Chương 6. Kế thừa và đa hình

TRẦN VIỆT KHÁNH

Email: tranvietkhanh79@gmail.com

Cập nhật: 04 tháng 09 năm 2016

Nội dung

- Khái niệm về tính kế thừa
- Thiết kế lớp kế thừa
- Thiết lập và hủy trong kế thừa

Đặt vấn đề

Giả sử đã xây dựng lớp CDate hoàn chỉnh

→ Cần xây dựng ứng dụng tính tiền lãi của một ngân hàng thành lập ngày 14/3/1997

→ Cần xây dựng ứng dụng quản lý sinh viên có thuộc tính ngày tháng năm sinh (sinh viên phải từ 17 tuổi trở lên)

Đặt vấn đề

Cách 1: Sửa lại lớp CDate cho phù hợp với các yêu cầu của lớp CDate trong ứng dụng trên → Sửa lại hàm kiểm tra → Ảnh hưởng đến các chương trình khác có sử dụng lớp CDate ở dạng tổng quát.

Đặt vấn đề

Cách 2: Xây dựng lớp CDate mới độc lập với lớp CDate → Tốn nhiều công sức.

Cách 3: Sao chép lớp CDate để tạo lớp CDate mới và sau đó sửa lại theo yêu cầu của chương trình → Khó khăn do thực hiện thủ công khi mở rộng, cập nhật, ...

Đặt vấn đề

➔ Cần có cơ chế cho phép khai báo lớp CDate mới là lớp CDate cũ với 1 số các sửa đổi bổ sung.

Đặt vấn đề

Tương tự cho chương trình đánh caro, cờ tướng trên máy tính. Mỗi quân cờ được xem như 1 điểm ký tự (CDiemKT) nhưng mỗi quân cờ có những đặc điểm khác nhau. Do vậy cần sử dụng lớp CDiemKT bổ sung và sửa đổi một số phần chứ không phải tốn công sức để xây dựng lại từ đầu.

