

Chương 3: Lớp và đối tượng

TRẦN VIỆT KHÁNH

Email: tranvietkhanh79@gmail.com

Nội dung

#2

1. Cách cài đặt 1 lớp trong C#
2. Khái niệm hàm khởi tạo (constructors), hàm hủy (destructors)
3. Cách khai thác và sử dụng 1 lớp

1. Cách cài đặt 1 lớp trong C#

#3

- Lớp đối tượng: Định nghĩa các đặc điểm/ thông tin (thuộc tính) và hành động/ chức năng/ (phương thức) chung cho tất cả các đối tượng của cùng một loại.
- Đối tượng: Thể hiện (instance) cụ thể của một lớp đối tượng.

Ví dụ

#4

VD: Lớp SINHVIEN gồm

- Thuộc tính: Họ tên, giới tính, ngày tháng năm sinh, điểm tb, đối tượng ưu tiên, ...
- Phương thức: Học bài, làm bài thi, bài tập, ...

Sinh viên Nguyễn Văn A, Lý Thị B là đối tượng thuộc lớp SINHVIEN

Đối tượng trong LTHĐT

#5

Tách biệt giữa giao tiếp và cài đặt cụ thể

Làm cái gì?

interface

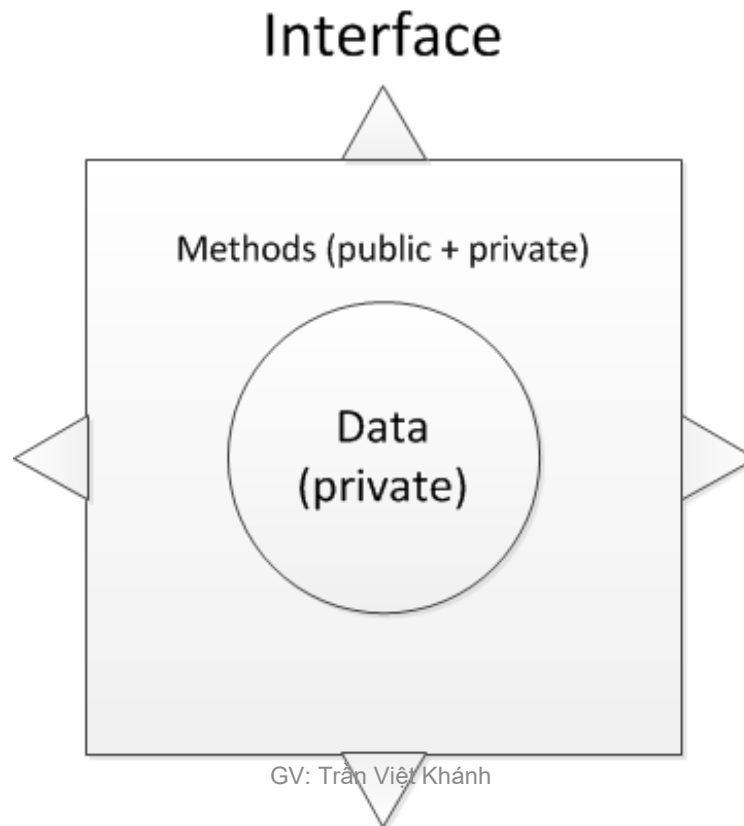
Làm bằng
cách nào?

Implementation

Một cách thể hiện điển hình

#6

Che giấu dữ liệu và các “giải thuật” cụ thể ở bên trong lớp (class)



Cú pháp định nghĩa lớp (class)

#7

```
<từ khóa truy xuất> class <TênLớp>  
{  
    <từ khóa truy xuất> các thuộc tính;  
    <từ khóa truy xuất> phương thức ()  
    {  
        Cài đặt  
    }  
}
```

Từ khóa truy xuất

#8

- **private** (mặc định): Truy xuất trong nội bộ lớp (thường sử dụng cho thuộc tính).
- **protected**: Truy xuất trong nội bộ lớp/ lớp con (được sử dụng cho lớp cơ sở)
- **public**: Truy xuất mọi nơi (thường sử dụng cho phương thức).
- **static**: truy xuất không cần khởi tạo đối tượng của lớp.

VD: định nghĩa lớp CHocSinh

#9

```
public class CHocSinh
{
    private string hoten;
    private int toan, van;
    private float dtb;
    public void Nhap()
    { }
    public void Xuat()
    { }
}
```

Tạo và sử dụng đối tượng

#10

- Tạo đối tượng

<TênLớp> TênĐốiTượng = **new** <TênLớp>();

VD: HOCSINH hsA = new HOCSINH();

- Sử dụng đối tượng

TênĐốiTượng.TênPhươngThức([tham số]);

VD: hsA.Nhap();

hsA.Xuat();

VD: Nhập vào họ tên, điểm văn và điểm toán của 1 học sinh. Tính điểm trung bình và in kết quả

