LAB 3. LỚP VÀ ĐỐI TƯỢNG

THỜI LƯỢNG : 4 TIẾT

A. Mục tiêu

- Giúp sinh viên biết cách tạo ra các kiểu dữ liệu mới bằng cách định nghĩa các lớp.
- Hướng dẫn cách tạo ra các thể hiện (hay đối tượng) của một lớp và sử dụng các thành viên.
- Sau khi hoàn thành bài thực hành này, sinh viên cần:
  - Hiểu và giải thích được các thành phần của sơ đồ lớp.
  - Nắm vững cách định nghĩa lớp, thuộc tính, phương thức
  - Biết cách định nghĩa phương thức khởi tạo
  - Hiểu rõ về mức truy xuất: public, internal, protected, private.
  - Phân biệt lớp và đối tượng.
  - Biết cách nạp chồng phương thức.

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: Lab03_MSSV_HoVaTen.docx.
- Tạo thư mục, đặt tên là MSSV_Lab03 để lưu tất cả bài làm trong bài thực hành này.
- Sinh viên phải hoàn thành tối thiểu 4 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

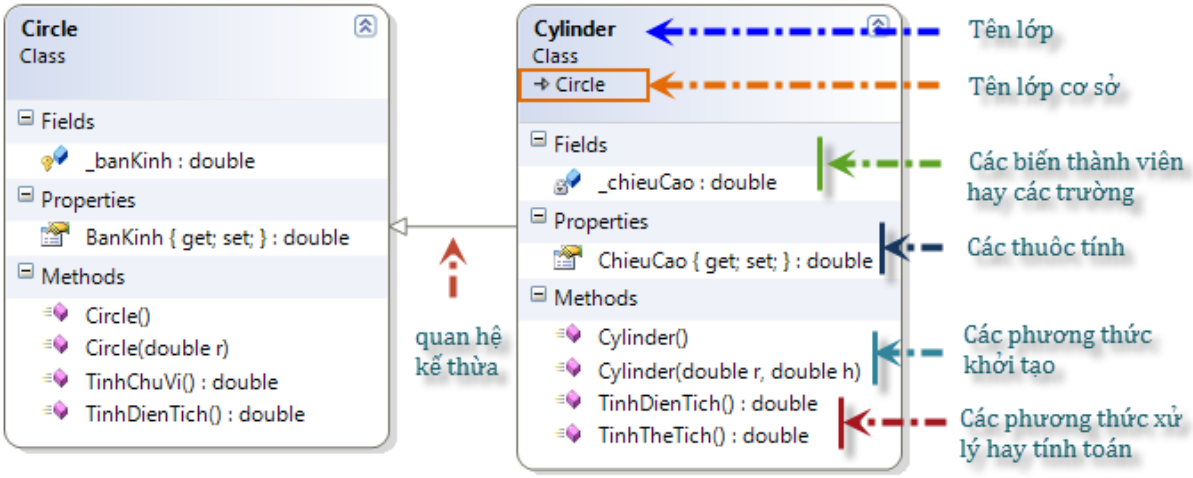
C. Hướng dẫn thực hành

1. Đọc sơ đồ lớp

Phần này hướng dẫn cách đọc sơ đồ lớp và các thành phần trong một sơ đồ lớp. Quan sát hình dưới đây, ta thấy:

- Có hai lớp là Circle (hình tròn) và Cylinder (hình trụ tròn).
- Lớp Cylinder kế thừa (được dẫn xuất) từ lớp Circle.
- Lớp Circle gọi là lớp cơ sở của lớp Cylinder.
- Lớp Cylinder gọi là lớp dẫn xuất từ lớp Circle.

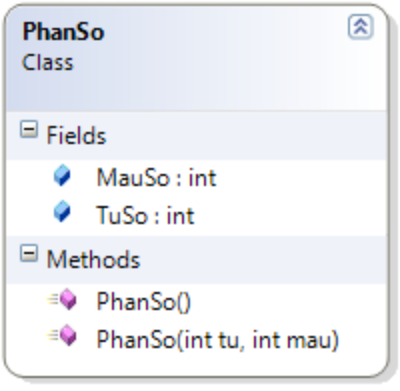
- Trong mỗi lớp, mục Fields cho biết các biến thành viên của lớp.
- Mục Properties cho biết các thuộc tính của mỗi lớp.
- Mục Methods cho biết các phương thức (hàm) được định nghĩa trong lớp.
- Những phương thức trùng tên với tên lớp gọi là phương thức khởi tạo
- Ký hiệu ở đầu mỗi dòng cho biết mức truy xuất của các thành viên.
- Từ khóa get, set là các bộ truy xuất dùng khi định nghĩa thuộc tính.



Trong ví dụ này, lớp Circle có một biến thành viên tên _banKinh, có kiểu dữ liệu là double và mức truy xuất là protected. Lớp Cylinder có một biến thành viên tên _chieuCao, có kiểu dữ liệu là double và mức truy xuất là private.

Tương tự, mỗi lớp có một thuộc tính. Tất các các phương thức trong hai lớp đều có phạm vi truy xuất là public. Phần nằm trong dấu ngoặc sau tên phương thức gọi là tham số. Phần nằm sau dấu ngoặc và dấu ":" là kiểu dữ liệu trả về của phương thức.

2. Định nghĩa một lớp đơn giản



Phần này hướng dẫn cách định nghĩa một lớp đơn giản có tên là PhanSo (phân số) như hình bên. Lớp này có hai biến thành viên public là TuSo và MauSo có kiểu số nguyên. Nó có hai phương thức khởi tạo, một có tham số và một không có tham số.

Bước 1. Tạo một thư mục, đặt tên theo dạng MSSV-Lab03. Ví dụ: 1210234-Lab03.

Bước 2. Tạo một dự án kiểu Console Application mới, đặt tên là Lab03_Bai01_LopDonGian và lưu vào thư mục trên.

Bước 3. Tạo một lớp mới bằng cách nhấp phải chuột vào dự án, chọn Add > Class … Đặt tên lớp là PhanSo. Cài đặt lớp phân số như đoạn mã sau:

```
// Định nghĩa lớp phân số đơn giản
class PhanSo
{
    // Khai báo các biến thành viên (hay trường)
    public int TuSo;
    public int MauSo;

    // Định nghĩa phương thức khởi tạo không có tham số
    public PhanSo() {}

    // Định nghĩa phương thức khởi tạo có hai tham số
    public PhanSo(int tu, int mau)
    {
        TuSo = tu;
        MauSo = mau;
    }
}
```

Trong phần tiếp theo, ta sẽ tạo các thể hiện (đối tượng) của lớp PhanSo.