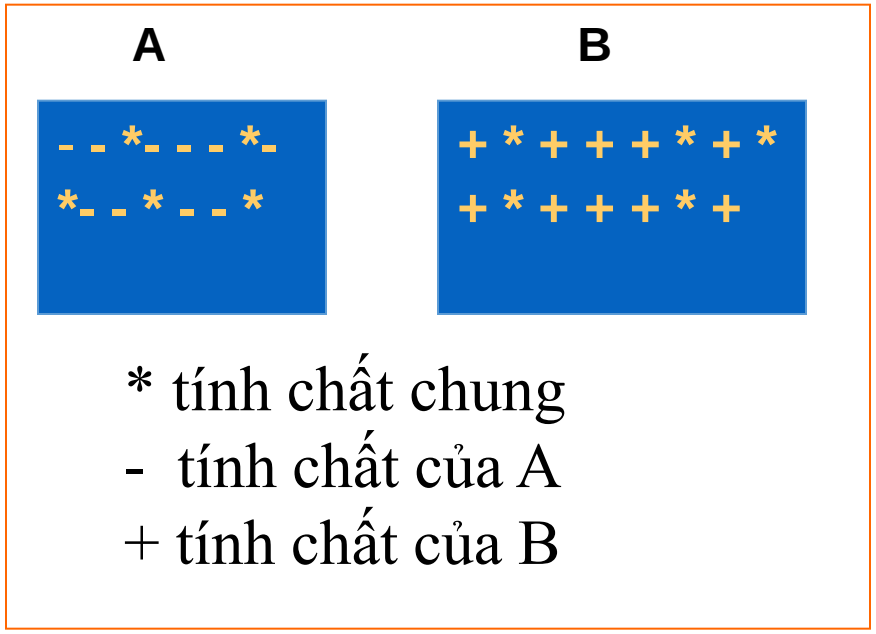
Khái niệm

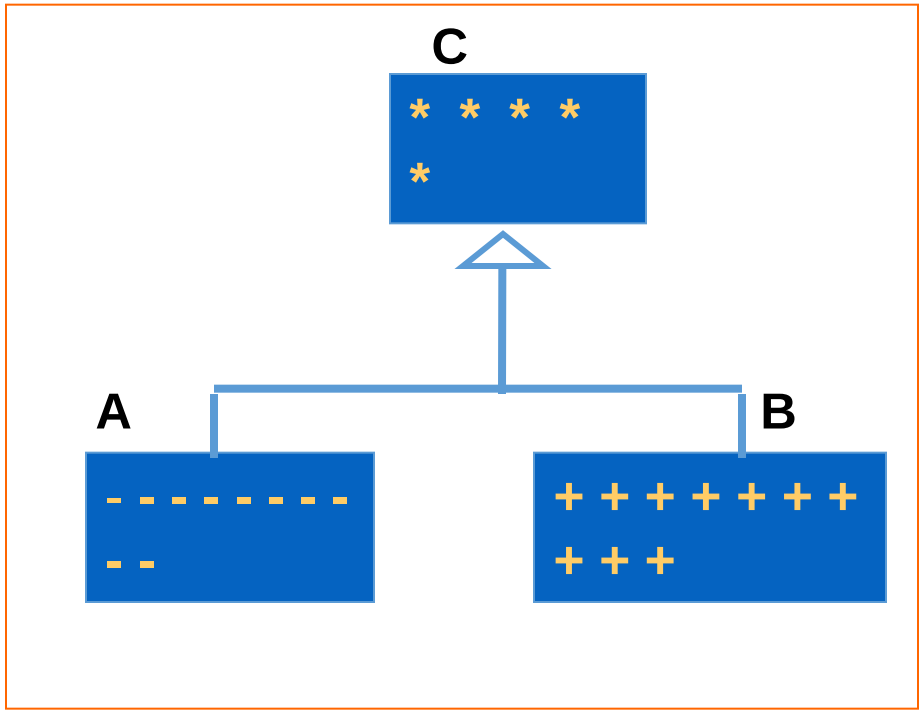
Kế thừa cho phép khai báo 1 lớp B là 1 lớp dẫn xuất từ lớp A. Khi đó B sẽ có tất cả các thuộc tính và đặc điểm của A, ngoài ra B có thể có thêm những thuộc tính và những hành động mới.

Khái niệm

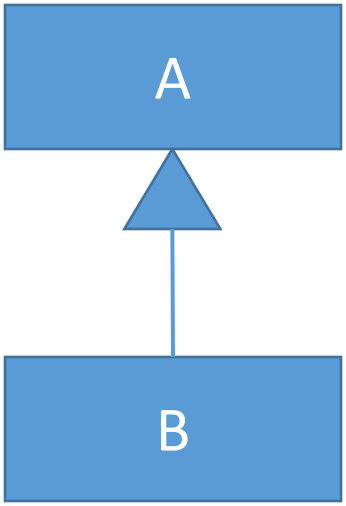
- Kế thừa thể hiện khả năng tái sử dụng các lớp đã được định nghĩa.
- Có thể định nghĩa lớp đối tượng mới dựa trên 1 hay nhiều lớp đối tượng đã có sẵn.
- Lớp có sẵn được gọi là lớp cơ sở (based class) và lớp kế thừa được gọi là lớp dẫn xuất (derived class)