#11

```
public class HOCSINH
{
    private string hoten;
    private int toan, van;
    private float dtb;
    public void Nhap()
    {
        Console.Write("Nhap ho ten: "); hoten = Console.ReadLine();
        Console.Write("Nhap diem van: "); van = int.Parse(Console.ReadLine());
        Console.Write("Nhap diem toan: "); toan = int.Parse(Console.ReadLine());
        dtb = (float)(toan + van) / 2;
    }
    public void Xuat()
    {
        Console.WriteLine("Diem trung binh: {0:0.00}", dtb);
    }
}
```

class Program

{

#12

static void Main(string[] args)

{

HOCSINH hsA = new HOCSINH();

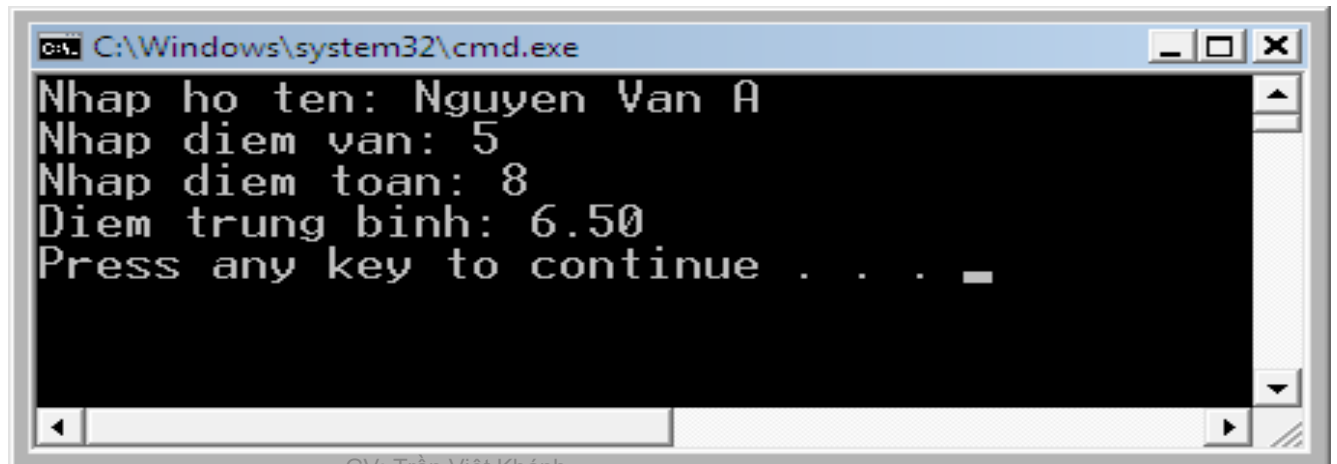
hsA.Nhap();

hsA.Xuat();

}

}

Kết quả:



```
C:\Windows\system32\cmd.exe
Nhap ho ten: Nguyen Van A
Nhap diem van: 5
Nhap diem toan: 8
Diem trung binh: 6.50
Press any key to continue . . . _
```

Chia khai báo lớp thành nhiều file

#13

//File1.cs

```
namespace PC
{
```

```
    partial class A
    {
```

```
        int num = 0;
```

```
        void MethodA()
        {
```

```
            //Cài đặt
```

```
        }
```

```
        partial void MethodC();
```

```
    }
```

```
}
```

//File2.cs

```
namespace PC
{
```

```
    partial class A
    {
```

```
        void MethodB()
        {
```

```
            //Cài đặt
```

```
        }
```

```
        partial void MethodC()
        {
```

```
            //Cài đặt
```

```
        }
```

```
    }
```

```
}
```

Thiết kế thuộc tính

#14

Đối với mỗi đối tượng, xác định các thông tin cần lưu trữ. Sau đó lập bảng mô tả thuộc tính như sau:

Stt	Thuộc tính	Kiểu/ lớp	Ràng buộc	Diễn giải

Nếu có ràng buộc liên thuộc tính thì lập thêm bảng sau:

Stt	Mô tả ràng buộc	Thuộc tính liên quan	Ghi chú

30/09/2018 GV: Trần Việt Khánh

Ràng buộc

#15

Ràng buộc trên lớp là các quy định, quy tắc áp đặt trên các giá trị thuộc tính của đối tượng trong lớp, sao cho đối tượng này thể hiện đúng với thực tế.

Ràng buộc

#16

- Ràng buộc tĩnh: ràng buộc trên giá trị thuộc tính.
- Ràng buộc động: ràng buộc trên biến đổi giá trị thuộc tính.

VD:

- “Lương của nhân viên ít nhất là 1.500.000 đồng” → Ràng buộc tĩnh.
- “Lương của nhân viên chỉ có thể tăng” → Ràng buộc động.

