LAB 5. KẾ THỪA (KHÔNG DÙNG TÍNH ĐA HÌNH)

THỜI LƯỢNG : 4 TIẾT

A. Mục tiêu

- Giúp sinh viên biết cách tạo ra các lớp phân cấp kế thừa và hiểu hơn về phép ép kiểu cũng như các mức truy xuất.
- Sau khi hoàn thành bài thực hành này, sinh viên cần:
  - Biết cách tạo ra các lớp có kế thừa từ một lớp khác
  - Hiểu rõ và nắm vững khái niệm kế thừa, không dùng tính đa hình
  - Hiểu rõ về phép ép kiểu và các mức truy xuất (public, internal, protected, private)
  - 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
  - Biết cách sử dụng và hiểu rõ tác dụng của từ khóa base

B. Yêu cầu

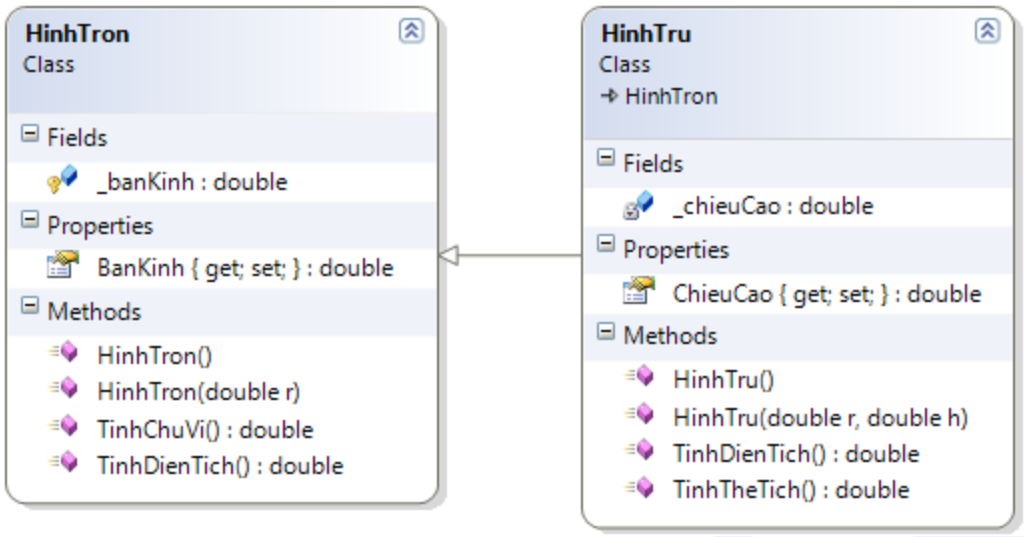
- Sinh viên phải trả lời đầy đủ các câu hỏi (nếu có) hoặc chụp màn hình kết quả, ghi lại vào tập tin Word theo yêu cầu ghi trong phần hướng dẫn thực hành.
- Đặt tên tập tin Word theo dạng: Lab05_MSSV_HoVaTen.rar.
- Tạo thư mục, đặt tên là MSSV_Lab05 để lưu tất cả bài làm trong bài thực hành này.
- Sinh 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

1. Định nghĩa các lớp có sử dụng tính kế thừa

Phần này hướng dẫn cách định nghĩa hai lớp HinhTron và HinhTru. Trong đó, lớp HinhTru kế thừa từ lớp HinhTron như được thể hiện trong hình bên dưới. Bài tập cho thấy cách lớp HinhTru gọi phương thức khởi tạo của lớp HinhTron & thừa hưởng các biến cũng như thuộc tính của HinhTron

Bước 1. Tạo một thư mục, đặt tên theo dạng MSSV-Lab05. Ví dụ: 1210234-Lab05.
Bước 2. Tạo dự án dạng Console Application, đặt tên là Lab05_Bai01_KeThuaVaEpKieu.
Bước 3. Tạo hai lớp mới, đặt tên lần lượt là HinhTron và HinhTru.