Khái niệm

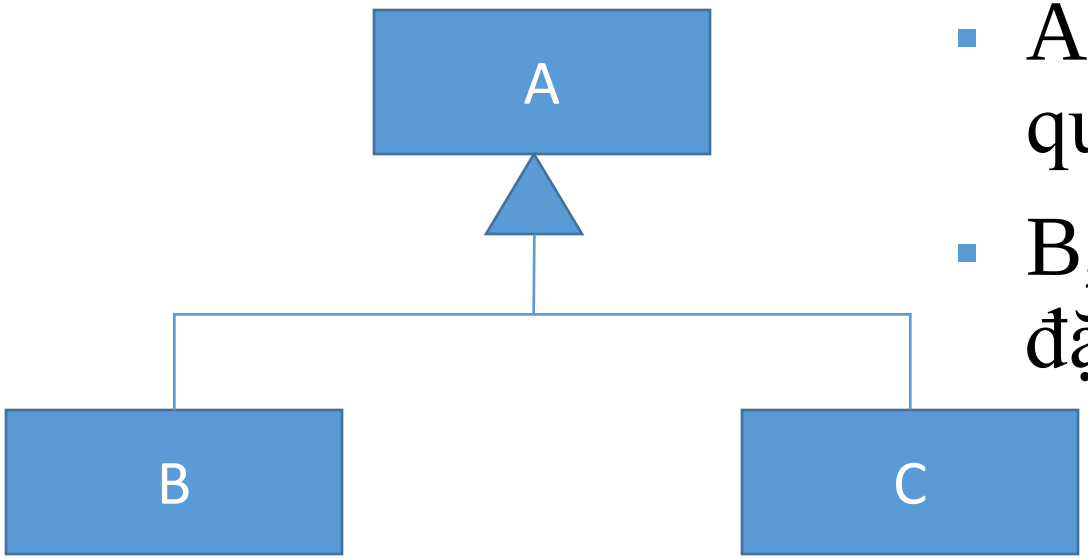




Ký hiệu

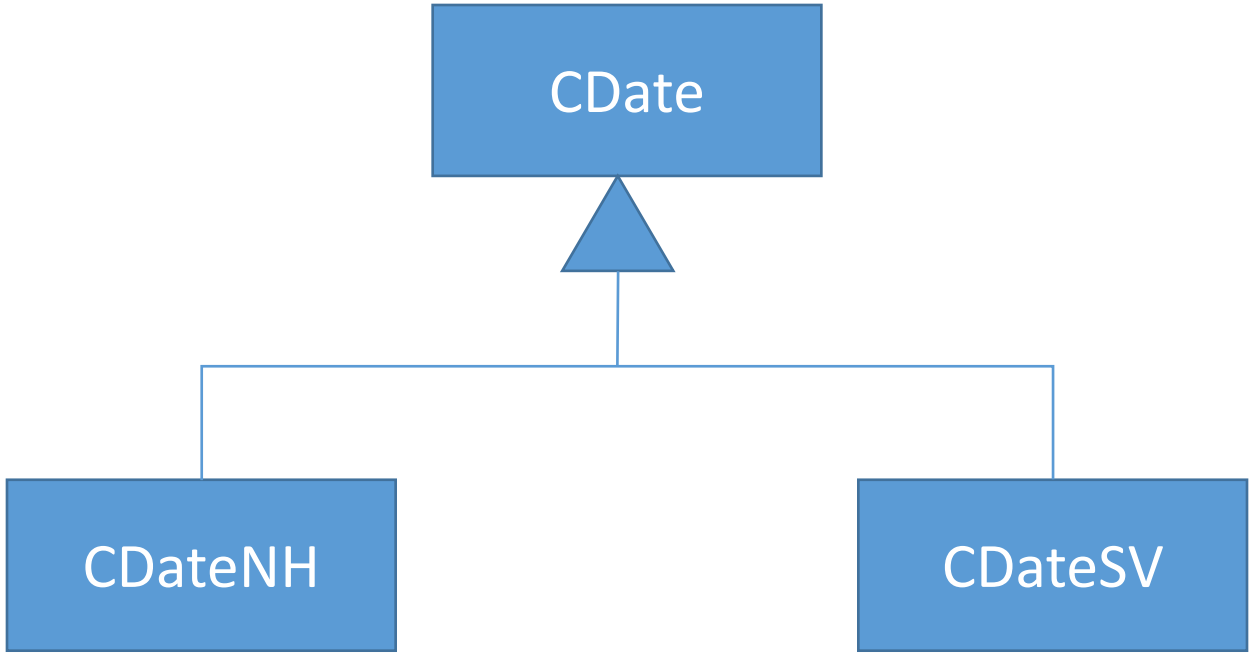


- A: Là trường hợp tổng quát của B
- B: Là trường hợp đặc biệt của A



- A: Là trường hợp tổng quát của B và C
- B, C: Là trường hợp đặc biệt của A

VD: Lớp ngày cho ngân hàng và sinh viên



Khai báo


```
class TênLớpCha
{
    Thuộc tính và phương thức của lớp cha
}
class TênLớpDẫnXuất : TênLớpCha
{
    Thuộc tính và phương thức bổ sung của
    lớp dẫn xuất
}
```

Từ khóa base

Từ khóa base thường được sử dụng để gọi phương thức khởi tạo(phương thức dựng) của lớp cha
Ví dụ:

```
Class NhanVien
{
        string strMaSo,strHoTen;
        public NhanVien()
        {
                strMaSo="";
                strHoTen="";
        }
}
```

```
Class NVBienChe:NhanVien
{
        int bacluong;
        public NVBienChe():base()
        {
                bacluong = 0;
        }
}
```