Ràng buộc tĩnh

#17

Gồm 2 loại:

- Ràng buộc trên thuộc tính (Ràng buộc MGT)
- Ràng buộc liên thuộc tính

VD1: Xét lớp điểm ký tự (CDiemKT) trên cửa sổ Console

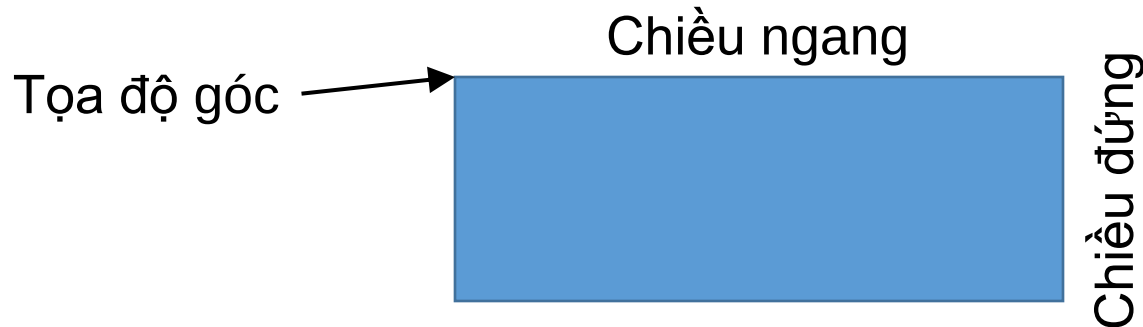
#18

Stt	Thuộc tính	Kiểu/ lớp	Ràng buộc	Diễn giải
1	x	Số nguyên	$0 \leq x < \text{Kích thước ngang}$	Cột
2	y	Số nguyên	$0 \leq y < \text{Kích thước dọc}$	Dòng
3	ch	Ký tự		Ký tự hiển thị

VD2: Xét lớp hình chữ nhật (CHCN) trên cửa sổ Console

Mô tả thuộc tính

#19



Stt	Thuộc tính	Kiểu/ lớp	Ràng buộc	Diễn giải
1	goc	CDiemKT		Toạ độ góc
2	ngang	Số nguyên	$1 < \text{ngang} < \text{Kích thước ngang}$	Chiều ngang
3	dung	Số nguyên	$1 < \text{dung} < \text{Kích thước dọc}$	Chiều đứng

VD2: Xét lớp hình chữ nhật (CHCN) trên cửa sổ Console

Mô tả ràng buộc liên thuộc tính

#20

STT	Mô tả ràng buộc	Thuộc tính liên quan	Ghi chú
1	Tổng của hoành độ góc và m nhỏ hơn kích thước ngang	Goc, m	
2	Tổng của tung độ góc và n nhỏ hơn kích thước dọc	Goc, n	

VD3: Mô tả ràng buộc liên thuộc tính cho lớp CDate

#21

STT	Mô tả ràng buộc	Thuộc tính liên quan	Ghi chú
1	Nếu Th là 4, 6, 9, 11 thì Ng tối đa là 30	Ng, Th	
2	Nếu Th là 2 và Nm nhuận thì Ng tối đa là 29 Nếu Th là 2 và Nm không nhuận thì Ng tối đa là 28	Ng, Th, Nm	

Bài tập

#22

▪ Định nghĩa cấu trúc

```
public struct Sanpham
{
    public string Ma;
    public string Ten;
    public decimal Gia;
    public SanPham(string ma, string ten, decimal gia)
    {
        this.Ma = ma;
        this.Ten = ten;
        this.Gia = gia;
    }
    public string hienThiThongTin()
    {
        return Ma + "\n" + Ten + "\n" + Gia;
    }
}
```

Bài tập

#23

- Khai báo và gán giá trị cho cấu trúc

```
SanPham sp;  
sp.Ma = "sp001";  
sp.Ten = "Dream 100";  
sp.Gia = 20.000.000;
```

```
Console.WriteLine(sp.hienThiThongTin());
```

- Khởi tạo giá trị cho cấu trúc

```
SanPham sp2 = new SanPham("002", "SH", 120.000.000);  
Console.WriteLine(sp2.hienThiThongTin());
```

Bài tập

#24

- Định nghĩa lớp
public class Sanpham
{
.....
}
- Tạo đối tượng từ lớp
Sanpham dtSanpham;
dtSanpham = new Sanpham();

2. Khái niệm hàm khởi tạo (constructors) – hàm hủy (destructors)

#25

- Khởi tạo không có tham số
public Sanpham()
{
}
- Khởi tạo có tham số
public Sanpham(decimal gia)
{
}

Định nghĩa các thuộc tính và phương thức cho lớp

#26

- Định nghĩa các thuộc tính
private int soluong;
public decimal gia;
- Định nghĩa các phương thức
public void Hienthi()
{
}
public int Hienthi()
{
}

Điều khiển truy cập

#27

Phạm vi	Public	Internal	Protected	private
Cùng lớp	Y	Y	Y	Y
Lớp kế thừa	Y	Y	Y	N
Cùng assembly	Y	Y	N	N
Khác assembly	Y	N	N	N

- Điều khiển truy cập mặc định là private.
- Khi build solution, mọi project trong solution được build thành một assembly(chương trình)

Kế thừa

#28

Lớp cơ sở
(lớp cha)

