LAB 6. ĐA HÌNH THÔNG QUA SỰ KẾ THỪA

THỜI LƯỢNG : 4 TIẾT

A. Mục tiêu

- Giúp sinh viên hiểu rõ tính đa hình trong lập trình hướng đối tượng. Ở đây, tính đa hình được thực hiện qua việc kế thừa, sử dụng lớp abstract và ghi đè phương thức.
- Sau khi hoàn thành bài thực hành này, sinh viên cần:
  - Hiểu rõ và phân biệt sự khác nhau giữa từ khóa new và override.
  - Biết cách tạo lớp trừu tượng và tạo lớp dẫn xuất từ lớp trừu tượng.
  - Hiểu rõ và phân biệt được phương thức ảo và phương thức trừu tượng.
  - Biết cách viết tạo và sử dụng các đối tượng trong cây phân cấp kế thừa.
  - Nắm vững cơ chế gọi hàm từ thể hiện của các lớp trong cây phân cấp kế thừa.
  - Áp dụng được tính kế thừa và đa hình vào các bài toán thực tế.

B. Yêu cầu

- Sinh viên phải trả lời đầy đủ các câu hỏi (nếu có) hoặc chụp màn hình kết quả, ghi lại vào tập tin Word theo yêu cầu ghi trong phần hướng dẫn thực hành.
- Đặt tên tập tin Word theo dạng: Lab06_MSSV_HoVaTen.rar.
- Tạo thư mục, đặt tên là MSSV_Lab06 để lưu tất cả bài làm trong bài thực hành này.
- Sinh viên phải hoàn thành tối thiểu 2 bài tập bắt buộc.
- Phải tạo một dự án riêng cho mỗi phần hoặc mỗi bài tập.
- Cuối buổi thực hành, nén thư mục trên & nộp bài qua email: phucnguyen5555@gmail.com

C. Hướng dẫn thực hành

Trong bài thực hành trước, ta đã tạo các lớp trong cây phân cấp kế thừa. Tuy nhiên, việc gọi các phương thức khó khăn vì phải biết (ép) kiểu của đối tượng trước khi gọi. Lý do là việc sử dụng từ khóa new ở phương thức trong lớp dẫn xuất sẽ làm ẩn (hay che giấu) phương thức ở lớp cơ sở chứ không ghi đè. Vì vậy, khi đối tượng tron có kiểu Hinhtron nhưng tham chiếu đến một thể hiện của lớp HinhTru, phương thức được gọi từ đối tượng tron không phải là phương thức được định nghĩa trong lớp HinhTru mà chính là phương thức trong lớp HinhTron. Để giải quyết vấn đề này, ta sử dụng tính đa hình. Tính đa hình có thể được thực hiện bởi việc ghi đè các hàm ảo, kế thừa lớp trừu tượng hoặc thực thi một giao diện (interface).

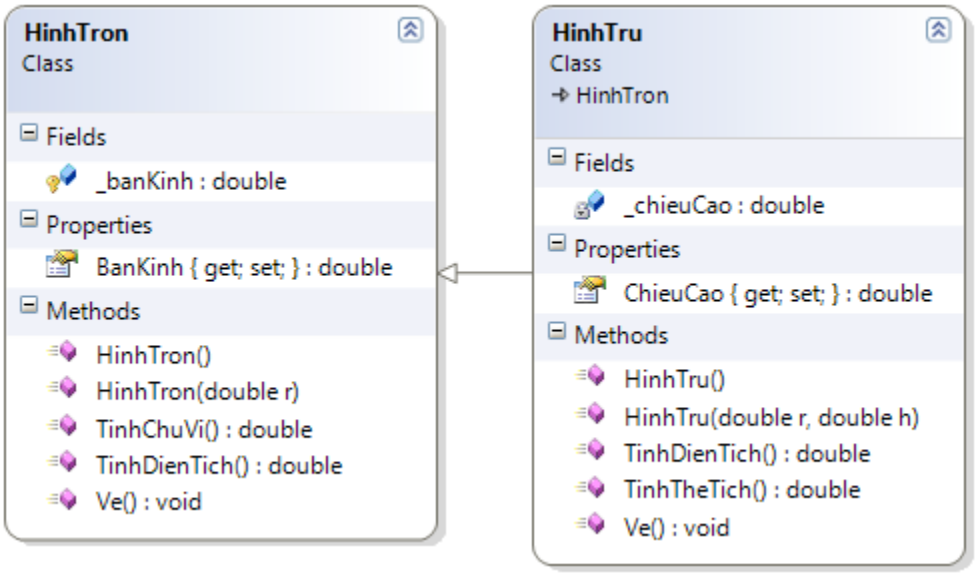
Trong bài Lab này, ta sẽ thực thi tính đa hình bằng cách ghi đè các phương thức ảo và sử dụng lớp trừu tượng. Interface sẽ được đề cập trong Lab08.

1. Đa hình thông qua việc ghi đè phương thức ảo

Bước 1. Tạo một thư mục, đặt tên theo dạng MSSV-Lab06. Ví dụ: 1210234-Lab06.

Bước 2. Tạo dự án dạng Console Application, đặt tên là Lab06_Bai01_GhiDePhuongThucAo.

Bước 3. Tạo 2 lớp HinhTru và HinhTron tương tự như Lab05_Bai01.