Khai báo

Có 2 cách để định nghĩa hành động bổ sung cho phương thức đã có sẵn ở lớp cha trong lớp dẫn xuất (phương thức lớp dẫn xuất trùng tên với phương thức lớp cha)

- Dùng từ khóa new
- Dùng từ khóa virtual và override

Khai báo – Dùng từ khóa new


```
class COSO
{
        protected kiểu data1;
        protected kiểu data2;
        public void Method1()
        {}
        public void Method2()
        {}
}
```

```
class DANXUAT : COSO
{
        private kiểu data3;
        public new void Method1()
        {}
        public void Method4()
        {}
}
```

Khai báo – Dùng virtual & override


```
class COSO
{
        protected kiểu data1;
        protected kiểu data2;
        public virtual void Method1()
        {}
        public virtual void Method2()
        {}
}
```

```
class DANXUAT : COSO
{
        private kiểu data3;
        public override void Method1()
        {}
        public void Method4()
        {}
}
```

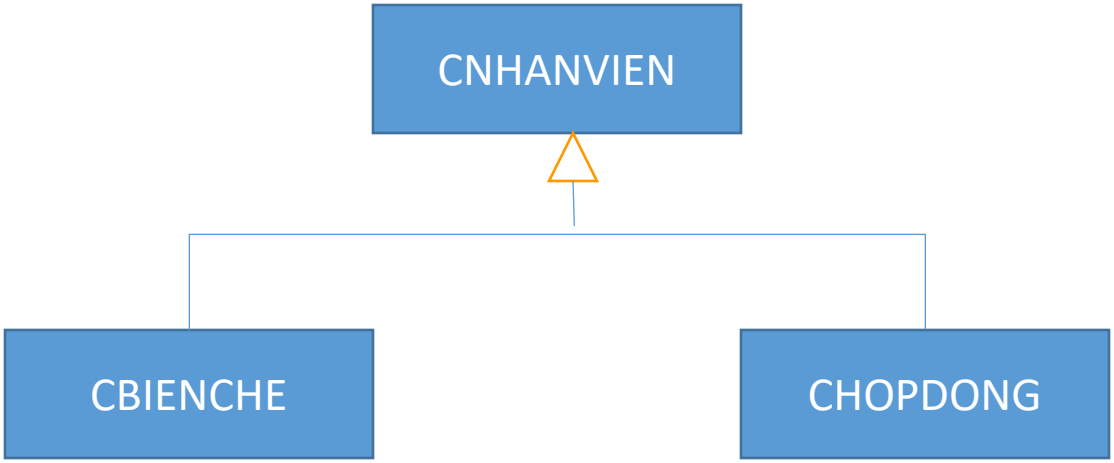
Ví dụ

Viết chương trình nhập xuất nhân viên, biết rằng gồm 2 loại nhân viên: Nhân viên biên chế và nhân viên hợp đồng. Thông tin của nhân viên gồm: Mã số, Họ tên.

- Nhân viên biên chế có thông tin riêng là bậc lương.
- Nhân viên hợp đồng có thông tin riêng là số giờ làm.

Ví dụ

Ta có cây kế thừa sau:



VD dùng từ khoá new


```
class CNHANVIEN
  {  protected int maso;
     protected string hoten;
      public void Nhap()
     {  Console.Write("Nhap ma so nhan vien: ");
        maso = int.Parse(Console.ReadLine());
        Console.Write("Nhap ho ten nhan vien: ");
        hoten = Console.ReadLine();
     }
     public void Xuat()
     {  Console.WriteLine("Ma so: {0}\nHo ten: {1}", maso, hoten);
     }
  }
```

Ví dụ – Dùng từ khoá new


```
class CBIENCHE : CNHANVIEN
  {
     private float hesoluong;
     public new void Nhap()
     {  base.Nhap();
        Console.Write("Nhap he so luong: ");
        hesoluong = float.Parse(Console.ReadLine());
     }
     public new void Xuat()
     {   base.Xuat();
        Console.WriteLine("He so luong: " + hesoluong);
     }
  }
```

Ví dụ – Dùng từ khoá new


```
class CHOPDONG : CNHANVIEN
  {  private float sogio;
     public new void Nhap()
     {  base.Nhap();
        Console.Write("Nhap so gio lam viec: ");
        sogio = float.Parse(Console.ReadLine());
     }
     public new void Xuat()
     {  base.Xuat();
        Console.WriteLine("So gio lam viec: " + sogio);
     }
  }
```