Sản phẩm	
String	Ma
String	Ten
Decimal	Gia
String	hienThiThongTin()

Lớp dẫn xuất
(lớp con)

Sách	
String	Ma
String	Ten
Decimal	Gia
String	hienThiThongTin()
String	Tacgia

Phần mềm	
String	Ma
String	Ten
Decimal	Gia
String	hienThiThongTin()
String	phienban

Cài đặt lớp cơ sở

#29

- Viết từ khóa virtual trước phương thức có thể sẽ được ghi đè ở các lớp dẫn xuất (lớp con)

```
public class SanPham
{
    public string Ma;
    public string Ten;
    public decimal Gia;
    public SanPham(string ma, string ten, decimal gia)
    {
        this.Ma = ma;
        this.Ten = ten;
        this.Gia = gia;
    }
    public virtual string hienthithongtin()
    {
        return Ma + "\n" + Ten + "\n" + Gia;
    }
}
```

Cài đặt lớp dẫn xuất

#30

- Định nghĩa lớp dẫn xuất

public class TenLopDanXuat : TenLopCoSo

- Định nghĩa phương thức khởi tạo gọi đến phương thức khởi tạo của lớp cơ sở

public class TenLop (danhSachThamSo) : base(danhSachThamSo)

- Gọi đến phương thức hoặc thuộc tính của lớp cơ sở

base.tenPhuongThuc(danhSachThamSo)

base.tenThuocTinh

- Ẩn phương thức non-virtual

public new kieuTraVe tenPhuongThuc

- Ghi đè lên phương thức virtual

public override kieuTraVe tenPhuongThuc

Cài đặt lớp dẫn xuất

#31

```
class Sach : SanPham
{
    public string Tacgia;
    public Sach(string ma, string ten, decimal gia, string tacgia) : base(ma, ten, gia)
    {
        this.Tacgia = tacgia;
    }
    public override string hienThiThongTin()
    {
        return base.hienThiThongTin();
    }
}
```

Cài đặt lớp dẫn xuất

#32

- Cách khác để ghi đè lên một phương thức

```
public override string hienThiThongTin()  
{  
    return base.Ma + "\n" + base.Ten + "\n" + base.Gia + "\n" + Tacgia;  
}
```

```
public override string hienThiThongTin()  
{  
    return base.hienThiThongTin() + "\n" + Tacgia;  
}
```

**Gọi đến phương thức
hienThiThongTin() của
lớp cha**

So sánh LT cấu trúc với LT Hướng Đối Tượng

#33

	Giống nhau	Khác nhau
Lập trình cấu trúc	- Dùng để định nghĩa các biến (kiểu giá trị) - Định nghĩa các hàm/phương thức.	- Tiết kiệm bộ nhớ hơn. - Không hỗ trợ thuộc tính tự khởi tạo. - Không hỗ trợ phương thức khởi tạo không có tham số. - Các thành viên chỉ được gọi khi được khởi tạo
Lập trình hướng đối tượng	- Dùng để định nghĩa các biến (kiểu đối tượng) - Định nghĩa các hàm/phương thức.	- Tốn bộ nhớ. - Hỗ trợ thuộc tính tự khởi tạo. - Hỗ trợ phương thức khởi tạo không có tham số. - Hỗ trợ cơ chế ghi đè virtual, override.

Bài tập: thiết kế thuộc tính các lớp

#34

- Lớp thời gian CTime
- Lớp ngày tháng năm CDate
- Lớp phân số CPhanSo
- Lớp CDaThuc (Đa thức 1 ẩn)