Bước 4. Cài đặt lớp HinhTron như sau:

```
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 double TinhChuVi()
    {
        return 2 * Math.PI * _banKinh;
    }

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

Bước 5. Cài đặt lớp HinhTru như trong đoạn mã sau:

```
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 new 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;
    }
}
```

Bước 6. Tiếp theo, tạo hai thể hiện của hai lớp trên trong hàm Main

```
// Tạo một thể hiện của lớp HinhTron
HinhTron tron = new HinhTron(10);

// Tạo một thể hiện của lớp HinhTru
HinhTru tru = new HinhTru(10, 30);
```

Bước 7. Gọi hàm tính chu vi từ hai đối tượng tron, tru như sau

```
// Gọi hàm xuất chu vi của hai hình
Console.WriteLine(tron.TinhChuVi());
Console.WriteLine(tru.TinhChuVi());
```

Bước 8. Chạy chương trình và ghi nhận kết quả.

Bước 9. Giá trị xuất ra màn hình trên hai dòng có giống nhau không? Giải thích tại sao?

Bước 10. Tại sao trong lớp HinhTru, ta không hề định nghĩa phương thức TinhChuVi nhưng ở đây vẫn gọi được hàm đó (tru.TinhChuVi)?

Bước 11. Tiếp tục bổ sung đoạn mã sau trong hàm Main để gọi hàm tính diện tích

```
// Gọi hàm tính diện tích hình tròn
Console.WriteLine(tron.TinhDienTich());

// Gọi hàm tính diện tích hình trụ
Console.WriteLine(tru.TinhDienTich());
```

Bước 12. Chạy chương trình và cho biết kết quả có giống nhau hay không? Tại sao?

Bước 13. Hãy cho biết ý nghĩa của lệnh base.TinhDienTich trong phương thức TinhDienTich của lớp HinhTru. Nếu bỏ từ khóa base hoặc thay bằng this, điều gì sẽ xảy ra?

Bước 14. Có thể sử dụng lệnh Console.WriteLine(tron._banKinh) hay Console.WriteLine(tru._chieuCao) hay không? Tại sao?

Bước 15. Trong hàm Main, nhập thêm đoạn mã sau để tính thể tích của hình trụ

```
// Gọi hàm tính thể tích hình trụ
Console.WriteLine(tru.TinhTheTich());
```

Bước 16. Chạy chương trình và ghi nhận kết quả.

Bước 17. Có thể gọi phương thức TinhTheTich từ đối tượng tron được hay không? Tại sao?

Bước 18. Tại sao trong phương thức TinhThetich của lớp HinhTru lại sử dụng lệnh base.TinhDienTich mà không phải là this.TinhDienTich?

2. Ép kiểu lên (UpCast) và ép kiểu xuống (DownCast)

Ép kiểu lên: Lớp con (HinhTru) thừa hưởng mọi thuộc tính, biến thành viên và các phương thức từ lớp lớp cha (HinhTron). Vì nó có đầy đủ các đặc trưng của lớp cha nên ta có thể gán một thể hiện của lớp con cho một biến của lớp cha. Phép ép kiểu này được hiểu ngầm (implicit).

Bước 19. Bổ sung thêm đoạn mã sau trong hàm Main

```
HinhTru con = new HinhTru(5, 20);
HinhTron epLen = con;

Console.WriteLine("Con : {0}", con.TinhDienTich());
Console.WriteLine("Cha : {0}", epLen.TinhDienTich());
```

Bước 20. Chạy chương trình và cho biết kết quả.

Bước 21. Giải thích tại sao cả hai biến con và epLen đều tham chiếu tới cùng một đối tượng hình trụ nhưng kết quả xuất ra lại khác nhau?

Bước 22. Tại sao biến epLen đang tham chiếu tới một đối tượng HinhTru nhưng ta lại không thể gọi phương thức TinhTheTich từ nó? Nghĩa là không thể dùng: epLen.TinhTheTich().

Ép kiểu xuống: Ngược lại, khi một biến của lớp cha đang tham chiếu đến (hay lưu trữ) một thể hiện của lớp con (như ví dụ trên), ta có thể ép nó ngược trở về kiểu của lớp con ban đầu. Lưu ý rằng, đối với ép kiểu xuống, ta phải viết tường minh (có thêm kiểu dữ liệu vào trước tên biến).

Bước 23. Trong hàm Main, bổ sung đoạn mã sau:

