()
13         { return Hour; }
14         public void GetTime(ref int h, ref int m, ref int s)
15         { h = Hour; m = Minute; s = Second; }
16         public Time( System.DateTime dt)
17         { Year = dt.Year; Month = dt.Month; Date = dt.Day;
18           Hour = dt.Hour; Minute = dt.Minute; Second = dt.Second; }
19         private int Year, Month, Date, Hour, Minute, Second;
20     }
21     public class Tester
22     {
23         static void Main(string[] args)
24         { System.DateTime currentTime = System.DateTime.Now;
25           Time t = new Time( currentTime);
26           t.DisplayCurrentTime();
27           int theHour = 0, theMinute = 0, theSecond = 0;
28           t.GetTime(ref theHour, ref theMinute, ref theSecond);
29           System.Console.WriteLine("Current time: {0}:{1}:{2}", theHour, theMinute, theSecond);
30           System.Console.ReadLine(); }
31     }
32 }

```

25/4/2011/ 20:31:50
Current time: 20:31:50

```

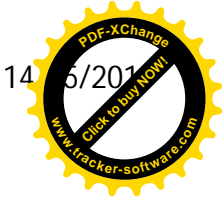
6 namespace truyenhamso
7 {
8     public class Time
9     {
10         public void DisplayCurrentTime()
11         { Console.WriteLine("{0}/{1}/{2}/ {3}:{4}:{5}", Date, Month, Year, Hour, Minute, Second); }
12         public int GetHour()
13         { return Hour; }
14         public void GetTime(out int h, out int m, out int s)
15         { h = Hour; m = Minute; s = Second; }
16         public Time( System.DateTime dt)
17         { Year = dt.Year; Month = dt.Month; Date = dt.Day;
18           Hour = dt.Hour; Minute = dt.Minute; Second = dt.Second; }
19         private int Year, Month, Date, Hour, Minute, Second;
20     }
21     public class Tester
22     {
23         static void Main(string[] args)
24         { System.DateTime currentTime = System.DateTime.Now;
25           Time t = new Time( currentTime);
26           t.DisplayCurrentTime();
27           int theHour, theMinute, theSecond;
28           t.GetTime(out theHour, out theMinute, out theSecond);
29           System.Console.WriteLine("Current time: {0}:{1}:{2}", theHour, theMinute, theSecond);
30           System.Console.ReadLine(); }
31     }
32 }

```

Truyền tham chiếu với biến chưa khởi tạo

2.4 Một số vấn đề hỗ trợ

- Encapsulation - Tính đóng gói
- Abstraction - Trừu tượng hóa
- Inheritance - Kế thừa
- Polymorphism - Đa hình



Encapsulation

```

6 namespace donggoi
7 {
8     public class Time
9     {
10         public void DisplayCurrentTime()
11         { Console.WriteLine("{0}/{1}/{2}/ {3}:{4}:{5}", date, month, year, hour, minute, second); }
12         public Time(System.DateTime dt)
13         { year = dt.Year; month = dt.Month; date = dt.Day;
14           hour = dt.Hour; minute = dt.Minute; second = dt.Second; }
15         public int Hour
16         { get {return hour;}
17           set { hour = value; } }
18         private int year, month, date, hour, minute, second;
19     }
20     public class Tester
21     {
22         static void Main(string[] args)
23         { System.DateTime currentTime = System.DateTime.Now;
24           Time t = new Time(currentTime);
25           t.DisplayCurrentTime();
26           int theHour = t.Hour;
27           System.Console.WriteLine("\nthe hour {0}\n", theHour);
28           theHour++;
29           System.Console.WriteLine("\nCập nhật the hour {0}\n", theHour);
30           System.Console.ReadLine(); }
31     }
32 }

```

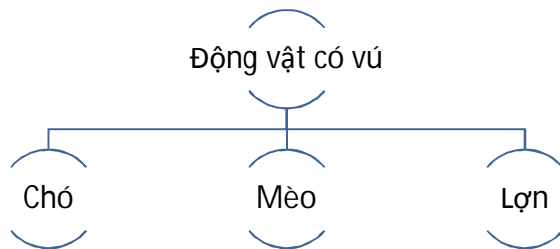
53

Encapsulation

- Phương thức get:
 - Trả về một đối tượng kiểu là một đặc tính của lớp
 - Cú pháp: `get { return <tên biến>; }`
- Phương thức set:
 - Thiết lập giá trị một property của đối tượng và có trả về là void
 - Có thể ghi vào CSDL hay cập nhật biến thành viên khi cần
 - Cú pháp: `set {<tên biến> = value;}`

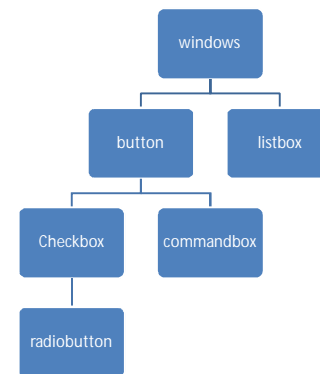
54

Abstraction



55

Abstraction



56

Abstraction

- Là lớp có ít nhất một phương thức trừu tượng
- Phương thức trừu tượng :
 - Không có sự thực thi
 - Chỉ đơn giản tạo ra một tên phương thức và kí hiệu phương thức
 - Nó không định nghĩa phần thân, thay vào đó chúng được cài đặt trong phương thức ghi đè của lớp dẫn xuất
- Khai báo lớp trừu tượng :