Ví dụ – Dùng virtual & override


```
class CNHANVIEN
    {   protected int maso;  protected string hoten;
      public virtual void Nhap()
       {  Console.Write("Nhap ma so nhan vien: ");
          maso = int.Parse(Console.ReadLine());
          Console.Write("Nhap ho ten nhan vien: ");
          hoten = Console.ReadLine();
       }
       public virtual void Xuat()
       {  Console.WriteLine("Ma so: {0}\nHo ten: {1}", maso, hoten);
       }
    }
```

Ví dụ – Dùng virtual & override


```
class CBIENCHE : CNHANVIEN
  {
     private float hesoluong;
     public override void Nhap()
     {  base.Nhap();
        Console.Write("Nhap he so luong: ");
        hesoluong = float.Parse(Console.ReadLine());
     }
     public override void Xuat()
     {  base.Xuat();
        Console.WriteLine("He so luong: " + hesoluong);
     }
  }
```

Ví dụ – Dùng virtual & override


```
class CHOPDONG : CNHANVIEN
  {  private float sogio;
     public override void Nhap()
     {  base.Nhap();
        Console.Write("Nhap so gio lam viec: ");
        sogio = float.Parse(Console.ReadLine());
     }
     public override void Xuat()
     {  base.Xuat();
        Console.WriteLine("So gio lam viec: " + sogio);
     }
  }
```

Ví dụ - Sử dụng phương thức trong Main()


```
static void Main(string[] args)
{  CBIENCHE nvbc = new CBIENCHE();
   nvbc.Nhap();
   CHOPDONG nvhd = new CHOPDONG();
   nvhd.Nhap();
   Console.WriteLine("\nNhan vien bien che: ");
   nvbc.Xuat();
   Console.WriteLine("\nNhan vien hop dong: ");
   nvhd.Xuat();
}
```

Phạm vi kế thừa

Có 3 phạm vi kế thừa:

- public
- protected
- private

Lưu ý: Nếu không nói rõ là phạm vi kế thừa gì, chúng ta ngầm định đó là kế thừa public