$$P_n(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + \dots + a_nx^n$$

- Lớp đường thẳng trong mặt phẳng
CDuongThang

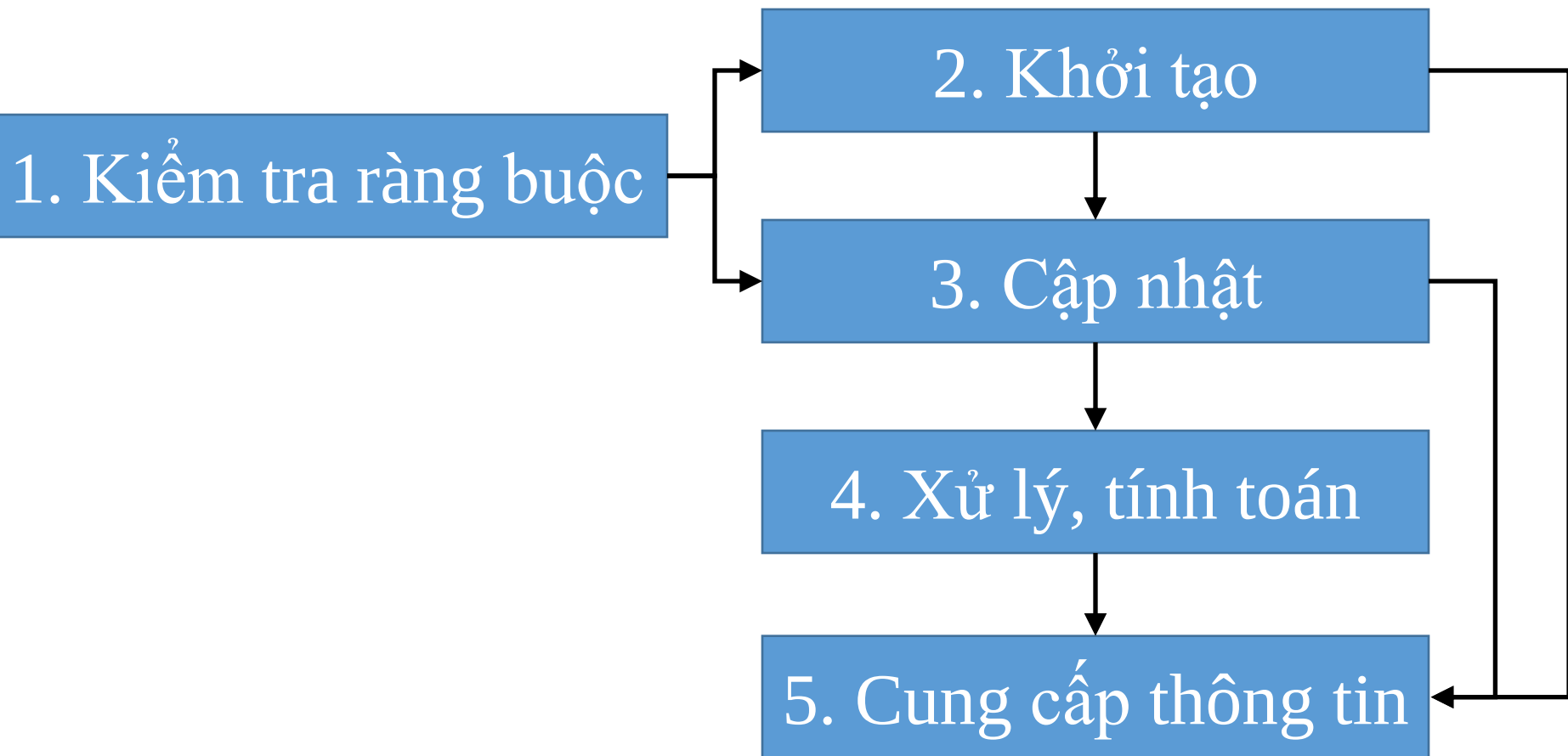
3. Cách khai thác và sử dụng 1 lớp

#35

1. **Nhóm kiểm tra ràng buộc:** Kiểm tra tính hợp lệ giá trị thuộc tính của đối tượng
2. **Nhóm khởi tạo:** Cung cấp giá trị ban đầu cho đối tượng
3. **Nhóm cập nhật:** Thay đổi giá trị thuộc tính của đối tượng
4. **Nhóm xử lý tính toán:** Xử lý tính toán các yêu cầu từ thông tin của đối tượng
5. **Nhóm cung cấp thông tin:** Cung cấp thuộc tính nội bộ của đối tượng

Thiết kế các hành động của lớp

#36



Mẫu thiết kế phương thức ràng buộc

#37

Mẫu: public **bool** KiemTra... (tham số)

- Giá trị trả về
 - **true**: Thoả ràng buộc.
 - **false**: Không thoả ràng buộc.
- Tham số
 - *Ràng buộc miền giá trị*: Chỉ có 1 tham số ứng với tham số cần kiểm tra.
 - *Ràng buộc liên thuộc tính*: Có tham số là các thuộc tính liên quan.

Mẫu thiết kế phương thức ràng buộc

#38

- Tên phương thức

Bắt đầu bằng chữ **KiemTra**

- *Ràng buộc miền giá trị*: Ghép thêm tên thuộc tính
- *Ràng buộc liên thuộc tính*: Ghép thêm số thứ tự ràng buộc

VD thiết kế phương thức kiểm tra ràng buộc cho lớp CHCN

#39

```
class CHCN  
{  
    private CDiem Goc;  
    private int ngang, dung;  
    public bool KiemTraNgang(int ng);  
    public bool KiemTraDung(int d);  
    public bool KiemTra1(int ng, CDiem X);  
    public bool KiemTra2(int d, CDiem Y);  
}
```

Cài đặt phương thức khởi tạo và cập nhật

#40

- Các phương thức thuộc nhóm khởi tạo và cập nhật có liên quan đến ràng buộc phải được bổ sung thêm kiểm tra ràng buộc
- Việc kiểm tra tham số thoả hoặc không thoả ràng buộc bằng cách gọi phương thức kiểm tra ràng buộc tương ứng

public **bool** Tên hàm (Tham số)

#41

//Trả về true: thực hiện được

//Trả về false: không thực hiện được

bool kq = false;

if(Tham số **thoả** ràng buộc)

{

 gán giá trị tương ứng cho thuộc tính của lớp

 kq=true;