```
HinhTru epXuong = (HinhTru) epLen;
Console.WriteLine(epXuong.TinhDienTich());
```

Bước 24. Chạy chương trình và ghi nhận kết quả. Cho biết kết quả có đúng hay không?

Lưu ý quan trọng: Việc ép kiểu xuống chỉ thực hiện được khi một biến lớp cha đang tham chiếu đến một thể hiện của lớp con. Ta không thể ép một thể hiện của lớp cha thành kiểu lớp con.

Ví dụ: Ta có thể đặt hình trụ thẳng trước mặt và nhìn vào mặt đáy của nó. Khi đó, ta chỉ nhìn thấy hình tròn. Tuy nhiên, không có cách nào biến một hình tròn thành hình trụ.

Bước 25. Trong hàm Main, bổ sung đoạn mã sau:

```
HinhTron cha = new HinhTron(7);
HinhTru error = cha;
```

Bước 26. Đường răng cưa màu đỏ cho biết có lỗi xảy ra. Ghi nhận và giải thích thông báo lỗi đó

Bước 27. Sửa đổi đoạn mã ở bước 25 thành đoạn mã sau:

```
HinhTron cha = new HinhTron(7);
HinhTru error = (HinhTru) cha;
```

Bước 28. Chạy chương trình và cho biết có lỗi xảy ra hay không? Nếu có, ghi nhận và giải thích thông báo lỗi đó.

3. Từ khóa is và as

Để hạn chế lỗi xảy ra khi ép kiểu, ta dùng từ khóa **is** và **as**. Từ khóa is dùng để kiểm tra một biến có kiểu cho trước hay không. Từ khóa as dùng để ép một đối tượng về một kiểu cho trước. Nếu phép ép kiểu không thành công, kết quả trả về là null.

Bước 29. Xóa bỏ dòng lệnh: `HinhTru error = (HinhTru) cha;`. Tiếp tục nhập đoạn mã sau:

```
// Kiểm tra kiểu bằng từ khóa is
if (epLen is HinhTru)
{
    HinhTru ong = (HinhTru) epLen;
    Console.WriteLine(ong.TinhTheTich());
}
```

Bước 30. Chạy chương trình và cho biết có lỗi xảy ra không? Tại sao?

Bước 31. Bổ sung đoạn mã sau vào hàm Main

```
// Ép kiểu dùng từ khóa as
HinhTru que = epLen as HinhTru;
Console.WriteLine(que.TinhTheTich());

// Lệnh sau vẫn được thực thi nhưng biến
// rong sẽ mang giá trị null vì không thể
// ép kiểu từ HinhTron sang HinhTru
HinhTru rong = cha as HinhTru;