3. Tạo đối tượng

Bước 4. Trong hàm Main ở lớp Program, nhập đoạn mã sau:

```
// Tạo thể hiện (hay đối tượng) của lớp PhanSo
// Cách 1. Sử dụng phương thức khởi tạo không có tham số
PhanSo x = new PhanSo();

// Gán giá trị cho tử và mẫu của phân số x
x.TuSo = 1;
x.MauSo = 2;

// Xuất phân số x ra màn hình
Console.WriteLine("{0}/{1}", x.TuSo, x.MauSo);

// Cách 2. Sử dụng phương thức khởi tạo có tham số
PhanSo y = new PhanSo(3, 4);

// Xuất phân số y ra màn hình
Console.WriteLine("{0}/{1}", y.TuSo, y.MauSo);
```

Bước 5. Chạy chương trình và ghi nhận kết quả. Tử số và mẫu số của phân số y có giá trị là bao nhiêu? Tại sao có giá trị đó?

Bước 6. Tiếp tục tạo thêm 5 phân số khác và xuất chúng ra màn hình.

4. Định nghĩa phương thức

Bước 7. Trong lớp PhanSo, tiếp tục cài đặt phương thức sau:

```
// Định nghĩa phương thức tính giá trị của phân số
public double TinhGiaTri()
{
    return (double) TuSo / MauSo;
}
```

Bước 8. Trong hàm Main, thay đổi **hai lệnh xuất** phân số x và y ra màn hình ở bước 4 bởi các lệnh sau:

```
// Tính giá trị phân số x, lưu vào biến gtx
double gtx = x.TinhGiaTri();

// Xuất phân số x ra màn hình
Console.WriteLine("{0}/{1} = {2}", x.TuSo, x.MauSo, gtx);


// Tính giá trị phân số y, lưu vào biến gty
double gty = y.TinhGiaTri();

// Xuất phân số y ra màn hình
Console.WriteLine("{0}/{1} = {2}", y.TuSo, y.MauSo, gty);
```

Bước 9. Chạy chương trình và ghi nhận kết quả. Có gì khác so với kết quả ở bước 5.

Bước 10. Tiếp theo, ta sẽ định nghĩa phương thức để tìm phân số nghịch đảo. Bổ sung phương thức sau vào lớp PhanSo

```
// Định nghĩa phương thức tìm phần số nghịch đảo
public PhanSo NghichDao()
{
    return new PhanSo(MauSo, TuSo);
}
```

Bước 11. Trong hàm Main, bổ sung đoạn mã dưới đây vào sau lệnh xuất phân số y ở bước 8

```
// Gọi hàm tìm phân số nghịch đảo của phân số y
PhanSo z = y.NghichDao();

// Gọi hàm tính giá trị phân số z
double gtz = y.TinhGiaTri();

// Xuất phân số z ra màn hình
Console.WriteLine("{0}/{1} = {2}", z.TuSo, z.MauSo, gtz);
```

Bước 12. Chạy chương trình và ghi nhận kết quả.

5. Định nghĩa phương thức tĩnh

Các phương thức tìm nghịch đảo và tính giá trị của một phân số ở trên đều được gọi thông qua một đối tượng (hay thể hiện của một lớp). Chúng tác động lên hoặc sử dụng giá trị nội tại (hay trạng thái) của chính đối tượng đó. Tuy nhiên, trên thực tế, có những phương thức không phụ thuộc vào trạng thái của đối tượng. Vì vậy, chúng có thể được định nghĩa là phương thức tĩnh. Các phương thức tĩnh được gọi thông qua tên lớp thay vì đối tượng.

Trong phần tiếp theo, ta sẽ tạo phương thức rút gọn một phân số. Phương thức này cần gọi một phương thức khác để tìm ước chung lớn nhất của hai số nguyên. Ta đặt tên phương thức này là UCLN và được định nghĩa là phương thức tĩnh trong lớp ToanHoc (toán học).

Bước 13. Tiếp tục dự án trên bằng cách tạo một lớp mới, đặt tên là ToanHoc.

Bước 14. Trong lớp ToanHoc, định nghĩa phương thức sau:

```
// Định nghĩa phương thức tìm ước chung
// lớn nhất của hai số nguyên.
public static int UCLN(int x, int y)
{
    // Chuyển sang số dương nếu đang âm
    if (x < 0) x = -x;
    if (y < 0) y = -y;

    // Nếu 1 trong 2 số bằng 0 thì UCLN = 1
    if (x == 0 || y == 0) return 1;

    while (x != y)
    {
        if (x > y) x -= y;
        else y -= x;
    }

    return x;
}
```

Bước 15. Trở lại lớp PhanSo, định nghĩa thêm phương thức rút gọn phân số như sau:

```
// Định nghĩa phương thức rút gọn phân số
public PhanSo RutGon()
{
    // Gọi phương thức tìm ước chung lớn nhất
    int uc = ToanHoc.UCLN(TuSo, MauSo);
    return new PhanSo(TuSo / uc, MauSo / uc);
}
```

Bước 16. Trong hàm Main, bổ sung đoạn mã sau để gọi hàm rút gọn phân số:

```
// Tạo một phân số khác
PhanSo k = new PhanSo(6, 9);

// Gọi hàm rút gọn phân số
PhanSo g = k.RutGon();

// Xuất cả hai phân số và giá trị
Console.WriteLine("{0}/{1} = {2}/{3} = {4}",
                  k.TuSo, k.MauSo,
                  g.TuSo, g.MauSo,
                  g.TinhGiaTri());
```

Bước 17. Chạy chương trình và cho biết kết quả.

6. Phương thức ToString()

Phương thức ToString() được dùng để tạo ra một chuỗi biểu diễn một đối tượng. Nó được định nghĩa sẵn trong lớp System.Object. Vì mọi lớp đều kế thừa từ lớp Object nên chúng đều thừa hưởng phương thức ToString() này. Tuy nhiên, mặc định phương thức này xuất ra tên của namespace và tên lớp. Vì vậy, ta cần phải định nghĩa lại phương thức ToString() để xuất ra đúng nội dung mong muốn.

Bước 18. Tiếp tục dự án trên. Phương thức ToString của lớp PhanSo được định nghĩa như sau

```
// Định nghĩa phương thức tạo ra một chuỗi
// mô tả trạng thái (hay nội dung) của đối tượng
public override string ToString()
{
    return string.Format("{0}/{1}", TuSo, MauSo);
}
```

Bước 19. Phương thức trên chỉ đơn giản là tạo ra một chuỗi có dạng: tu/mau. Trong đó tu và mau là tử số và mẫu số của một phân số. Phương thức ToString có thể được gọi tường minh hoặc ngầm định. Trong hàm Main, bổ sung đoạn mã sau:

```
// Gọi phương thức ToString tường minh
Console.WriteLine(x.ToString());
Console.WriteLine("{0} = {1}", y.ToString(), y.TinhGiaTri());

// Gọi phương thức ToString ngầm
Console.WriteLine(z);
Console.WriteLine("{0} = {1} = {2}",
                  k, g, g.TinhGiaTri());
```

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

7. Định nghĩa thuộc tính

Trong phần này, ta sẽ định nghĩa các thuộc tính cho lớp PhanSo. Tuy nhiên, trước tiên, ta sẽ xem xét một vấn đề xảy ra với lớp PhanSo hiện tại.