}

return kq;

}

hoặc

#42

```
public bool Tên hàm ( Tham số )
```

```
{
```

```
//Trả về true: thực hiện được, false: không  
thực hiện được
```

```
if(Tham số không thoả ràng buộc)
```

```
    return false;
```

```
    gán giá trị tương ứng cho thuộc tính của lớp
```

```
    return true;
```

```
}
```

```
public bool CapNhatX(int xx)
{
```

#43

```
    if(!KiemTraX(xx))
```

```
        return false;
```

```
    x=xx;
```

```
    return true;
```

```
}
```

```
public bool CapNhatM(int mm)
```

```
{
```

```
    if(!KiemTra1(mm, Goc))
```

```
        return false;
```

```
    m=mm;
```

```
    return true;
```

```
}
```

VD1: Thiết kế các hành động của lớp CDiemKT

#44

1. Nhóm kiểm tra ràng buộc

```
public bool KiemTraX(int xx);
```

```
public bool KiemTraY(int yy);
```

2. Nhóm khởi tạo

```
public void Nhap();
```

```
public bool KhoiTao (int xx, int yy, char cc);
```

```
public void PhatSinh();
```

VD1: Thiết kế các hành động của lớp CDiemKT

3. Nhóm cập nhật

#45

//Trực tiếp

public bool CapNhatX(int xx);

public bool CapNhatY(int yy);

public void CapNhatCh(char c);

//Gián tiếp

public bool DichPhai(uint k);

public bool DichTrai(uint k);

public bool DichLen(uint k);

public bool DichXuong(uint k);

public bool DichXien1(uint k);

public bool DichXien2(uint k);

VD1: Thiết kế các hành động của lớp CDiemKT

#46

4. Nhóm xử lý tính toán

```
public double KhoangCach(CDiemKT M);  
public int KhoangCachX(CDiemKT M);  
public int KhoangCachY(CDiemKT M);
```

5. Nhóm cung cấp thông tin

```
public void Xuat();  
public void Xoa();  
public int GiaTriX();  
public int GiaTriY();  
public char GiaTriCh();
```

VD2: Thiết kế các hành động của lớp CHCN

#47

1. Nhóm kiểm tra ràng buộc

```
public bool KiemTraM(int mm);
```

```
public bool KiemTraN(int nn);
```

2. Nhóm khởi tạo

```
public void Nhap();
```

```
public bool KhoiTao(CDiemKT M,int cng, int cd);
```

```
public bool KhoiTao(int x, int y, int cng, int cd);
```

```
public void KhoiTao(CDiemKT X, CDiemKT Y);
```

```
public void PhatSinh();
```

VD2: Thiết kế các hành động của lớp CHCN

#48

3. Nhóm cập nhật

//Trực tiếp

```
public bool CapNhatGoc(CDiemKT M);
```

```
public bool CapNhatNgang(int cng);
```

```
public bool CapNhatDung(int cd);
```

VD2: Thiết kế các hành động của lớp CHCN

3. Nhóm cập nhật

#49

//Gián tiếp

public bool DichPhai(int k);

public bool DichTrai(int k);

public bool DichLen(int k);

public bool DichXuong(int k);

public bool TangNgang(int k);

public bool GiamNgang(int k);

public bool TangDung(int k);

public bool GiamDung(int k);

public bool XoayThuan();

public void XoayNghich();

VD2: Thiết kế các hành động của lớp CHCN

#50

4. Nhóm xử lý tính toán

```
public int XetViTri(CDiemKT M);
```

// -1: Bên trong, 0: Trên cạnh, 1: Bên ngoài

```
public int KhoangCachX(CDiemKT M);
```

```
public int KhoangCachY(CDiemKT M);
```

VD2: Thiết kế các hành động của lớp CHCN

#51

5. Nhóm cung cấp thông tin

```
public void Xuat();  
public void Xoa();  
public CDiemKT ToaDoGoc();  
public int ChieuNgang();  
public int ChieuDung();  
public int ChuVi();  
public long DienTich();  
public double DuongCheo();
```

Bài tập

#52

Thiết kế các hành động cho các lớp sau:

- Lớp phân số (CPhanSo)
- Lớp thời gian (CTime)
- Lớp ngày tháng năm (CDate)

Cài đặt phương thức

#53

- Truy xuất và cập nhật dữ liệu (property get-set)
- Trùng tên phương thức (overload)
- Phương thức thiết lập (constructor)
- Cài đặt phép toán (operator)
- Kỹ thuật bổ sung, thay thế chức năng phương thức có sẵn (override)

Truy xuất và cập nhật dữ liệu

#54

```
class CViDu
{
    private int a, b;
    public void Nhap()
    {
    }
    public void Xuat()
    {
    }
}
```

30/09/2018

```
class Program
{
    static void Main(string[] args)
    {
        CViDu vd = new CViDu();
        vd.a = 5; // Lỗi
        vd.b = 4; //Lỗi
        vd.Xuat();
    }
}
```