Bước 4. Cài đặt lớp HinhTron như sau (chú ý có từ khóa virtual):

```
class HinhTron
{
    // Khai báo các biến thành viên
    protected double _banKinh;

    // Định nghĩa thuộc tính
    public double BanKinh
    {
        get { return _banKinh; }
        set { _banKinh = value; }
    }

    // Định nghĩa phương thức khởi tạo
    public HinhTron()
    {
        _banKinh = 0;
    }

    public HinhTron(double r)
    {
        _banKinh = r;
    }
```

```
    // Định nghĩa phương thức tính chu vi
    public virtual double TinhChuVi()
    {
        return 2 * Math.PI * _banKinh;
    }

    // Định nghĩa phương thức tính diện tích
    public virtual double TinhDienTich()
    {
        return _banKinh * _banKinh * Math.PI;
    }

    // Định nghĩa hàm (giả) vẽ hình tròn
    public virtual void Ve()
    {
        Console.WriteLine("Ve hinh tron ban kinh : {0}",
                          _banKinh);
    }
}
```

Bước 5. Cài đặt lớp HinhTru như trong đoạn mã sau (chú ý từ khóa override):

```
class HinhTru : HinhTron
{
    // Khai báo các biến thành viên
    private double _chieuCao;

    // Định nghĩa thuộc tính
    public double ChieuCao
    {
        get { return _chieuCao; }
        set { _chieuCao = value; }
    }

    // Định nghĩa phương thức khởi tạo
    public HinhTru() : base() {
        _chieuCao = 0;
    }

    // Ở đây, ta dùng base(r) để gọi phương
    // thức khởi tạo của lớp HinhTron
    public HinhTru(double r, double h) : base(r)
    {
        _chieuCao = h;
    }

    // Định nghĩa phương thức tính diện tích
    public override double TinhDienTich()
    {
        double dtThan = base.TinhChuVi() * _chieuCao;
        return 2 * base.TinhDienTich() + dtThan;
    }
```

```
    // Định nghĩa phương thức tính diện tích
    public override double TinhDienTich()
    {
        double dtThan = base.TinhChuVi() * _chieuCao;
        return 2 * base.TinhDienTich() + dtThan;
    }

    // Định nghĩa hàm tính thể tích hình trụ
    // Dùng base để gọi hàm tính diện tích ở lớp HinhTron
    public double TinhTheTich()
    {
        return base.TinhDienTich() * _chieuCao;
    }

    // Định nghĩa hàm (giả) vẽ hình trụ
    public override void Ve()
    {
        Console.WriteLine("Ve hinh tru cao : {0}",
                          _chieuCao);
    }
}
```

Bước 6. Trong hàm Main, tạo thể hiện của hai lớp HinhTron và HinhTru như sau

```
// Tạo thể hiện của lớp HinhTru
HinhTru tru = new HinhTru(5, 10);

// Ép kiểu đối tượng tru về kiểu HinhTron
HinhTron tron = tru;
```

Bước 7. Gọi hàm xuất thông tin và hàm tính chu vi

```
// gọi hàm vẽ hình
tru.Ve();
tron.Ve();

// gọi hàm tính chu vi
Console.WriteLine(tron.TinhChuVi());
Console.WriteLine(tru.TinhChuVi());
```

Bước 8. Chạy chương trình và ghi nhận kết quả. Giải thích tại sao có kết quả đó?

Bước 9. Tại sao trong lớp HinhTru không định nghĩa phương thức TinhChuVi, cũng không ghi đè phương thức đó nhưng vẫn có thể gọi phương thức được?

Bước 10. Gọi hàm tính diện tích của các hình như sau:

```
// gọi hàm tính diện tích
Console.WriteLine(tron.TinhDienTich());
Console.WriteLine(tru.TinhDienTich());
```

Bước 11. Chạy lại chương trình và ghi nhận kết quả. So sánh với kết quả ở bước 20, phần 1, Lab05 và cho nhận xét.

Bước 12. Có thể gọi phương thức TinhTheTich từ đối tượng tron được không? Tại sao?

Bước 13. Thử lại với phép ép kiểu xuống tương tự như đã làm trong Lab5 như sau

```
HinhTru ong = (HinhTru) tron;
// Hoặc: HinhTru ong = tron as HinhTru;

Console.WriteLine(ong.TinhDienTich());
```

Bước 14. Chạy chương trình và ghi nhận kết quả. So sánh với kết quả ở bước 11 và nhận xét.

Bước 15. Thay thế hai lệnh thứ 2 trong bước 6 bởi dòng lệnh sau:

```
HinhTron tron = new HinhTru(5, 10);
```

Bước 16. Chạy lại chương trình và kiểm tra xem kết quả có thay đổi hay không.

Bước 17. Giải thích ý nghĩa của lệnh ở bước 15.

Bước 18. Trong lớp HinhTru, thay từ khóa override bởi từ khóa new.

Bước 19. Chạy lại chương trình và ghi nhận kết quả. Giải thích tại sao có kết quả đó?

Bước 20. Hãy cài đặt lại Lab05_Bai02 bằng cách dùng phương thức ảo và ghi đè phương thức. Cho biết những lớp và phương thức nào cần phải sửa đổi?