```
abstract class viduloptruutuong  
{  
    // Code of members  
}
```

57

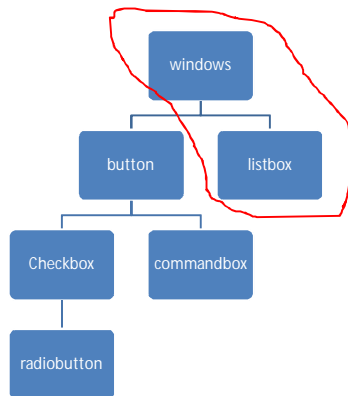
Inheritance

- Trong ngôn ngữ C#, quan hệ đặc biệt hóa được thực thi bằng cách sử dụng sự kế thừa
- Đây cách chung nhất và tự nhiên nhất để thực thi quan hệ này
- Trong ngôn ngữ C# để tạo một lớp dẫn xuất từ một lớp ta thêm dấu hai chấm (:) vào sau tên lớp dẫn xuất và trước tên lớp cơ sở

```
public class ListBox : Window
```

58

Inheritance



- Ta có thể nói ListBox **kế thừa** hay được dẫn xuất từ Window :
 - Window là lớp cơ sở
 - ListBox là lớp dẫn xuất
- Như vậy, ListBox :
 - Dẫn xuất tất cả các thuộc tính và hành vi từ lớp Window
 - Thêm những phần đặc biệt riêng để xác nhận ListBox

59

```
6 namespace kethua
7 {
8     public class Window
9     {
10        // Hàm khởi dựng lấy hai số nguyên chỉ đến vị trí của cửa sổ trên console
11        public Window(int top, int left)
12        { this.top = top; this.left = left; }
13        // mô phỏng vẽ cửa sổ
14        public void DrawWindow()
15        { Console.WriteLine("Drawing Window at {0}, {1}", top, left); }
16        // Có hai biến thành viên private => sẽ không thấy bên trong lớp dẫn xuất.
17        private int top, left;
18    }
19    public class ListBox : Window // ListBox dẫn xuất từ Window
20    {
21        public ListBox(int top, int left, string theContents)
22            : base(top, left) // gọi khởi dựng của lớp cơ sở
23        { mListBoxContents = theContents; }
24        public new void DrawWindow() // Tạo một phiên bản mới cho phương thức DrawWindow
25        { base.DrawWindow();
26          Console.WriteLine("ListBox write: {0}", mListBoxContents); }
27        private string mListBoxContents; // biến thành viên private
28    }
29    public class Tester
30    {
31        public static void Main()
32        { Window w = new Window(5, 10); // tạo đối tượng cho lớp cơ sở
33          w.DrawWindow();
34          ListBox lb = new ListBox(20, 10, "Hello world!"); // tạo đối tượng cho lớp dẫn xuất
35          lb.DrawWindow();
36          System.Console.ReadLine(); }
37    }
38 }
```

60

Polymorphism

- Có hai cách thức khá mạnh để thực hiện việc kế thừa :
 - Một là sử dụng lại mã nguồn
 - Hai là sử dụng tính đa hình (polymorphism)

61

Đa hình



62

Sử dụng

- Để tạo một phương thức ảo tính đa hình, chúng ta cần phải khai báo khóa **virtual** trong phương thức của lớp cơ sở

```
public class ListBox : Window

public virtual void DrawWindow()
```

63

Câu hỏi

- Lớp là gì ? Đối tượng là gì?
- Lớp trừu tượng là thế nào? Có thể tạo đối tượng cho lớp trừu tượng hay không?
- Có phải khi tạo một lớp thì phải kế thừa từ một lớp nào không?
- Có thể kế thừa từ một lớp cơ sở được viết trong ngôn ngữ khác ngôn ngữ C#?
- Khái niệm đa hình là gì? Khi nào thì cần sử dụng tính đa hình?
- Từ khóa new được sử dụng làm gì trong các lớp?
- Hãy xây dựng cây phân cấp các lớp đối tượng sau: Xe_Toyota, Xe_Dream, Xe_Spacy, Xe_BMW, Xe_Fiat, Xe_DuLich, Xe_May, Xe?

64