Bước 21. Tiếp tục với dự án trên. Che giấu tất cả các dòng lệnh trong hàm Main bằng chức năng Edit > Edvance > Comment Selection (hoặc nhấn tổ hợp phím Ctrl + K + C).

Bước 22. Sau đó, bổ sung đoạn mã sau trong hàm Main:

```
// Tạo một phân số có mẫu bằng 0.
PhanSo e = new PhanSo();
e.TuSo = 2;
e.MauSo = 0;

// Gọi phương thức tính giá trị phân số e
double gte = e.TinhGiaTri();

// Xuất phân số e và giá trị của nó
Console.WriteLine("{0} = {1}", e, gte);
```

Bước 23. Chạy chương trình và ghi nhận kết quả. Giải thích kết quả xuất ra màn hình.

Từ ví dụ trên cho thấy, ta có thể gán một số nguyên bất kỳ cho TuSo và MauSo. Việc này có thể dẫn đến một kết quả sai hoặc gây ra một lỗi trong quá trình thực thi (chạy chương trình). Đó là do lớp phân số chưa có cơ chế để ràng buộc dữ liệu. Ta có thể khắc phục điều này bằng cách tạo ra các thuộc tính và đóng gói các biến thành viên hiện có cũng như các ràng buộc về dữ liệu.

Bước 24. Trong lớp phân số, sửa đổi lại mã nguồn như sau:

```
// Định nghĩa lớp phân số đơn giản
class PhanSo
{
    // Khai báo các biến thành viên (hay trường)
    private int _tu;
    private int _mau;

    // Định nghĩa các thuộc tính
    public int TuSo
    {
        get { return _tu; }
        set { _tu = value; }
    }

    public int MauSo
    {
        get { return _mau; }
        set
        {
            if (value == 0)
```

```
            _mau = 1;
        else
            _mau = value;
    }
}

// Định nghĩa phương thức khởi tạo không có tham số
public PhanSo() {}
```

Bước 25. Chạy chương trình với hàm Main giữ nguyên như bước 22 và ghi nhận kết quả. Hãy giải thích kết quả xuất ra màn hình.

Bước 26. Trong đoạn mã trên, mỗi thuộc tính tương ứng với một biến thành viên. Ngoài ra, mỗi thuộc tính có 2 bộ truy xuất: get và set. Tuy nhiên, điều này là không nhất thiết. Ta có thể định nghĩa nhiều thuộc tính lấy hoặc tác động lên dữ liệu từ một biến thành viên và ngược lại. Chẳng hạn, ta có thể định nghĩa thuộc tính GiaTri (cho biết giá trị của phân số) trong lớp PhanSo như sau:

```
// Định nghĩa thuộc tính chỉ đọc
public double GiaTri
{
    get { return TinhGiaTri(); }
}
```

Bước 27. Thay đổi lại đoạn mã trong hàm Main như sau:

```
// Tạo một phân số mới.
PhanSo e = new PhanSo();
e.TuSo = 2;
e.MauSo = 5;

// Xuất phân số e và giá trị của nó
Console.WriteLine("{0} = {1}", e, e.GiaTri);
```

Bước 28. Chạy chương trình và cho biết kết quả. Có thể gán e._tu = 2 được không? Tại sao?

8. Nạp chồng phương thức

Nạp chồng phương thức là việc định nghĩa nhiều phương thức cùng tên nhưng khác nhau về tham số. Phần này minh họa cách nạp chồng phương thức nhân phân số với một số hoặc một phân số.

Bước 29. Trong lớp PhanSo, định nghĩa thêm hai phương thức sau:

```
// Định nghĩa phương thức nhân phân số với 1 số
public PhanSo Nhan(int k)
{
    return new PhanSo(TuSo * k, MauSo);
}
```

```
// Định nghĩa phương thức nhân phân số
// hiện tại với một phân số khác
public PhanSo Nhan(PhanSo x)
{
    return new PhanSo(TuSo * x.TuSo, MauSo * x.MauSo);
}
```

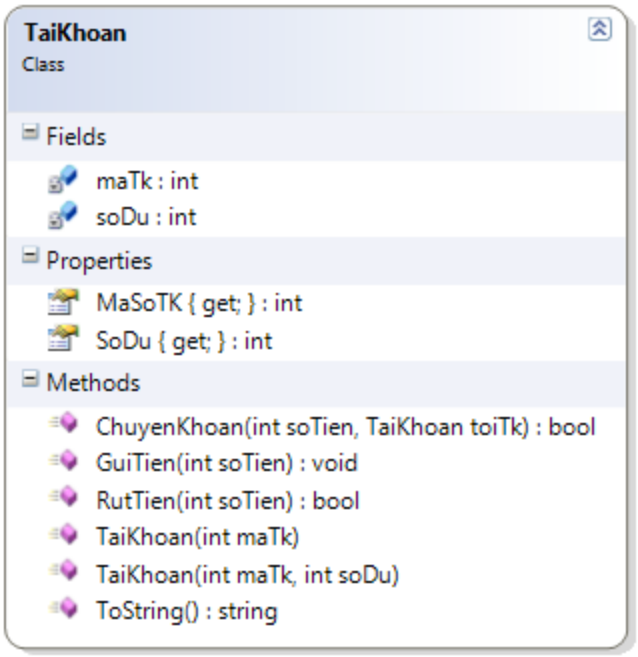
Bước 30. Trong hàm Main, bổ sung thêm đoạn mã sau để gọi phương thức Nhan:

```
PhanSo d = new PhanSo(6, 4);

// Gọi phương thức nhân hai phân số
PhanSo tich = d.Nhan(e);

// Gọi phương thức nhân phân số e với số k=3
PhanSo nhanK = e.Nhan(3);

// Xuất kết quả
Console.WriteLine("{0} * {1} = {2}", d, e, tich);
Console.WriteLine("{0} * {1} = {2}", e, 3, nhanK);
```

Bước 31. Chạy chương trình và ghi nhận kết quả

9. Sử dụng từ khóa this

Từ khóa this có thể được sử dụng để làm tham số khi muốn truyền đối tượng hiện tại vào một hàm hoặc được dùng khi biến thành viên của lớp trùng tên với tham số. Nó cũng được dùng thể gọi một phương thức khởi tạo từ một phương thức khởi tạo khác hay dùng để tạo chỉ mục (index).