2. Tạo lớp trừu tượng và phương thức trừu tượng

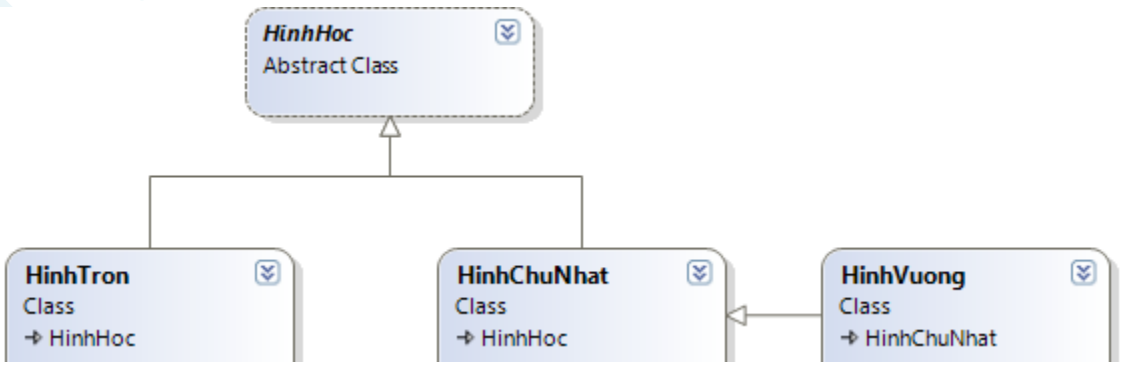
Một giải pháp khác để thực thi tính đa hình là sử dụng lớp và phương thức trừu tượng. Phần này sẽ hướng dẫn cách cài đặt lại Lab05_Bai02. Trong đó, lớp HinhHoc chuyển thành lớp trừu tượng.

Bước 1. Tạo dự án dạng Console Application, đặt tên là Lab06_Bai02_SuDungLopTruuTuong.

Bước 2. Tạo một enum, đặt tên là LoaiHinh và cài đặt enum như sau:

```
public enum LoaiHinh { TatCa, HinhTron, HinhVuong, HinhChuNhat }
```

Bước 3. Tạo bốn lớp mới, đặt tên lần lượt là HinhHoc, HinhTron, HinhVuong, HinhChuNhat. Ta sẽ tạo một cây phân cấp kế thừa như hình bên dưới.



Bước 4. Cài đặt lớp HinhHoc như sau (chú ý từ khóa **abstract**):

```
public abstract class HinhHoc
{
    // Khai báo biến thành viên
    protected double canh;

    // Định nghĩa phương thức khởi tạo
    protected HinhHoc(double cd)
    {
        canh = cd;
    }

    // Khai báo hàm tính diện tích
    public abstract double TinhDienTich();

    // Khai báo hàm xuất thông tin
    public abstract void Xuat();
}
```

Bước 5. Trong hàm Main, nhập đoạn mã sau:

```
HinhHoc hinh = new HinhHoc(10);
```

Bước 6. Chương trình có báo lỗi gì không? Nếu có, ghi lại lỗi và giải thích tại sao có lỗi đó?

Bước 7. Xóa đoạn mã trong bước 5 và cài đặt tiếp lớp HinhTron như sau (chú ý từ **override**):

```
public class HinhTron : HinhHoc
{
    // Định nghĩa thuộc tính
    public double BanKinh
    {
        get { return canh; }
    }
    // Định nghĩa phương thức khởi tạo
    public HinhTron(double banKinh) : base(banKinh)
    {
    }

    // Định nghĩa hàm tính diện tích
    public override double TinhDienTich()
    {
        return canh * canh * Math.PI;
    }
    // Định nghĩa hàm xuất thông tin hình tròn
    public override void Xuat()
    {
        Console.WriteLine("Hinh tron ban kinh {0}, dien tich = {1}",
                          canh, TinhDienTich());
    }
}
```

Bước 8. Trong hàm Main, nhập đoạn mã sau

```
HinhTron vong = new HinhTron(5);
HinhHoc tron = new HinhTron(5);

vong.Xuat();
tron.Xuat();

Console.WriteLine(vong.TinhDienTich());
Console.WriteLine(tron.TinhDienTich());
```

Bước 9. Chạy chương trình và ghi nhận kết quả. Tại sao kết quả xuất ra ứng với 2 đối tượng vong và tron đều giống nhau?

Bước 10. Tiếp tục cài đặt lớp HinhChuNhat như sau:

```
public class HinhChuNhat : HinhHoc
{
    // Khai báo biến thành viên. Xem biến canh là chiều dài,
    // ta phải thêm 1 biến chiều rộng
    protected double rong;

    // Định nghĩa thuộc tính
    public double ChieuDai
    {
        get { return canh; }
    }

    public double ChieuRong
    {
        get { return rong; }
    }

    // Định nghĩa phương thức khởi tạo
    public HinhChuNhat(double dai, double rong) : base(dai)
    {
        this.rong = rong;
    }

    // Định nghĩa phương thức tính diện tích
    public override double TinhDienTich()
    {
        return canh * rong;
    }

    // Định nghĩa phương thức xuất thông tin HCN
    public override void Xuat()
    {
        Console.WriteLine("Hinh chu nhat dai {0}, " +
        "rong {1}, dien tich = {2}", canh, rong, TinhDienTich());
    }
}
```

Bước 11. Cài đặt lớp HinhVuong

```
public class HinhVuong : HinhChuNhat
{
    // định nghĩa phương thức khởi tạo
    public HinhVuong(double canh) : base(canh, canh)
    {
    }

    // Định nghĩa phương thức xuất thông tin HCN
    public override void Xuat()
    {
        Console.WriteLine("Hinh vuong co canh {0}, " +
                "dien tich = {1}", canh, TinhDienTich());
    }
}
```

Bước 12. Trong hàm Main, xóa bỏ đoạn mã đã nhập ở bước 8, nhập đoạn mã sau

```
// Tạo thể hiện của các loại hình
HinhHoc tron = new HinhTron(5);
HinhHoc vuong = new HinhVuong(5);
HinhHoc cnhat = new HinhChuNhat(5, 3);

// Xuất thông tin của các hình
tron.Xuat();
vuong.Xuat();
cnhat.Xuat();
```

Bước 13. Chạy chương trình và cho biết có xuất đúng thông tin của các hình không?