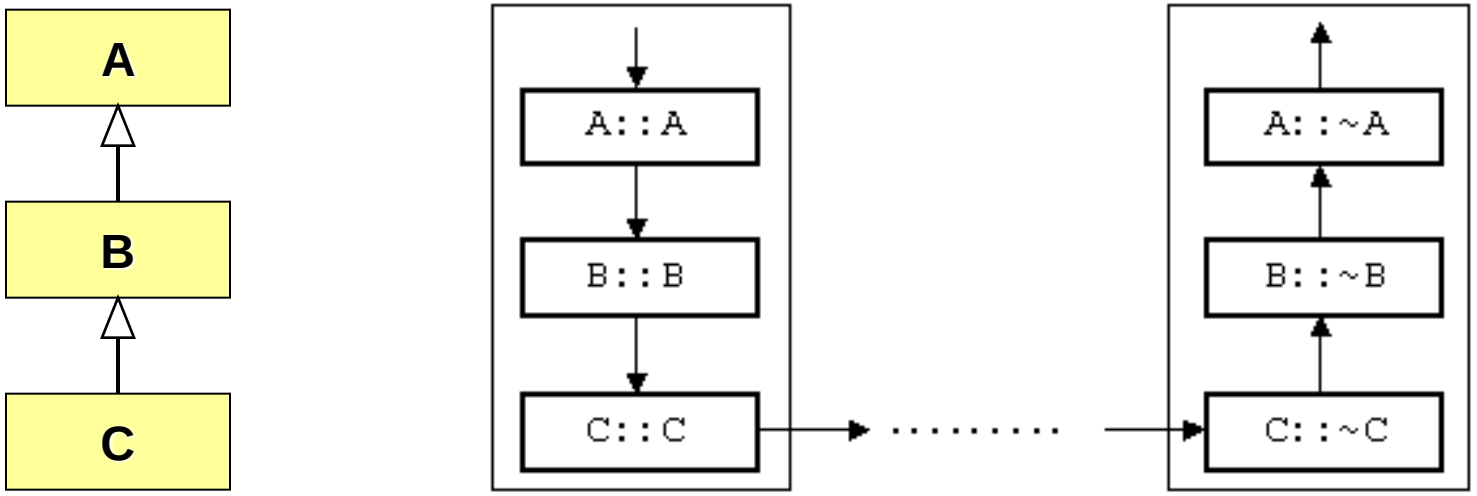
Phạm vi kế thừa

- public: thành phần public & protected của lớp cơ sở là thành phần public & protected của lớp dẫn xuất.
- protected: thành phần public & protected của lớp cơ sở là thành phần protected của lớp dẫn xuất.
- private: thành phần public & protected của lớp cơ sở là thành phần private của lớp dẫn xuất.

Phương thức thiết lập & huỷ trong kế thừa

- Khi khởi tạo đối tượng:
  - Phương thức thiết lập của **lớp cha** sẽ được gọi trước
  - Sau đó mới là phương thức thiết lập của **lớp con**.
- Khi hủy đối tượng:
  - Phương thức hủy của **lớp con** sẽ được gọi trước
  - Sau đó mới là phương thức hủy của **lớp cha**.

Phương thức thiết lập & huỷ trong kế thừa



Phương thức thiết lập & huỷ trong kế thừa

Trong phương thức thiết lập của lớp dẫn xuất, chúng ta có thể chỉ định phương thức thiết lập nào của lớp cơ sở sẽ được gọi thực hiện. Nếu không chỉ định, phương thức thiết lập mặc định của lớp cơ sở sẽ được gọi.

Phương thức thiết lập & huỷ trong kế thừa


```
class A
{
        public A(){}
        public A(int){}
}
class B : public A
{
        public B(int) //Thực hiện A()
        {}
}
```

Phương thức thiết lập & huỷ trong kế thừa


```
class A
{
        public A(){}
        public A(int){}
}
class B : public A
{
        public B(int) : base(int) //Thực hiện A(int)
        {}
}
```

Bài tập

Thiết kế chương trình quản lý các đối tượng sau trong một Viện khoa học: nhà khoa học, nhà quản lý và NV phòng thí nghiệm. Các thành phần dữ liệu của các đối tượng trên:

- Nhà khoa học: họ tên, năm sinh, bằng cấp, chức vụ, số bài báo đã công bố, số ngày công trong tháng, bậc lương
- Nhà quản lý họ tên, năm sinh, bằng cấp, chức vụ, số ngày công trong tháng, bậc lương
- NV phòng thí nghiệm: họ tên, năm sinh, bằng cấp, lương trong tháng.

Thực hiện các yêu cầu sau:

- Các phương thức thiết lập để nhập liệu, biết rằng nhân viên phòng thí nghiệm lãnh lương khoán, còn lương của nhà khoa học và nhà quản lý bằng số ngày công trong tháng * bậc lương.
- Xuất dữ liệu ra màn hình
- In tổng lương đã chi trả cho từng loại đối tượng.