Phần này hướng dẫn sử dụng từ khóa this thông qua lớp TaiKhoan đơn giản như hình bên.

Bước 1. Tạo một dự án kiểu Console Application mới, đặt tên là Lab03_Bai02_TuKhoaThis và lưu vào thư mục đã tạo ở đầu bài Lab.

Bước 2. Tạo một lớp mới bằng cách nhấp phải chuột vào dự án, chọn Add > Class … Đặt tên lớp là TaiKhoan (tài khoản). Cài đặt lớp phân số như đoạn mã sau:

```
// Định nghĩa lớp mô tả một tài khoản ngân hàng đơn giản
// Lớp này chứa các thuộc tính chỉ đọc
class TaiKhoan
{
    // Khai báo các biến thành viên
    private int maTk;       // Mã số tài khoản
    private int soDu;       // Số tiền còn trong tài khoản

    // Định nghĩa các thuộc tính
    public int MaSoTK
    {
        get { return maTk; }
    }

    public int SoDu
    {
        get { return soDu; }
    }

    // Định nghĩa phương thức khởi tạo
    // Sử dụng từ khóa this để tránh xung đột tên
    public TaiKhoan(int maTk, int soDu)
    {
        this.maTk = maTk;
        this.soDu = soDu;
    }

    // Sử dụng từ khóa this để gọi một phương thức tạo lập
    public TaiKhoan(int maTk)
        : this(maTk, 0)
    {
    }


    // Định nghĩa phương thức gửi tiền
    public void GuiTien(int soTien)
    {
        this.soDu += soTien;
    }

    // Định nghĩa phương thức rút tiền
    public bool RutTien(int soTien)
    {
        // Kiểm tra số dư tài khoản trước khi rút
        if (soDu >= soTien) {
            this.soDu -= soTien;
            return true;
        }
        else {
            Console.WriteLine("Tai khoan khong co du tien");
            return false;
        }
    }
```

```
        // Định nghĩa phương thức chuyển một số tiền
        // từ tài khoản hiện tại tới một tài khoản khác
        // Sử dụng this khi gọi một hàm (không nhất thiết)
        public bool ChuyenKhoan(int soTien, TaiKhoan toiTk)
        {
            var thanhCong = this.RutTien(soTien);
            if (thanhCong)
                toiTk.GuiTien(soTien);

            return thanhCong;
        }

        // Sinh ra chuỗi mô tả thông tin về tài khoản
        public override string ToString()
        {
            return string.Format("Tai khoan {0} co so du {1}",
                                 maTk, soDu);
        }
    }
```

Bước 3. Trong hàm Main, nhập đoạn mã sau:

```
        // Tạo tài khoản nguồn có mã 1234
        TaiKhoan nguon = new TaiKhoan(1234);

        // Xuất thông tin tài khoản nguồn
        Console.WriteLine("Ban dau:");
        Console.WriteLine(nguon);
        Console.WriteLine("================================");
```

Bước 4. Chạy chương trình và cho biết kết quả.

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

```
        // Nạp số tiền 10 000 000 vào tài khoản nguồn
        nguon.GuiTien(10000000);

        // Xuất thông tin tài khoản nguồn
        Console.WriteLine("Sau khi nap so tien 10 000 000");
        Console.WriteLine(nguon);
        Console.WriteLine("================================");
```

Bước 6. Chạy chương trình và cho biết kết quả.

Bước 7. Trong hàm Main, tiếp tục nhập đoạn mã sau:

```
        // Rút một số tiền 2 500 000 từ tài khoản
        if (nguon.RutTien(2500000))
            Console.WriteLine("Rut tien thanh cong");
        else
            Console.WriteLine("So du khong du de rut");
```

```
// Xuất thông tin tài khoản nguồn
Console.WriteLine("Sau khi rut so tien 2 500 000");
Console.WriteLine(nguon);
Console.WriteLine("================================");
```

Bước 8. Chạy chương trình và cho biết kết quả.

Bước 9. Trong hàm Main, tiếp tục bổ sung đoạn mã sau:

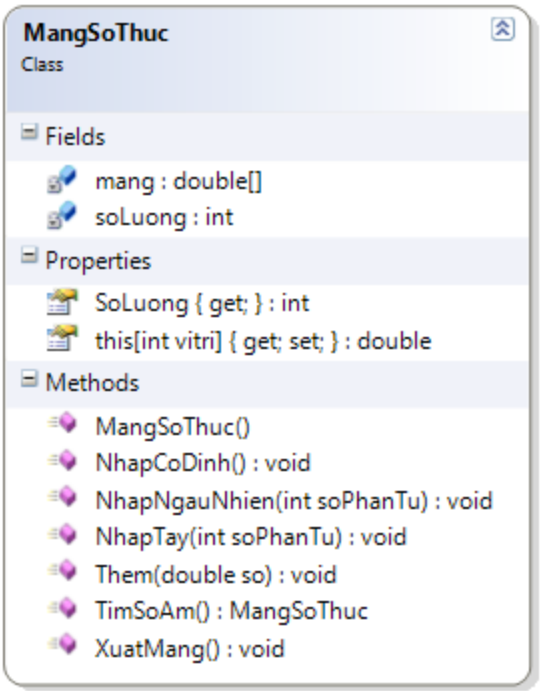
```
// Tạo tài khoản đích có mã 9876 với số dư 3 triệu
TaiKhoan dich = new TaiKhoan(9876, 3000000);

// Gọi hàm chuyển khoản để chuyển 5 triệu từ
// tài khoản nguồn sang tài khoản đích
bool ketQua = nguon.ChuyenKhoan(5000000, dich);
if (ketQua)
{
    Console.WriteLine("Chuyen khoan thanh cong");
    Console.WriteLine(nguon);
    Console.WriteLine(dich);
    Console.WriteLine("================================");
}

// Gọi hàm chuyển khoản lần 2, chuyển thêm 3 triệu
nguon.ChuyenKhoan(3000000, dich);
```

Bước 10. Chạy chương trình và ghi nhận kết quả.

Bước 11. Hãy mô tả lại một cách ngắn gọn hoạt động của toàn bộ chương trình trên.

10. Tạo chỉ mục bằng từ khóa this

Chỉ mục (index) cho phép truy xuất tới một đối tượng giống như truy xuất tới mảng. Tuy nhiên, chỉ số không bị giới hạn là số nguyên mà có thể có kiểu bất kỳ. Phần này minh họa cách tạo một chỉ mục đơn giản để truy xuất tới các phần tử của mảng được chứa bên trong một đối tượng.



Bước 1. Tạo một dự án kiểu Console Application mới, đặt tên là Lab03_Bai03_TaoChiMuc.