Bước 14. So sánh với kết quả ở bước 13, phần 3 của Lab05_Bai02. Có nhận xét gì về cách gọi hàm so với bước 14, phần 3, Lab05_Bai02?

Bước 15. Hãy viết mã lệnh để xuất diện tích của 3 hình trên và ghi lại kết quả.

3. Cài đặt lớp danh sách hình học (DanhSachHinhHoc)

Bước 16. Tạo một lớp mới, đặt tên là DanhSachHinhHoc, cài đặt lớp này như sau:

```
public class DanhSachHinhHoc
{
    // Khai báo các biến thành viên
    private HinhHoc[] danhSach;
    private int soLuong;

    // Định nghĩa thuộc tính
    public int SoLuong
    {
        get { return soLuong; }
    }
```

```
    // Định nghĩa chỉ mục
    public HinhHoc this[int index]
    {
        get { return danhSach[index]; }
        set { danhSach[index] = value; }
    }

    // Định nghĩa phương thức khởi tạo
    public DanhSachHinhHoc()
    {
        danhSach = new HinhHoc[100];
        soLuong = 0;
    }

    // Định nghĩa hàm thêm một hình vào danh sách
    public void Them(HinhHoc hinh)
    {
        danhSach[soLuong] = hinh;
        soLuong++;
    }
}
```

Bước 17. Bổ sung phương thức sau để nhập cố định một danh sách hình học

```
// Định nghĩa hàm nhập cố định 1 danh sách hình
public void NhapCoDinh()
{
    Them(new HinhTron(123));
    Them(new HinhVuong(32.14));
    Them(new HinhVuong(65));
    Them(new HinhChuNhat(30, 14.5));
    Them(new HinhChuNhat(100.789, 23.4));
    Them(new HinhVuong(3.75));
    Them(new HinhTron(10.23));
    Them(new HinhTron(90));
    Them(new HinhChuNhat(12.5, 4.2));
    Them(new HinhTron(4.5));
}
```

Bước 18. Tiếp tục cài đặt phương thức sau để xuất danh sách hình học

```
// Định nghĩa hàm xuất thông tin các hình
public void Xuat()
{
    for (int i = 0; i < soLuong; i++)
    {
        danhSach[i].Xuat();
    }
}
```

Bước 19. Trong hàm Main, nhập đoạn mã sau để kiểm tra hoạt động của lớp vừa tạo

```
// Tạo một thể hiện kiểu DanhSachHinhHoc
DanhSachHinhHoc mang = new DanhSachHinhHoc();

// Nhập cố định 1 danh sách hình
mang.NhapCoDinh();

Console.WriteLine("So hinh hoc trong danh sach {0}",
                  mang.SoLuong);

// Xuất thông tin các hình
mang.Xuat();
```

Bước 20. Chạy chương trình và ghi nhận lại kết quả.

4. Luyện tập: Cài đặt một số hàm xử lý

Phần này hướng dẫn cài đặt một số hàm xử lý cơ bản như tìm kiếm, sắp xếp, xóa các đối tượng, …

Bước 21. Trở lại lớp DanhSachHinhHoc, bổ sung phương thức sau để lấy kiểu hình học.

```
// Định nghĩa hàm lấy loại hình học
private LoaiHinh LayKieuHinh(HinhHoc hinh)
{
    if (hinh is HinhTron)
        return LoaiHinh.HinhTron;
    else if (hinh is HinhVuong)
        return LoaiHinh.HinhVuong;
    else if (hinh is HinhChuNhat)
        return LoaiHinh.HinhChuNhat;
    else
        return LoaiHinh.TatCa;
}
```

Bước 22. Cài đặt phương thức tìm danh sách các hình theo loại như sau

```
// Định nghĩa hàm tìm tất cả các hình theo loại
public DanhSachHinhHoc TimHinhTheoLoai(LoaiHinh kieuHinh)
{
    DanhSachHinhHoc ketQua = new DanhSachHinhHoc();

    for (int i = 0; i < soLuong; i++)
    {
        if (LayKieuHinh(danhSach[i]) == kieuHinh)
            ketQua.Them(danhSach[i]);
    }

    return ketQua;
}
```

Bước 23. Trong hàm Main, nhập đoạn mã sau để kiểm tra hàm vừa định nghĩa

```
// Tìm tất cả hình chữ nhật
DanhSachHinhHoc dsHCN = mang.TimHinhTheoLoai(LoaiHinh.HinhChuNhat);
dsHCN.Xuat();
```

Bước 24. Chạy chương trình và ghi lại kết quả.