Tính đa hình

TRẦN VIỆT KHÁNH
Email: tranvietkhanh79@gmail.com

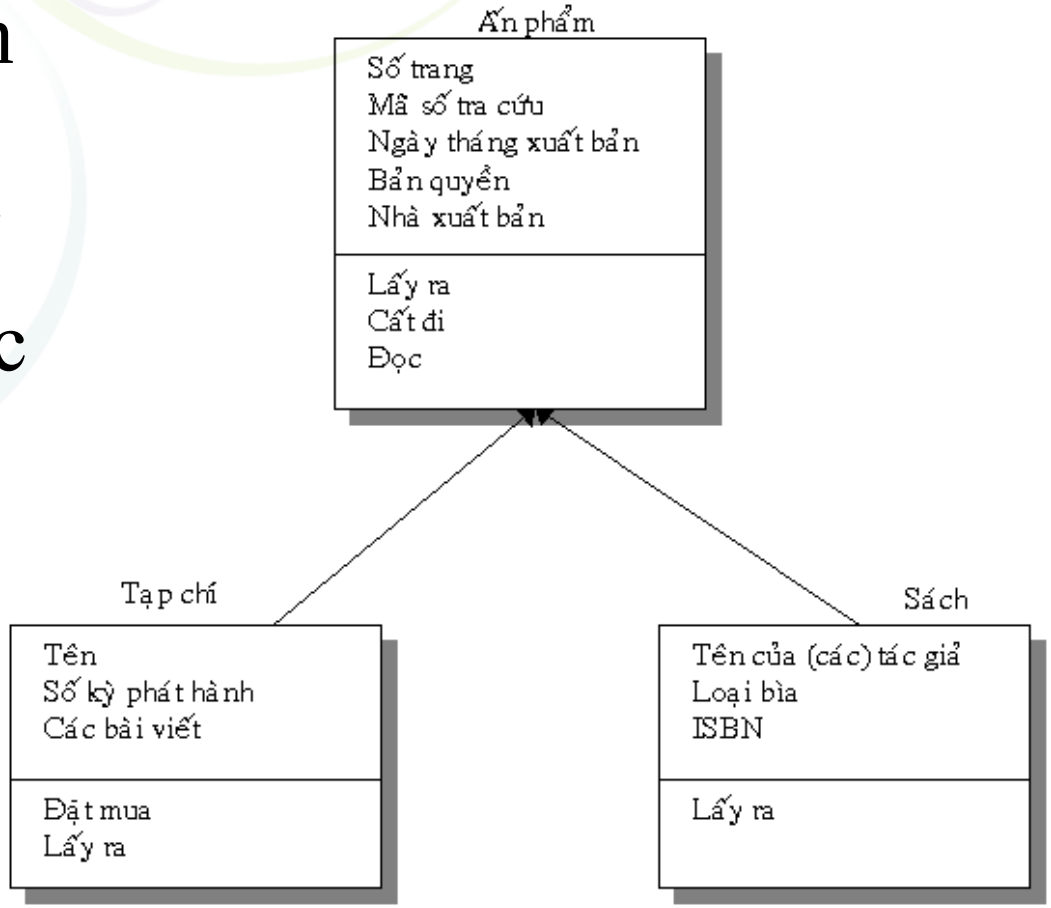
Cập nhật: 04 tháng 09 năm 2016

Nội dung

- Khái niệm về tính đa hình
- Thiết kế lớp trừu tượng
- Các ví dụ minh họa

Đặt vấn đề

Làm thế nào lưu danh sách (mảng) 2 loại ấn phẩm cùng lúc & thực thi đúng hành động "LayRa" của loại ấn phẩm đó ?



Khái niệm tính đa hình

- Tính đa hình là khả năng để cho một thông điệp có thể thực hiện bằng nhiều cách khác nhau tùy thuộc vào đối tượng cụ thể nhận thông điệp.
- Khi một lớp dẫn xuất được tạo ra, nó có thể thay đổi cách thực hiện các phương thức nào đó mà nó thừa hưởng từ lớp cơ sở.

Trừu tượng hóa

- Trừu tượng hóa là khả năng mô tả khái quát các thao tác chung của các lớp đối tượng.
- Đặc tính này giúp cho việc thiết kế lớp mang tính đa hình

Ví dụ

- Nhận xét đoạn code sau

```
static void Main()
{
        AnPham a = new AnPham();
        a.LayRa();

        TapChi t = new TapChi();
        t.LayRa();

        a = t;
        a.LayRa();
}
```

Ví dụ2

- Nhận xét đoạn code sau

```
static void Main()
{
        AnPham[] ds = new AnPham[100];
        for(int i=0;i<100;i++)
        {
                if(Nhập tạp chí ?)
                {
                        ds[i] = new TapChi();
                }
                else
                {
                         ds[i] = new Sach();
                }
        }
}
```

Lớp trừu tượng

Phương thức trừu tượng là phương thức chỉ có tên thôi và nó phải được cài đặt lại ở tất các các lớp kế thừa. Lớp trừu tượng chỉ thiết lập một cơ sở cho các lớp kế thừa mà nó không thể có bất kỳ một thể hiện nào tồn tại

```
abstract class COSO
{
   protected kiểu data1;
   protected kiểu data2;
   public abstract void Method1();
   public abstract void Method2();
}
```

```
class DANXUAT : COSO
{
   private kiểu data3;
   public override void Method1()
  {}
   public override void Method2()
   {}
}
```