Bước 2. Tạo một lớp mới bằng cách nhấp phải chuột vào dự án, chọn Add > Class ... Đặt tên lớp là MangSoThuc (mảng số thực). Cài đặt lớp MangSoThuc như sau:

```
// Lớp này biểu diễn mảng một chiều chứa các số thực
public class MangSoThuc
{
    // Khai báo các biến thành viên
    private double[] mang; // Mảng một chiều
    private int soLuong; // Số lượng phần tử

    // Định nghĩa thuộc tính chỉ đọc để lấy số phần tử
    public int SoLuong
    {
        get { return soLuong; }
    }

    // Định nghĩa chỉ mục dùng con trỏ this
    public double this[int vitri]
    {
        // Lấy phần tử tại vị trí
        get { return mang[vitri]; }

        // Gán giá trị cho phần tử tại vị trí
        set { mang[vitri] = value; }
    }

    // Định nghĩa phương thức khởi tạo
    public MangSoThuc()
    {
        mang = new double[1000];
        soLuong = 0;
    }

    // Định nghĩa hàm thêm một phần tử (số) mới
    public void Them(double so)
    {
        mang[soLuong] = so; // Lưu số vào cuối mảng
        soLuong++; // Tăng số lượng lên 1
    }
}
```

Bước 3. Trong hàm Main, nhập đoạn mã sau:

```
// Tạo một thể hiện của lớp MangSoThuc
MangSoThuc array = new MangSoThuc();

array.Them(10.5);
array.Them(3.0);
array.Them(1.75);
array.Them(100.2);
array.Them(-6.04);
array.Them(3.33);
array.Them(-89.345);
array.Them(-12.19);
array.Them(Math.Round(Math.PI, 3)); // Số PI
```

```
array.Them(Math.Round(Math.E, 3));  // Số e

// Truy xuất và lấy giá trị phần tử thứ 3
double giaTri = array[3];

// xuất giá trị vừa lấy
Console.WriteLine("array[3] = {0}", giaTri);

// Duyệt qua từng phần tử và xuất ra màn hình
for (int i = 0; i < array.SoLuong; i++)
{
    Console.Write("{0}\t", array[i]);
}

// Gán (thay đổi) giá trị cho phần tử thứ 6
array[6] = 10.077;

// Duyệt qua từng phần tử và xuất ra màn hình
for (int i = 0; i < array.SoLuong; i++)
{
    Console.Write("{0}\t", array[i]);
}
```

Bước 4. Chạy chương trình và ghi nhận kết quả.

Bước 5. Cài đặt hoàn chỉnh các phương thức sau:

Phương thức	Ý nghĩa
public void NhapTay(int soPhanTu)	Hàm nhập các phần tử từ bàn phím
public void NhapCoDinh()	Hàm nhập một mảng số thực cố định
public void NhapNgauNhien(int soPhanTu)	Hàm nhập các phần tử ngẫu nhiên
public void XuatMang()	Hàm xuất các phần tử ra màn hình
public MangSoThuc TimSoAm()	Hàm tìm tất cả các số âm

Bước 6. Trong hàm Main, viết các lời gọi hàm để kiểm tra các phương thức đã viết.

Bước 7. Chạy chương trình và ghi nhận kết quả.

11. Cài đặt lớp Menu để thực hiện các chức năng của chương trình

Trong phần này, ta sẽ cài đặt lại Lab02_Bai04_CacThaoTacMang. Tuy nhiên, các phương thức xử lý Menu được cài đặt tập trung trong lớp Menu.

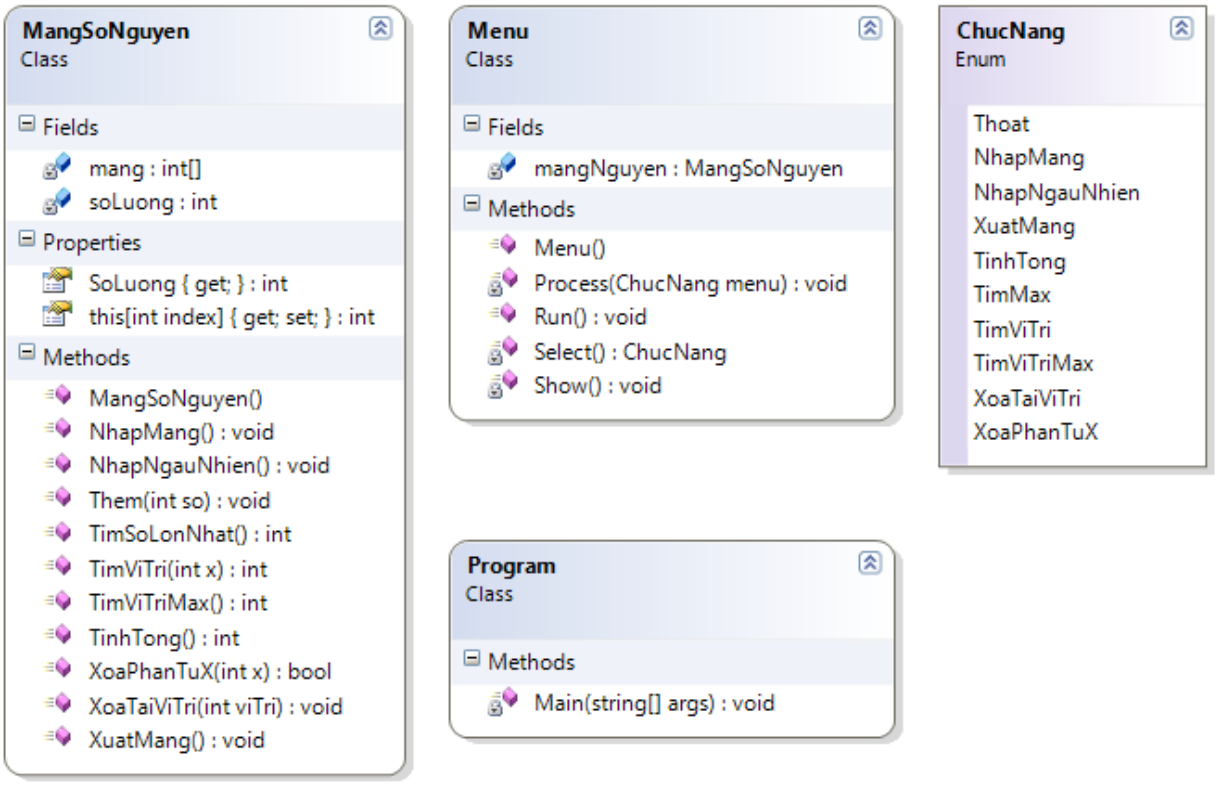
Bước 1. Tạo một dự án kiểu Console Application mới, đặt tên là Lab03_Bai04_TaoLopMenu.