if (rong == null)
    Console.WriteLine("Khong the ep hinh tron -> hinh tru");
```

Bước 32. Chạy chương trình và cho biết có lỗi xảy ra không? Tại sao? Kết quả xuất ra là gì?

4. Tạo cây phân cấp kế thừa

Một lớp B chỉ có thể kế thừa (hay dẫn xuất) từ một lớp A nào đó. Tuy nhiên, nó cũng có thể được kế thừa bởi nhiều lớp (C, D, E, … kế thừa từ B) khác. Tổng quát, chúng sẽ hình thành nên một cây phân cấp kế thừa.

Trong phần này, ta sẽ xây dựng một chương trình đơn giản để quản lý các đối tượng hình học như hình tròn, hình vuông, hình chữ nhật. Các đối tượng này đều có chung phương thức tính diện tích.

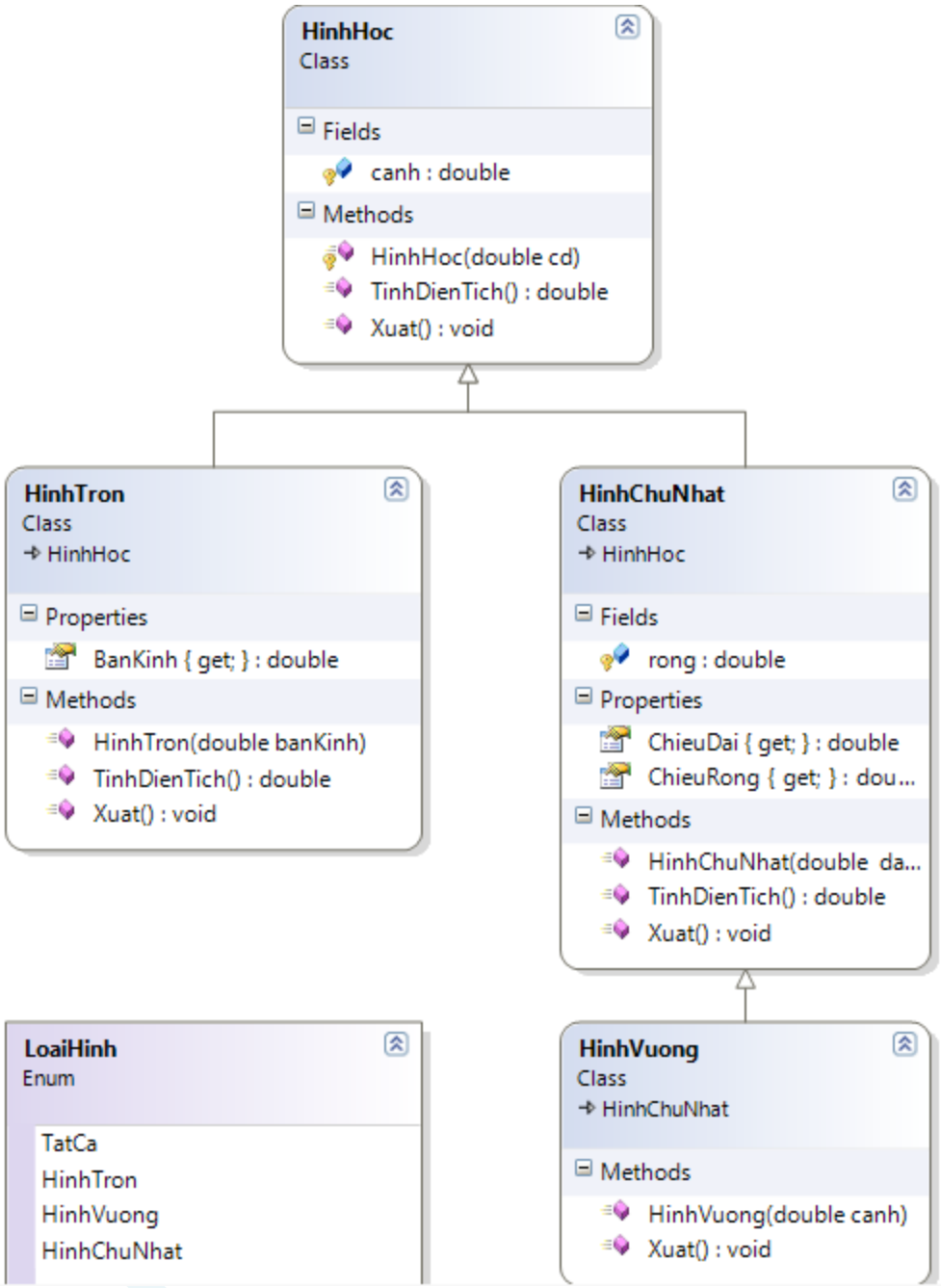
Bước 1. Tạo dự án dạng Console Application, đặt tên là Lab05_Bai02_KeThuaKhongDaHinh.

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:




```
// Khai báo biến thành viên
protected double canh;

// Định nghĩa phương thức khởi tạo (PTKT). Ở đây ta dùng
// protected để các lớp con có thể sử dụng PTKT này nhưng
// các lớp khác không thể tạo thể hiện của lớp HinhHoc
protected HinhHoc(double cd)
{
    canh = cd;
}
```

```
// Định nghĩa hàm tính diện tích
public double TinhDienTich()
{
    return 0;   // Tạm thời trả về 0
}

// Định nghĩa hàm xuất thông tin
public void Xuat()
{
    Console.WriteLine("Hinh hoc tong quat");
}
```

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

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

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:

```
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 new double TinhDienTich()
    {
        return canh * canh * Math.PI;
    }