GV: Trần Việt Khánh

Mẫu cài đặt property

public <kiểu thuộc tính> Tên property

#55

{

get

{

return <tên thuộc tính>;

}

set

{

if(kiểm tra value thỏa đk)

<tên thuộc tính> = value;

}

}

Mẫu cài đặt property

#56

- get : Đọc thuộc tính
- set : Gán giá trị cho thuộc tính
- Tên property : Đặt tên bất kỳ theo quy ước nhưng nên đặt dễ nhớ (tốt nhất là trùng tên với tên thuộc tính và ký tự đầu viết hoa)

```
class CViDu
```

```
{
```

```
#57
```

```
private int a, b;
```

```
public int A
```

```
{
```

```
    get{return a;}
    set{a=value;}
```

```
    }
```

```
public int B
```

```
{
```

```
    get {return b;}
    set { b = value; }
```

```
    }
```

```
}
```

```
}
```

```
class Program
```

```
{
```

```
    static void Main(string[] args)
```

```
{
```

```
        CViDu vd = new CViDu();
```

```
        vd.A = 5;
```

```
        vd.B = 4;
```

```
    }
```

```
}
```

Bài tập

#58

Cài đặt các property cho các lớp sau:

- Lớp phân số (CPhanSo)
- Lớp thời gian (CTime)
- Lớp ngày tháng năm (CDate)

Phương thức trùng tên

#59

Các phương thức có thể có tên trùng nhau ngay cả các phương thức này ở cùng trong 1 lớp nhưng trong số các tham số phải có ít nhất 1 tham số khác

- khác về KDL (nếu cùng số lượng tham số)
- hoặc khác về số lượng tham số

Phương thức trùng tên

#60

```
public void Nhap()
```

```
{ }
```

```
public void Nhap(int aa)
```

```
{ }
```

```
public void Nhap(float aa)
```

```
{ }
```

```
public void Nhap(int aa, float bb)
```

```
{ }
```

```
public int Nhap()
```

Phương thức trùng tên

#61

Phân biệt khi gọi phương thức ?

1. public void PhuongThuc(int a) {}

2. public void PhuongThuc(float a) {}

void Main()

{

int x=7;

float y=6.0;

PhuongThuc(x); //Gọi phương thức số 1

PhuongThuc(y); //Gọi phương thức số 2

}

Phương thức thiết lập

#62

Tự động thực hiện khi đối tượng được sinh ra, các phương thức này có các đặc điểm:

- Không có giá trị trả về.
- Tên phương thức trùng với tên lớp.
- Có thể có hoặc không có tham số.
- Sử dụng phải có từ khóa **new**

Mẫu phương thức thiết lập

#63

```
class <TênLớp>
{
    //Thuộc tính
    //Các phương thức
    public <TênLớp>([tham số])
    {
        Gán giá trị cho thuộc tính
    }
}
```

```
class CViDu
```

```
{
```

```
    private int a, b;
```

```
#64
```

```
    public CViDu()
```

```
    {
```

```
        a = 4;
```

```
        b = 2;
```

```
    }
```

```
    public CViDu(int aa, int bb)
```

```
    {
```

```
        a = aa;
```

```
        b = bb;
```

```
    }
```

```
    public void Xuat()
```

```
    {
```

```
        Console.WriteLine("a = {0}, b = {1}", a, b);
```

```
    }
```

```
} 30/09/2018
```

```
class Program
```

```
{
```

```
    static void Main(string[] args)
```

```
    {
```

```
        CViDu vd = new CViDu();
```

```
        vd.Xuat();
```

```
        vd = new CViDu(1, 23);
```

```
        vd.Xuat();
```

```
    }
```

```
}
```

Sử dụng từ khóa **this**

#65

```
class CViDu
{
    int a, b;
    public void Gan(int a, int b)
    {
        a = a;
        b = b;
    }
    public void Xuat()
    {
        Console.WriteLine("a={0}, b={1}", a, b);
    }
}
```

```
class Program
{
    static void Main(string[] args)
    {
        CViDu vd = new CViDu();
        vd.Gan(21, 9);
        vd.Xuat();
    }
}
```

Trường hợp tên tham số trùng với tên thuộc tính của đối tượng ta dùng từ khóa **this**. Từ khoá **this** được dùng để tham chiếu đến chính bản thân của đối tượng đó

#66

```
class CViDu
{
    int a, b;
    public void Gan(int a, int b)
    {
        this.a = a;
        this.b = b;
    }
    public void Xuat()
    {
        Console.WriteLine("a={0}, b={1}", a, b);
    }
}
```

Bài tập

#67

Cài đặt các phương thức thiết lập cho các lớp sau:

- Lớp phân số (CPhanSo)
- Lớp thời gian (CTime)
- Lớp ngày tháng năm (CDate)