Bước 2. Tạo lớp mới, đặt tên là MangSoNguyen. Cài đặt các phương thức cho lớp này theo sơ đồ lớp bên dưới. Các phương thức có thể được chép từ Lab02_Bai04_CacThaoTacMang.

Bước 3. Tạo một enum mới đặt tên là ChucNang (chức năng), tạo các hằng số như hình dưới.

Bước 4. Tạo một lớp mới, đặt tên là Menu.

Ta cần cài đặt các lớp (chủ yếu là lớp Menu) trên theo sơ đồ lớp sau:



Bước 5. Trong lớp Menu, bổ sung đoạn mã sau

```
// Khai báo các biến thành viên
private MangSoNguyen mangNguyen;

// Định nghĩa phương thức khởi tạo
public Menu()
{
    mangNguyen = new MangSoNguyen();
}
```

Bước 6. Tiếp tục cài đặt hàm xuất danh sách menu chức năng ra màn hình

```
// Định nghĩa hàm xuất thực đơn ra màn hình
private void Show()
{
    Console.WriteLine();
    Console.WriteLine("====================== MENU ======================");
    Console.WriteLine("{0}. Nhap mang bang tay", (int)ChucNang.NhapMang);
    Console.WriteLine("{0}. Nhap ngau nhien", (int)ChucNang.NhapNgauNhien);
    Console.WriteLine("{0}. Xuat cac phan tu cua mang", (int)ChucNang.XuatMang);
    Console.WriteLine("{0}. Tinh tong cac phan tu", (int)ChucNang.TinhTong);
    Console.WriteLine("{0}. Tim phan tu lon nhat", (int)ChucNang.TimMax);
    Console.WriteLine("{0}. Tim vi tri cua phan tu X", (int)ChucNang.TimViTri);
    Console.WriteLine("{0}. Tim vi tri phan tu lon nhat", (int)ChucNang.TimViTriMax);
```

```
    Console.WriteLine("{0}. Xoa phan tu tai vi tri cho truoc", (int)ChucNang.XoaTaiViTri);
    Console.WriteLine("{0}. Xoa phan tu X khoi mang", (int)ChucNang.XoaPhanTuX);
    Console.WriteLine("{0}. Thoat", (int)ChucNang.Thoat);
    Console.WriteLine("================================================");
}
```

Bước 7. Cài đặt hàm chọn menu như sau:

```
// Định nghĩa hàm chọn một thực đơn
private ChucNang Select()
{
    // Lấy số lượng menu
    int soMenu = Enum.GetNames(typeof(ChucNang)).Length;

    // Khai báo biến để lưu số thứ tự menu mà người dùng chọn
    int menu = 0;

    do
    {
        this.Show();
        Console.Write("Nhap so de chon menu (0..{0}) : ", soMenu);

        menu = int.Parse(Console.ReadLine());
    } while (menu < 0 || menu >= soMenu);
    // So 10 ở dòng trên là số lượng menu

    return (ChucNang)menu;
}
```

Bước 8. Định nghĩa hàm gọi các phương thức và xử lý theo chức năng (menu) được chọn

```
// Định nghĩa hàm thực hiện chức năng tương ứng với menu đã chọn
private void Process(ChucNang menu)
{
    // Khai báo các biến cục bộ
    int tong, x, max, viTri;

    switch (menu)
    {
        case ChucNang.NhapMang:
            mangNguyen.NhapMang();
            break;

        case ChucNang.NhapNgauNhien:
            mangNguyen.NhapNgauNhien();
            Console.WriteLine("Da nhap xong, chon chuc nang 3 de xem");
            break;

        case ChucNang.XuatMang:
            mangNguyen.XuatMang();
            break;
```

```
            case ChucNang.TinhTong:
                // Gọi hàm tính tổng
                tong = mangNguyen.TinhTong();
                Console.WriteLine("Tong cac phan tu cua mang la : {0}", tong);
                break;

            default:
                Console.WriteLine("Ket thuc chuong trinh");
                break;
        }
    }
```

Bước 9. Định nghĩa hàm lặp lại việc xuất -> chọn -> xử lý menu

```
// Định nghĩa hàm cho phép người dùng chọn menu
// và thực hiện các chức năng tương ứng
public void Run()
{
    ChucNang menu = ChucNang.Thoat;
    do
    {
        menu = this.Select();
        this.Process(menu);
    } while (menu != ChucNang.Thoat);
}
```

Bước 10. Trong hàm Main, nhập đoạn mã sau:

```
// Tạo một thể hiện của lớp Menu
Menu trinhDon = new Menu();

// Gọi hàm xử lý
trinhDon.Run();

// Chờ nhấn phím để kết thúc chương trình
Console.ReadKey();
```

Bước 11. Chạy chương trình, lần lượt chọn từng menu và ghi nhận lại kết quả.

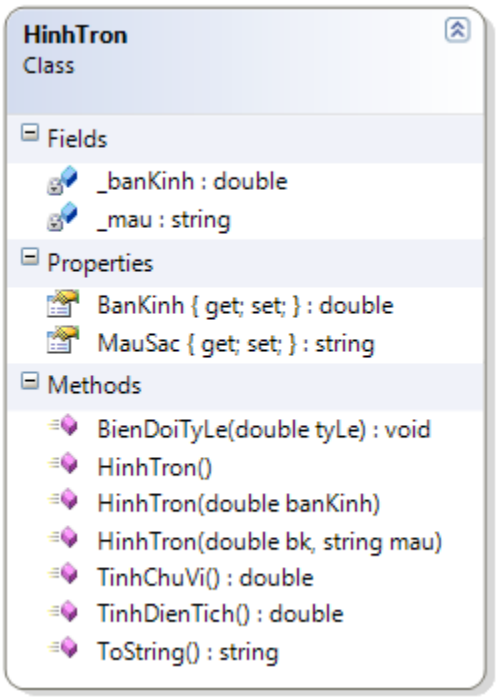
Bước 12. Mô tả ngắn gọn hoạt động của toàn bộ chương trình.

Bước 13. Tiếp tục hoàn thiện phương thức Process trong lớp Menu để xử lý trường hợp người dùng chọn các chức năng khác như TimMax, XoaTaiViTri, …

Bước 14. Chạy chương trình và ghi nhận kết quả.

D. Bài tập bắt buộc

1. Cài đặt lớp HinhTron (hình tròn) theo sơ đồ lớp và mô tả sau:



Trong đó, phương thức BienDoiTyLe dùng để phóng to, thu nhỏ hình tròn theo một tỷ lệ cho trước.