    // Định nghĩa hàm xuất thông tin hình tròn
    public new 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);
```

```
// Gọi hàm xuất thông tin hình
vong.Xuat();
tron.Xuat();

// Gọi hàm xuất diện tích của hình
Console.WriteLine(vong.TinhDienTich());
Console.WriteLine(tron.TinhDienTich());
```

Bước 9. Chạy chương trình và ghi nhận kết quả. Tại sao cả hai biến vong và tron đều tham chiếu đến thể hiện của lớp hình tròn với bán kính là 5 nhưng kết quả lại khác nhau?

Bước 10. 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 new double TinhDienTich()
    {
        return canh * rong;
    }

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

Bước 12. Trong hàm Main, xóa bỏ đoạn mã hiện có, 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 kết quả. Kết quả có như mong muốn không? Tại sao?

Bước 14. Thay cách lệnh xuất ở bước 12 bởi đoạn mã sau:

```
((HinhTron) tron).Xuat();
((HinhVuong) vuong).Xuat();
((HinhChuNhat) cnhat).Xuat();
```

Bước 15. Chạy chương trình và cho biết kết quả. Kết quả xuất đúng thông tin các hình chưa?

Bước 16. Tại sao phải sử dụng phép ép kiểu trong đoạn mã ở bước 14.

Bước 17. Hãy viết mã lệnh để xuất diện tích của 3 hình trên và ghi lại kết quả.

Bước 18. Tại sao trong lớp HinhHoc, phải đặt mức truy xuất cho biến cạnh là protected mà không phải là public, internal hay private?

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

Phần này hướng dẫn cách xây dựng một lớp danh sách (hay mảng) để chứa các đối tượng hình học.

Tất cả các lớp kiểu danh sách dạng này đều yêu cầu một mảng (tên là danhSach) để lưu trữ các đối tượng, một biến (tên là soLuong) để lưu số đối tượng có trong danh sách, các phương thức xử lý như thêm một đối tượng hay nhập một dãy các đối tượng vào danh sách, xuất các đối tượng, xóa đối tượng tại vị trí VT, tìm vị trí đối tượng và sắp xếp các đối tượng theo tiêu chí nào đó.

Bước 19. 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 20. 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 21. 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++)
    {
        if (danhSach[i] is HinhTron)
            ((HinhTron)danhSach[i]).Xuat();
        else if (danhSach[i] is HinhVuong)
            ((HinhVuong)danhSach[i]).Xuat();
        else if (danhSach[i] is HinhChuNhat)
            ((HinhChuNhat)danhSach[i]).Xuat();
    }
}
```

Bước 22. 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 23. Chạy chương trình và ghi nhận lại kết quả.

6. 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 24. 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 25. Tiếp tục định nghĩa phương thức sau để lấy diện tích của hình

```
// Định nghĩa hàm lấy diện tích hình
private double LayDienTich(HinhHoc hinh)
{
    double dienTich = 0;

    if (hinh is HinhTron)
        dienTich = ((HinhTron)hinh).TinhDienTich();
    else if (hinh is HinhVuong)
        dienTich = ((HinhVuong)hinh).TinhDienTich();
    else if (hinh is HinhChuNhat)
        dienTich = ((HinhChuNhat)hinh).TinhDienTich();

    return dienTich;
}
```

Bước 26. 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 27. 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 28. Chạy chương trình và ghi lại kết quả.

Bước 29. 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 = LayDienTich(danhSach[i]);
        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 (LayDienTich(danhSach[i]) == dtmax)
            ketQua.Them(danhSach[i]);
    }
    return ketQua;
}
```

Bước 30. 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 31. Trong lớp DanhSachHinhHoc, cài đặt thêm phương thức sắp xếp như sau