Bước 25. Trở lại lớp DanhSachHinhHoc, cài đặt thêm các phương thức sau

```
// Định nghĩa hàm tìm diện tích hình lớn nhất
public double TimDienTichLonNhat()
{
    double max = 0, dtich;

    for (int i = 0; i < soLuong; i++)
    {
        dtich = danhSach[i].TinhDienTich();
        if (dtich > max) max = dtich;
    }
    return max;
}

// Định nghĩa hàm tìm tất cả các hình có diện
// tích lớn nhất
public DanhSachHinhHoc TimHinhCoDienTichLonNhat()
{
    double dtmax = TimDienTichLonNhat();
    DanhSachHinhHoc ketQua = new DanhSachHinhHoc();

    for (int i = 0; i < soLuong; i++)
    {
        if (danhSach[i].TinhDienTich() == dtmax)
            ketQua.Them(danhSach[i]);
    }

    return ketQua;
}
```

Bước 26. Trong hàm Main, bổ sung thêm đoạn mã sau và chạy chương trình để kiểm tra

```
// Tìm các hình có diện tích lớn nhất
Console.WriteLine("================================================");
DanhSachHinhHoc hinhLon = mang.TimHinhCoDienTichLonNhat();
hinhLon.Xuat();

// Tìm các hình chữ nhật có diện tích lớn nhất
Console.WriteLine("================================================");
DanhSachHinhHoc hcnLon = dsHCN.TimHinhCoDienTichLonNhat();
hcnLon.Xuat();
```

Bước 27. Trong lớp DanhSachHinhHoc, cài đặt thêm phương thức sắp xếp như sau

```
// Định nghĩa hàm sắp xếp hình tăng theo diện tích
public void SapTangTheoDienTich()
{
    for (int i = 0; i < soLuong-1; i++)
        for (int j = i + 1; j < soLuong; j++)
            if (danhSach[i].TinhDienTich() > danhSach[j].TinhDienTich())
            {
                HinhHoc tam = danhSach[i];
                danhSach[i] = danhSach[j];
                danhSach[j] = tam;
            }
}
```

Bước 28. Tiếp tục nhập đoạn mã sau vào hàm Main

```
// Sắp tăng theo diện tích
mang.SapTangTheoDienTich();

// Xuất tất cả các hình, sắp tăng theo diện tích
Console.WriteLine("================================================");
mang.Xuat();

// Xuất danh sách hình vuông, sắp tăng theo diện tích
Console.WriteLine("================================================");
DanhSachHinhHoc dsVuong = mang.TimHinhTheoLoai(LoaiHinh.HinhVuong);
dsVuong.SapTangTheoDienTich();
dsVuong.Xuat();
```

Bước 29. Chạy chương trình và ghi nhận lại kết quả.

Bước 30. So sánh cách gọi hàm tính diện tích với cách gọi trong Lab05.

D. Bài tập bắt buộc

1. Tiếp tục với dự án Lab06_Bai02_ SuDungLopTruuTuong
   a. Hãy cài đặt các phương thức sau trong lớp DanhSachHinhHoc

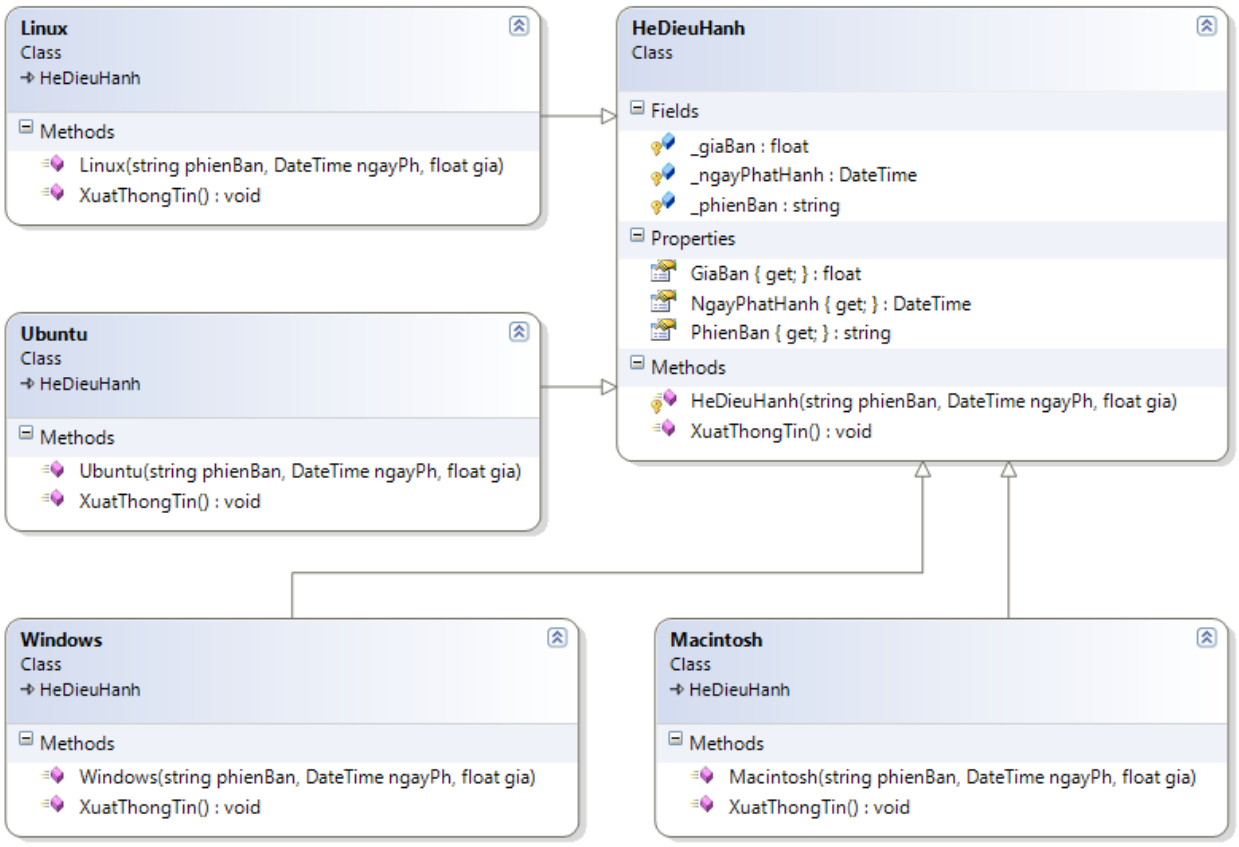
Phương thức	**Ý nghĩa**
DanhSachHinhHoc TimHinhCoDienTichNhoNhat()	Tìm hình có diện tích nhỏ nhất
DanhSachHinhHoc TimHinhTronNhoNhat()	Tìm hình tròn có diện tích lớn nhất
void SapGiamTheoDienTich()	Sắp các hình giảm dần theo diện tích
Int DemSoLuongHinh(LoaiHinh kieu)	Đếm số lượng hình theo loại
Double TinhTongDienTich()	Tính tổng diện tích các hình
DanhSachHinhHoc TimHinhCoDienTichLonNhat (LoaiHinh kieu)	Tìm hình có diện tích lớn nhất theo loại hình học cho trước