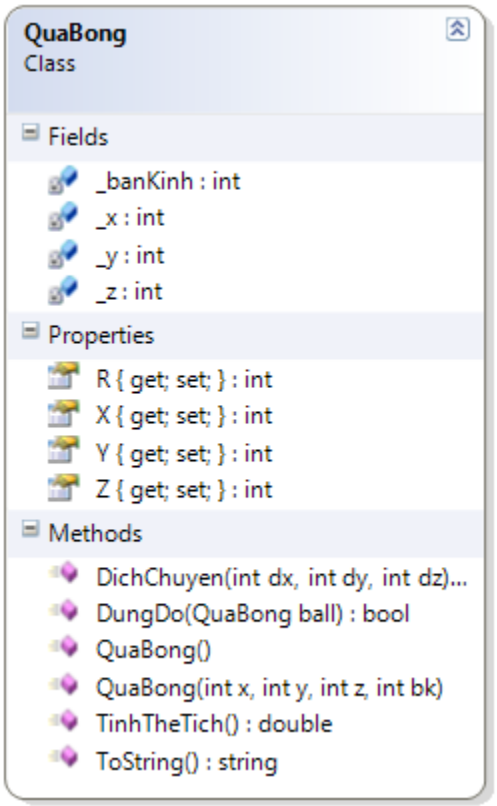
Thực hiện tiếp các yêu cầu sau:

- Tạo mảng chứa các hình tròn theo mô tả sau:

Bán kính	Màu	Tỷ lệ biến đổi
5.75	Xanh	30
23	Đỏ	7.5
9.12	Vàng	13.36
238.7635	Tím	2.71

- Sắp xếp các hình tròn tăng theo diện tích.
- Sắp xếp các hình tròn giảm theo chu vi.
- Gọi hàm biến đổi tỷ lệ ứng với mỗi hình tròn theo tỷ lệ được cho trong cột thứ 3 ở bảng trên.
- Tính tổng chu vi và diện tích tất cả các hình.

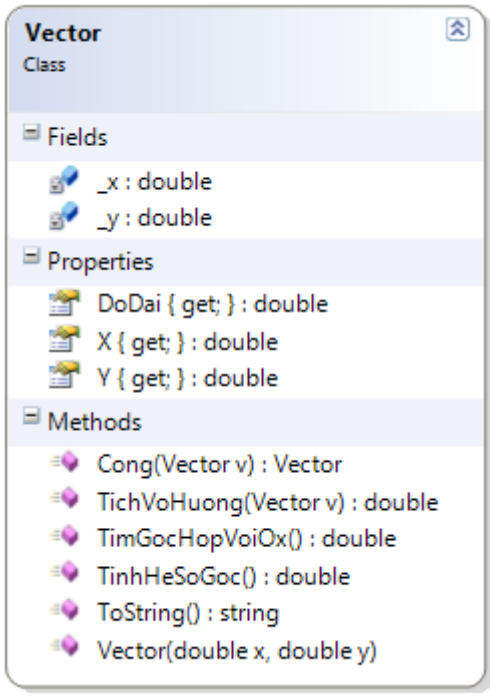
2. Cài đặt lớp QuaBong (quả bóng có hình cầu) theo đặc tả sau:



Trong đó, X, Y, Z là tọa độ tâm và R là bán kính của quả bóng. Hãy thực hiện các yêu cầu sau:

- Tạo hai thể hiện của lớp QuaBong, đặt tên là red và blue theo hai cách khởi tạo khác nhau.
- Xuất thông tin của hai đối tượng red, blue.
- Xuất thể tích của hai quả bóng
- Cho biết quả bóng nào có thể tích lớn hơn.
- Có nhất thiết phải so sánh thể tích của hai quả bóng để biết quả bóng nào lớn hơn hay không? Tại sao?
- Dịch chuyển từng quả bóng bằng cách gọi hàm dịch chuyển. Sau đó xuất ra tọa độ mới.
- Kiểm tra hai quả bóng red và blue có bị đụng độ nhau hay không? Ta nói hai quả bóng đụng độ nhau nếu khoảng cách giữa tâm của hai quả bóng nhỏ hơn hoặc bằng tổng bán kính của hai quả bóng đó.

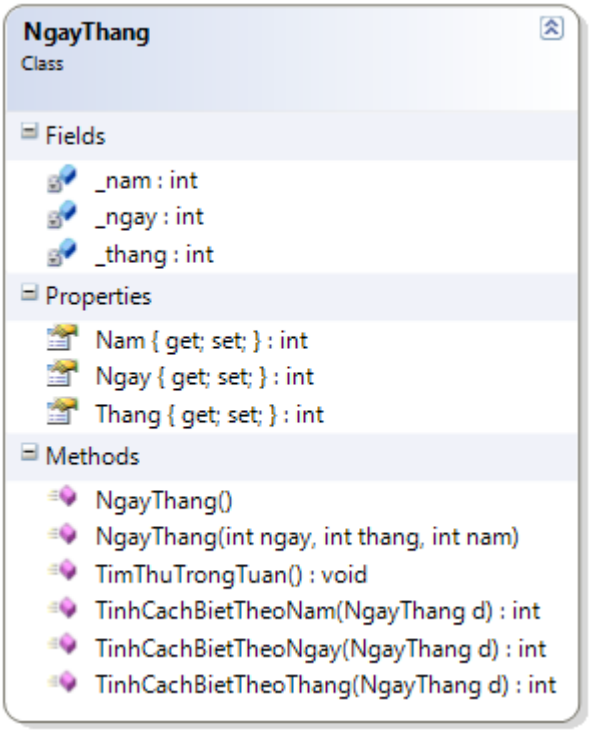
3. Cài đặt lớp Vector (vec-tơ) trong không gian hai chiều theo đặc tả sau:



Trong đó, thuộc tính DoDai trả về độ dài của vec-tơ. Hãy thực hiện các yêu cầu sau:

- Tạo hai vector u(5, -3) và v(4, 9).
- Xuất thông tin của hai vector u, v
- Tính và xuất độ dài của mỗi vector
- Tìm hệ số góc của mỗi vector
- Cài đặt phương thức tính góc hợp bởi vector với Ox
- Cho biết góc hợp bởi Ox với từng vector u, v
- Tính tích vô hướng của vector u và v.
- Xuất vector tổng của hai vector u và v.
- Cho biết độ dài của vector tổng
- Cài đặt thêm các hàm: tìm góc hợp bởi 2 vector, kiểm tra hai vector có song song hay không, nếu là song song thì chúng cùng chiều hay ngược chiều.