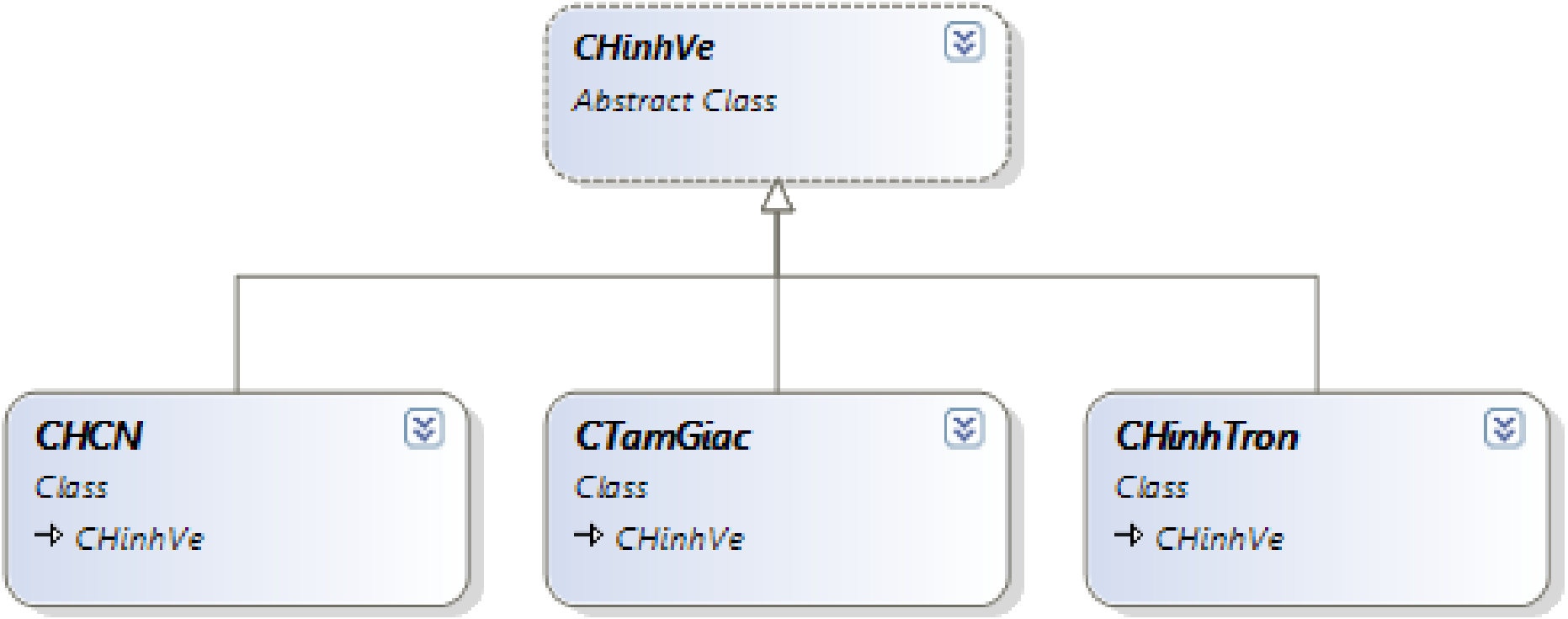
Lớp trừu tượng


```
abstract class Window
{
    protected int top,  left;
    public Window(int top, int left)
    {   this.top = top;   this.left = left; }
    abstract public void DrawWindow( );
}
class ListBox : Window
{
    private string listBoxContents;
    public ListBox(int top, int left, string contents) : base(top, left)
    {   listBoxContents = contents; }
    public override void DrawWindow( )
    {
        Console.WriteLine("Writing string to the listbox: {0}", listBoxContents);
    }
}
```

Lớp trừu tượng


```
public class Button : Window
{
    public Button( int top, int left): base(top, left)
    { }
    public override void DrawWindow( )
    {     Console.WriteLine("Drawing a button at {0}, {1}\n", top, left);  }
}
public class Tester
{
    static void Main( )
    {
        Window[] winArray = new Window[3];
        winArray[0] = new ListBox(1,2,"First List Box");
        winArray[1] = new ListBox(3,4,"Second List Box");
        winArray[2] = new Button(5,6);
        for (int i = 0;i < 3; i++)
        {         winArray[i].DrawWindow( ); }
    }
}
```

Ví dụ


CHinhVe
Abstract Class
CHCN
Class
CHinhVe
CTamGiac
Class
CHinhVe
CHinhTron
Class
CHinhVe

FAQs

XIN CẢM ƠN !