Int TimViTriCuaHinh(HinhHoc h)	Tìm vị trí của hình h trong danh sách
Bool XoaTaiViTri(int viTri)	Xóa một hình tại vị trí cho trước
DanhSachHinhHoc TimHinhTheoDTich(double dt)	Tìm hình theo diện tích
Bool XoaHinh(HinhHoc h)	Xóa một hình học khỏi danh sách
Void XoaHinhTheoLoai(LoaiHinh kieu)	Xóa tất cả các hình theo loại cho trước
DanhSachHinhHoc TimHinhCoChuViNhoNhat()	Tìm hình có chu vi nhỏ nhất
Void XuatHinhTheoChieuTangGiam(LoaiHinh kieu, bool tang)	Xuất danh sách hình theo loại cho trước và sắp tăng hoặc giảm
Double TinhTongChuVi(LoaiHinh kieu)	Tính tổng chu vi các hình theo loại
DanhSachHinhHoc TimHinhCoDienTichBangBinhPhuongChuVi()	Tìm những hình có diện tích bằng bình phương của chu vi
Void SapXep(KieuSapXep ksx)	Sắp xếp các hình học theo một tiêu chí cho trước

b. Viết lời gọi hàm trong hàm Main để kiểm tra các phương thức vừa định nghĩa

c. Tạo lớp Menu với các chức năng như bảng trên và viết mã để xử lý menu.

2. Hãy cài đặt các lớp biểu diễn các hệ điều hành theo sơ đồ dưới đây:

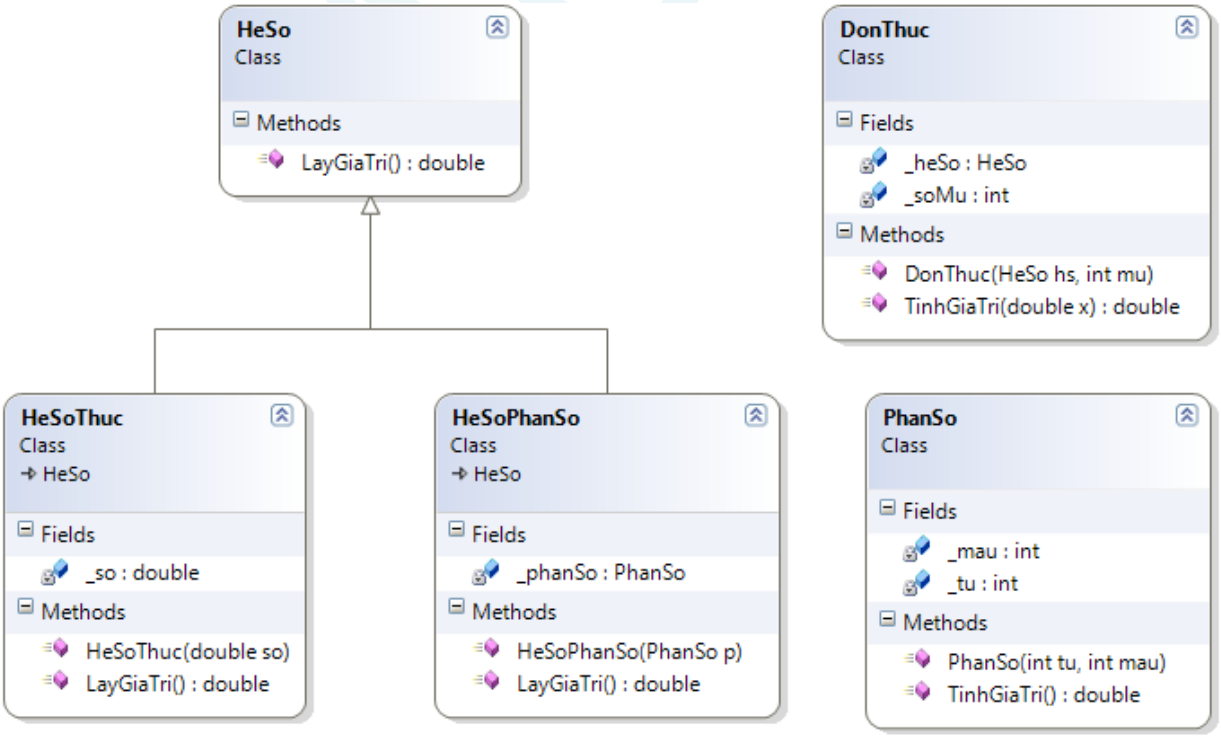


Hàm XuatThongTin( ) dùng để xuất các thông tin về hệ điều hành như tên hệ điều hành, phiên bản, ngày phát hành, công ty phát hành, …