```
public void SapTangTheoDienTich()
{
    for (int i = 0; i < soLuong-1; i++)
        for (int j = i + 1; j < soLuong; j++)
            if (LayDienTich(danhSach[i]) > LayDienTich(danhSach[j]))
            {
                HinhHoc tam = danhSach[i];
                danhSach[i] = danhSach[j];
                danhSach[j] = tam;
            }
}
```

Bước 32. 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 33. Chạy chương trình và ghi nhận lại kết quả.

D. Bài tập bắt buộc

1. Tiếp tục với dự án Lab05_Bai02_KeThuaKhongDaHinh trên
   a. Cài đặt enum KieuSapXep. Trong đó có các kiểu sắp xếp như: Tăng theo diện tích, giảm theo diện tích, sắp hình vuông tăng theo diện tích, sắp hình tròn giảm theo bán kính, …
   b. Cài đặt hàm void HoanVi(ref HinhHoc x, ref HinhHoc y) để hoán vị hai đối tượng hình học
   c. Cài đặt hàm int SoSanh(HinhHoc x, HinhHoc y, KieuSapXep ksx) để so sánh hai đối tượng hình học tùy theo kiểu sắp xếp cho trước. Hàm trả về 0 nếu hai đối tượng x, y giống nhau theo điều kiện sắp xếp. Trả về -1 nếu x nhỏ hơn (hoặc đứng trước) y theo điều kiện sắp xếp. Trả về 1 nếu ngược lại.
   d. Cài đặt hàm sắp xếp tổng quát: void SapXep(KieuSapXep ksx). Trong đó, sử dụng hàm HoanVi để hoán vị hai đối tượng hình học, sử dụng hàm SoSanh để làm điều kiện xác định việc hoán vị.

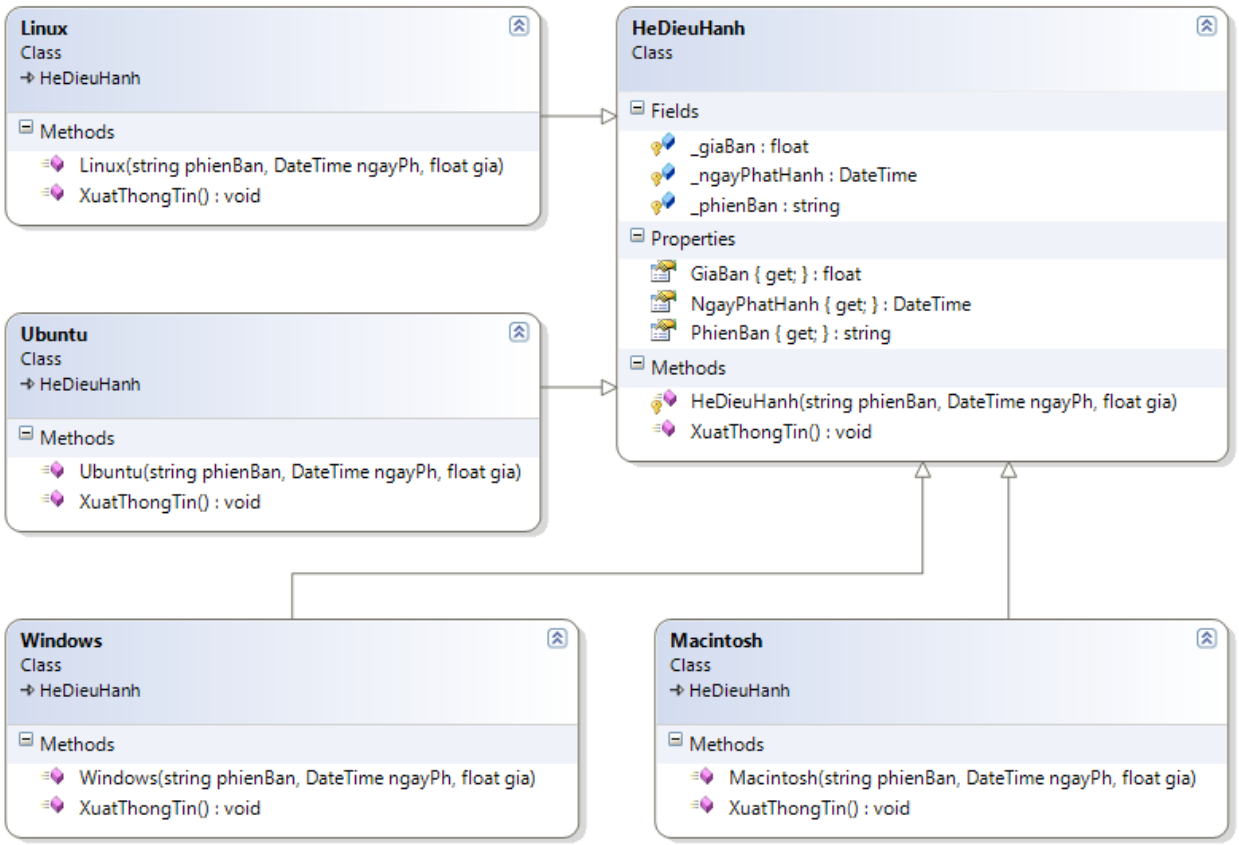
2. Tiếp tục với dự án Lab05_Bai02_KeThuaKhongDaHinh
   a. Hãy cài đặt các phương thức sau trong lớp DanhSachHinhHoc

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	Tìm hình có diện tích lớn nhất theo