Cài đặt phép toán

#68

Giả sử lớp phân số (CPhanSo) có phương thức cộng (Cong) và nhân (Nhan) hai phân số. Khi đó, ta muốn cộng 2 phân số a và b lưu vào phân số tổng c, ta phải viết là:

$$c = a.Cong(b);$$

Tương tự cho trường hợp nhân

$$c = a.Nhan(b);$$

Cài đặt phép toán

#69

Nếu muốn viết một cách tự nhiên như sau:

$$\mathbf{c = a + b;}$$

$$\mathbf{c = a * b;}$$

Thì phải cài đặt phương thức thông qua các ký hiệu phép toán (operator)

Mẫu cài đặt phép toán

#70

**public static <KDL> operator ký hiệu
(TênLớp trái, TênLớp phải)**

- Ký hiệu: Gồm các ký hiệu phép toán số học, logic và so sánh
- Trái: Tên tham số sẽ nằm bên trái phép toán
- Phải: Tên tham số sẽ nằm bên phải phép toán
- KDL: bool nếu operator so sánh
TênLớp nếu operator tính toán

VD: Cài đặt phép toán + cho lớp CPhanSo

#71

```
public static CPhanSo operator + (CPhanSo  
ps1, CPhanSo ps2)  
{  
    //Cài đặt  
}
```

Giả sử có 2 phân số a, b và phân số tổng c. Yêu cầu thực hiện như sau:

$$c = a + b;$$

Không dùng operator

class CPhanSo

{

#72

private int tuso, mauso;

public CPhanSo(int t, int m)

{

tuso = t;

mauso = m;

}

public CPhanSo Cong(CPhanSo ps2)

{

int tu = tuso*ps2.mauso + ps2.tuso*mauso;

int mau = mauso*ps2.mauso;

CPhanSo c = new CPhanSo(tu, mau);

return c.RutGon();

}

}

class Program

#73

```
{  
    static void Main(string[] args)  
    {  
        CPhanSo a = new CPhanSo(3, 5);  
        a.Xuat();  
        CPhanSo b = new CPhanSo(1, 2);  
        b.Xuat();  
        CPhanSo c=new CPhanSo();  
        c = a.Cong(b);  
        Console.WriteLine("Ket qua: ");  
        c.Xuat();  
    }  
}
```

Dùng operator

class CPhanSo

{

#74

private int tuso, mauso;

public CPhanSo(int t, int m)

{ tuso = t; mauso = m;

}

public static CPhanSo operator +(CPhanSo ps1, CPhanSo ps2)

{

int tu = ps1.tuso*ps2.mauso + ps2.tuso*ps1.mauso;

int mau = ps1.mauso*ps2.mauso;

CPhanSo c = new CPhanSo(tu, mau);

return c.RutGon();

}

public void Xuat()

{ Console.WriteLine("{0}/{1}", tuso, mauso);

}

} 30/09/2018

```
class Program
```

```
{  
#75
```

```
{  
    static void Main(string[] args)  
    {  
        CPhanSo a = new CPhanSo (3, 5);  
        a.Xuat();  
        CPhanSo b = new CPhanSo(1, 2);  
        b.Xuat();  
        CPhanSo c=new CPhanSo();  
        c = a + b;  
        Console.WriteLine("Ket qua: ");  
        c.Xuat();  
    }  
}
```

Bài tập

#76

Cài đặt các operator cho các lớp sau:

- Lớp phân số (CPhanSo)
- Lớp thời gian (CTime)
- Lớp ngày tháng năm (CDate)

Cài đặt bổ sung chức năng phương thức xuất

#77

- Phương thức Write() chỉ xuất những dữ liệu có kiểu cơ bản. Ví dụ:

```
int a = 4;
```

```
float b = 7;
```

```
Console.Write("a={0}, b={1}", a, b);
```

- Đối với đối tượng thì phương thức Write() không thực hiện được, giả sử có lớp phân số (CPhanSo)

```
CPhanSo ps = new CPhanSo(5, 3);
```

```
Console.Write("Phan so: " + ps);
```

Cài đặt bổ sung chức năng phương thức xuất

#78

Để sử dụng được `Console.Write("Phan so: "+ps)` phải cài đặt lại phương thức `ToString()` như sau:

```
public override string ToString()
```

```
{
```

```
    string s=Tạo chuỗi cần xuất;
```

```
    return s;
```

```
} 10/09/2018
```

GV: Trần Việt Khánh

Cài đặt bổ sung chức năng phương thức xuất

#79

```
class CPhanSo
{
    private int tuso, mauso;
    public CPhanSo(int t, int m)
    {
        tuso = t;
        mauso = m;
    }
    public override string ToString()
    {
        string s = tuso.ToString() + "/" + mauso.ToString();
        return s;
    }
}
```

Cài đặt bổ sung chức năng phương thức xuất

#80

```
static void Main(string[] args)
{
    CPhanSo a = new CPhanSo(3, 5);
    Console.Write("Phan so: " + a);
}
```

Bài tập

#81

Cài đặt bổ sung chức năng phương thức xuất
cho các lớp sau:

- CDiemKT
- CHCN
- CDate
- CTime

FAQs

#82



THANKS FOR LISTENING !