Thực hiện tiếp các yêu cầu sau:

a. Với mỗi loại hệ điều hành, tạo 2 thể hiện (phiên bản hệ điều hành khác nhau)
b. Xuất thông tin các hệ điều hành đã tạo ở câu a.
c. Tạo thêm lớp DanhSachHDH để lưu trữ một danh sách các hệ điều hành. Trong đó, cài đặt các hàm: Thêm một hệ điều hành vào danh sách, nhập cố định một danh sách các hệ điều hành, xuất thông tin các hệ điều hành ra màn hình.
d. Sắp xếp các hệ điều hành giảm dần theo ngày phát hành.
e. Sắp xếp các hệ điều hành tăng dần theo giá.
f. Tìm những hệ điều hành Windows có giá bán lớn hơn số tiền M cho trước.
g. Tìm những hệ điều hành được phát hành trong năm 2012.
h. Tìm những hệ điều hành phiên bản Professional
i. Xuất danh sách hệ điều hành phát hành trước năm 2010 và có giá bán thấp hơn M đôla.
j. Liệt kê các hệ điều hành được phát hành theo từng năm.

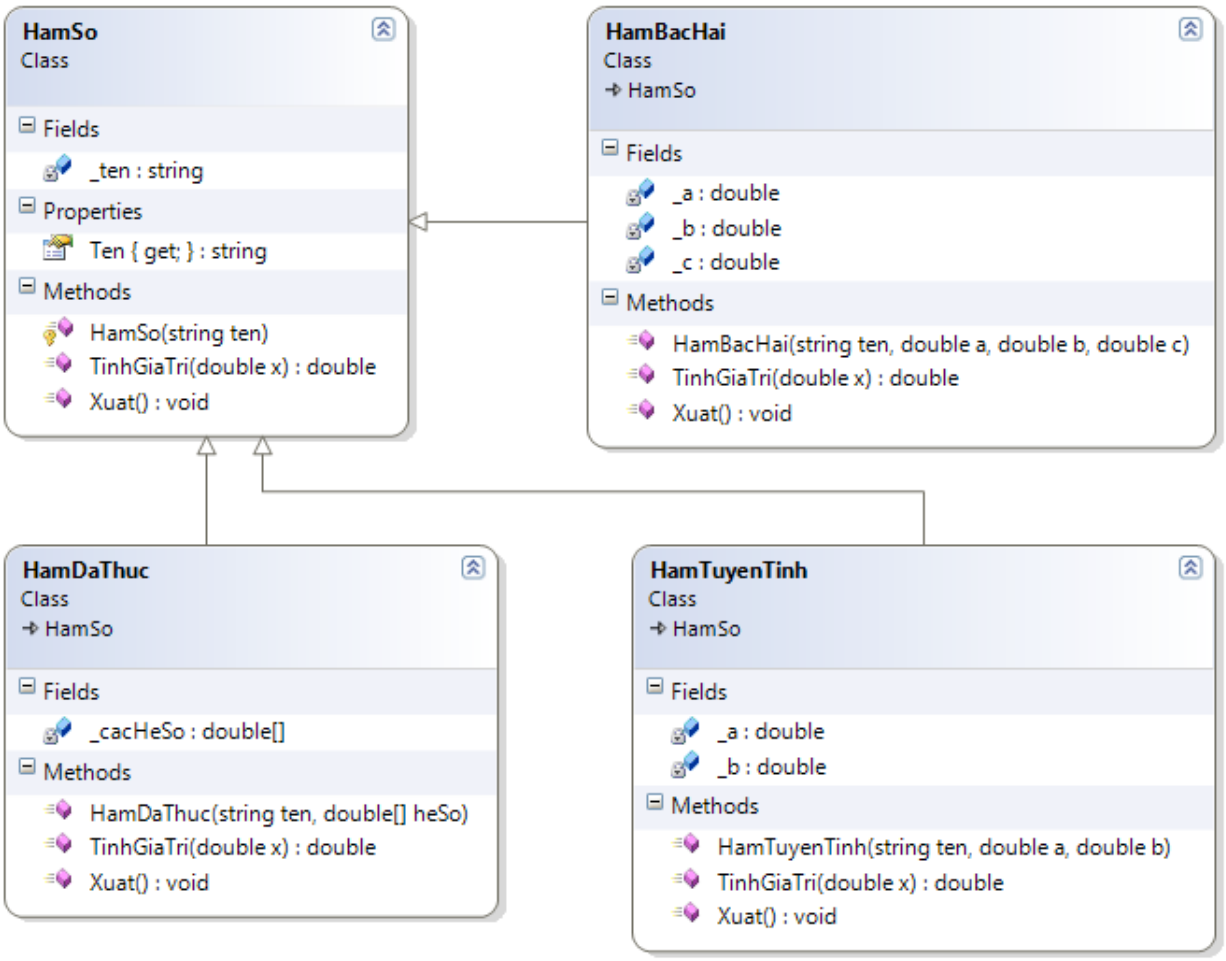
3. Biểu thức Ax^n gọi là một đơn thức. Trong đó, n được gọi là bậc của đơn thức hay số mũ, A là hệ số. Hãy cài đặt lớp biểu diễn đơn thức với hàm tính giá trị của đơn thức tại giá trị x cho trước. Biết rằng hệ số A có thể là một số nguyên hoặc một số thực hoặc là một phân số.