(LoaiHinh kieu)	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

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.

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

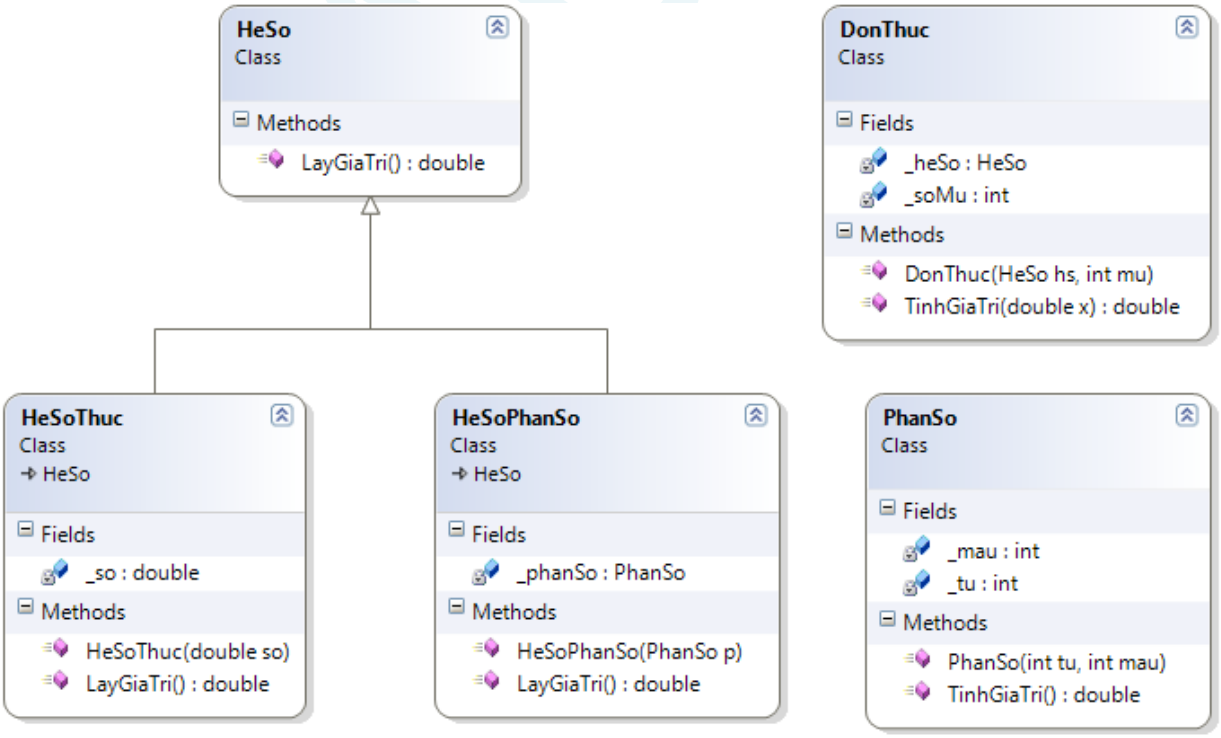


Hàm XuatThongTin( ) dùng để xuất các thông tin về hệ điều hành như tên hệ điều hành, phiên bản, ngày phát hành, công ty phát hành, …

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.

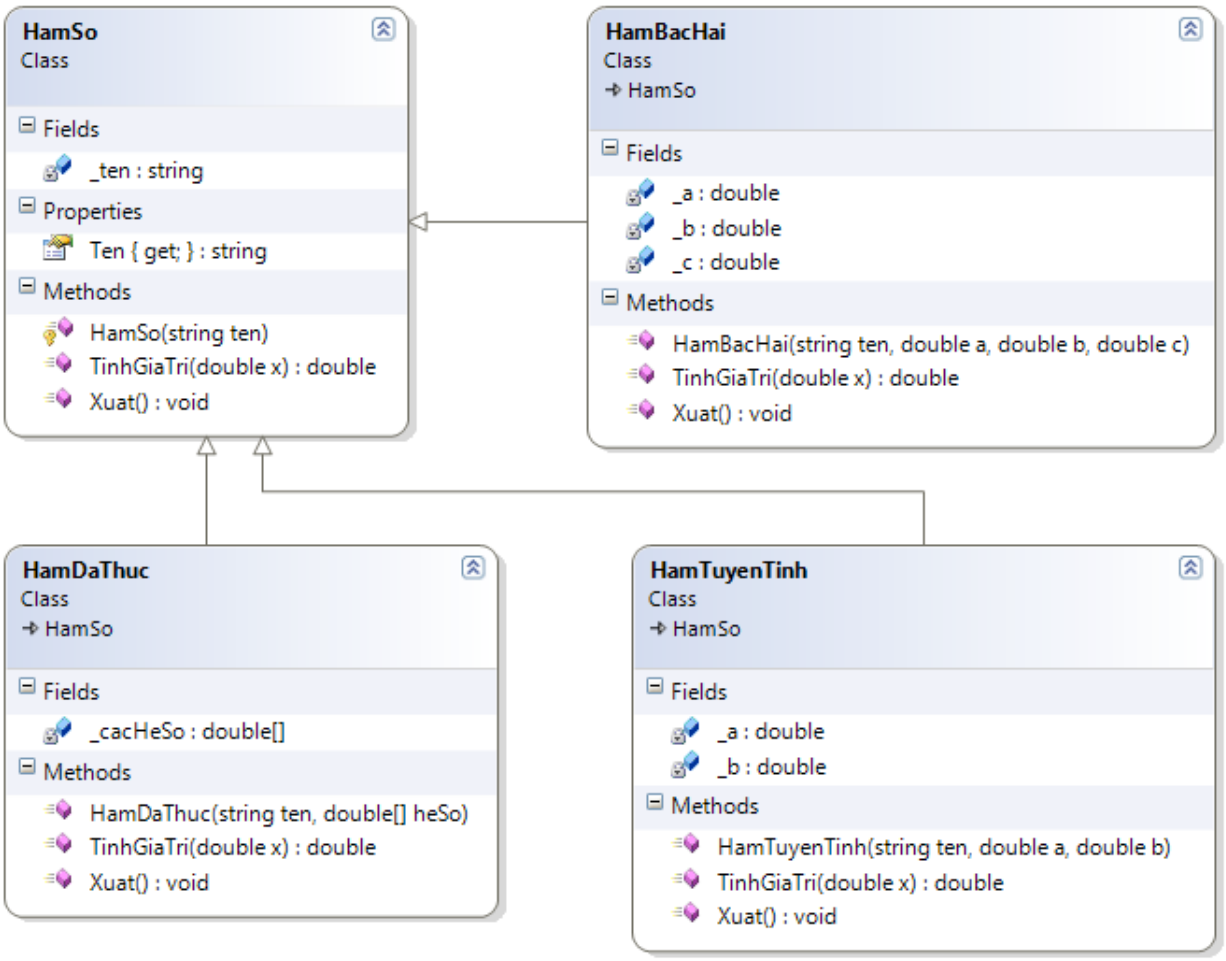
4. Biểu thức Ax^n gọi là một đơn thức. Trong đó, n được gọi là bậc của đơn thức hay số mũ, A là hệ số. Hãy cài đặt lớp biểu diễn đơn thức với hàm tính giá trị của đơn thức tại giá trị x cho trước. Biết rằng hệ số A có thể là một số nguyên hoặc một số thực hoặc là một phân số.



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.