Trình tự cài đặt: PhanSo, HeSo, HeSoThuc, HeSoPhanSo, DonThuc. Thực hiện tiếp các yêu cầu sau (Có thể sửa đổi các lớp đã cài đặt ở trên cho phù hợp):

a. Tạo 3 thể hiện của lớp đơn thức ứng với các đơn thức sau: $3x^5$, $10.75x^2$, $(3/4)x^4$.
b. Yêu cầu người dùng nhập 1 số thực từ bàn phím (x) và xuất giá trị của 3 đơn thức ứng với x
c. Cài đặt lớp MangDonThuc để lưu trữ một danh sách các đơn thức. Trong đó có các phương thức sau: Nhập cố định một danh sách đơn thức, Thêm một đơn thức vào mảng, Xuất danh sách đơn thức ra màn hình.
d. Cài đặt hàm tính tổng giá trị tất cả các đơn thức tại giá trị x cho trước.
e. Cài đặt hàm tính giá trị trung bình cộng của tất cả các đơn thức tại giá trị x cho trước.
f. Cài đặt hàm đếm số lượng đơn thức có hệ số là phân số
g. Cài đặt hàm tìm số mũ lớn nhất trong các đơn thức.
h. Cài đặt hàm tìm đơn thức với hệ số có giá trị nhỏ nhất trong các đơn thức.
i. Viết lời gọi hàm trong hàm Main để kiểm tra các phương thức vừa định nghĩa
j. Tạo lớp Menu với các chức năng như trên và viết mã để xử lý menu.

E. Bài tập làm thêm



1. Hãy cài đặt các lớp để biểu diễn các hàm số một biến theo sơ đồ lớp trên đây. Mỗi hàm số có một tên, chẳng hạn như f(x), g(x), … Trong đó:
   - Phương thức TinhGiaTri(x) dùng để tính giá trị của hàm số tại x cho trước.
   - Phương thức Xuat( ) dùng để xuất đa thức theo các dạng sau:
   - Hàm tuyến tính có dạng: $f(x) = ax + b$ ← Chỉ cần lưu hệ số a, b
   - Hàm bậc hai có dạng: $g(x) = ax^2 + bx + c$ ← Chỉ cần lưu hệ số a, b, c
   - Hàm đa thức có dạng: $h(x) = a_n x^n + a_{n-1}x^{n-1} + \dots + a_2x^2 + a_1x^1 + a_0x^0$.

Hàm đa thức được biểu diễn bởi một mảng một chiều chứa n+1 số thực. Mỗi phần tử của mảng tương ứng với một hệ số của đa thức. Chỉ số của phần tử tương ứng với số mũ.

Chỉ số	0	1	2	3	…	…	n-1	n
Hệ số	a_0	a_1	a_2	a_3	…	…	a_{n-1}	a_n

Thực hiện tiếp các yêu cầu sau (có thể sửa đổi các lớp trên cho phù hợp nếu cần):

a. Với mỗi loại hàm số ở trên, tạo một thể hiện của lớp tương ứng. Dữ liệu nhập tùy ý.
b. Xuất các hàm số đã tạo ở câu a ra màn hình. Dùng dấu ^ để thể hiện phép lũy thừa: x^2 = x^2
c. Nhập một số thực x từ bàn phím và xuất ra giá trị của các hàm số tại x.
d. Cài đặt lớp DanhSachHamSo để lưu trữ một mảng các hàm số. Trong đó có các phương thức: Thêm một hàm số vào danh sách, nhập cố định một danh sách các hàm số tùy ý, xuất các hàm số ra màn hình.
e. Đếm số lượng hàm số mỗi loại
f. Xuất danh sách các hàm số theo thứ tự: hàm tuyến tính, hàm bậc hai, hàm đa thức.
g. Nhập một giá trị x và xuất ra các hàm số kèm theo giá trị của hàm số tại giá trị x đó.
h. Như câu g nhưng xuất các hàm số theo tứ tự giảm dần của giá trị hàm số tại x.
i. Tìm bậc của các hàm đa thức. Bậc chính là số mũ lớn nhất.
j. Tìm và xuất các hàm đa thức có bậc lớn nhất.

2. Cài đặt các lớp biểu diễn các đối tượng hình học 3 chiều như hình hộp chữ nhật, hình trụ tròn, hình nón, hình cầu, … Các lớp này đều kế thừa từ lớp HinhHoc3D (hình học 3 chiều). Lớp HinhHoc3D có một phương thức để tính thể tích của hình. Sau đó cài đặt tiếp lớp DanhSachHinh3D để lưu trữ một danh sách các đối tượng hình học 3 chiều. Thực hiện tiếp các yêu cầu sau:
   a. Với mỗi loại đối tượng hình học 3D, tạo một thể hiện và xuất thông tin về nó ra màn hình.
   b. Tạo hay nhập cố định một danh sách các đối tượng hình học 3D
   c. Xuất danh sách hình học giảm dần theo thể tích.
   d. Tìm hình theo loại
   e. Tìm hình cầu có thể tích bé nhất
   f. Tìm hình có thể tích lớn nhất nhưng không phải là hình nón hay hình cầu.