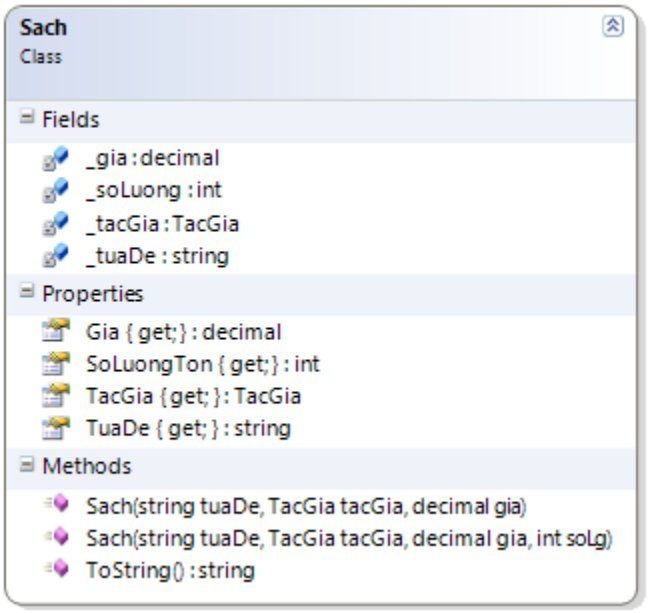
4. Cài đặt lớp NgayThang (ngày tháng) theo đặc tả sau:

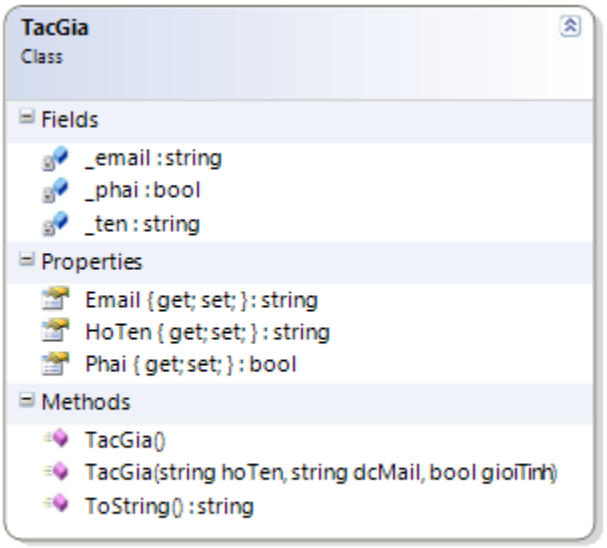


Hãy định nghĩa các phương thức và viết mã lệnh trong hàm Main để thực hiện các chức năng sau:

- Tạo 2 thể hiện của lớp NgayThang như sau: quaKhu = 23/09/1990, hienTai = ngày hiện tại theo hai cách khởi tạo khác nhau.
- Xuất thông tin ngày tháng của 2 đối tượng đó
- Xuất từng thành phần ngày, tháng, năm của đối tượng quaKhu.
- Cho biết thứ trong tuần ứng với ngày được lưu trong đối tượng quaKhu. Gợi ý: xem bài tập 10, Lab 2 để biết cách tính thứ trong tuần.
- Cho biết cách biệt giữa quaKhu và ngày hiện tại là bao nhiêu ngày, bao nhiêu tháng, bao nhiêu năm. Giả sử cách biệt là 124 tháng 3 ngày thì được xem là 125 tháng.
- Cài đặt hàm để kiểm tra 1 đối tượng NgayThang có phải là ngày trong năm nhuận?

5. Cài đặt lớp Sach (sách) và TacGia (tác giả) theo đặc tả sau:





Hãy thực hiện các yêu cầu sau:

- Tạo 3 thể hiện của lớp TacGia, đặt tên là an, binh, cau.
- Tạo 5 thể hiện của lớp Sach, đặt tên lần lượt là web, mobile, game, dos, card. Tác giả của những sách này có thể chọn tùy ý từ 3 đối tượng an, binh, cau.
- Xuất thông tin các tác giả (có canh lề các cột tương tự như bài tập 2, Lab 1)
- Xuất thông tin các cuốn sách (có canh lề các cột tương tự như bài tập 2, Lab 1)
- Với mỗi tác giả, hãy cho biết tên và số lượng sách do tác giả đó viết.
- Với mỗi tác giả, hãy xuất ra tên tác giả và danh mục sách do tác giả đó viết

6. ***Số phức*** là số có dạng **a+bi**, trong đó a và b là các số thực, i là đơn vị ảo, với i^2=-1. Trong biểu thức này, số a gọi là ***phần thực***, b gọi là ***phần ảo*** của số phức. Số phức có thể được biểu diễn trên mặt phẳng phức với trục hoành là trục thực và trục tung là trục ảo, do đó một số phức a+bi được xác định bằng một điểm có tọa độ (a,b). Một số phức nếu có phần thực bằng không thì gọi là ***số thuần ảo***, nếu có phần ảo bằng không thì trở thành là số thực.

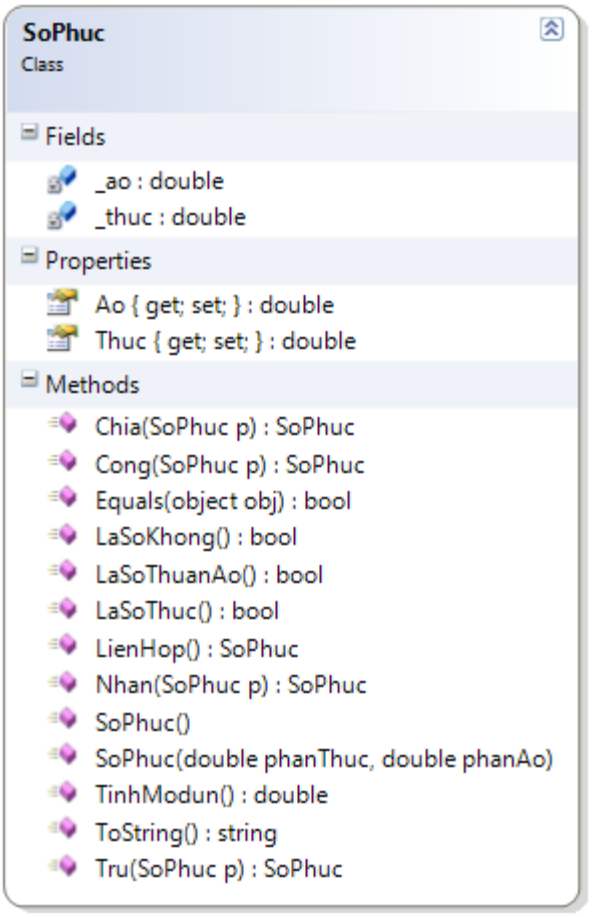
   Một số công thức tính toán liên quan:

   - Cộng, trừ, nhân, chia số phức

   ```
   (a + bi) + (c + di) = (a+c) + (b+d)i
   (a + bi) - (c + di) = (a-c) + (b-d)i
   (a + bi) * (c + di) = (ac - bd) + (ad + bc)i
   (a + bi) / (c + di) = [(a + bi) * (c - di)] / (c2 + d2)
   ```

   - Số phức liên hợp (conjugate) của số phức a + bi là một số phức có dạng: a – bi
   - Modun hay độ lớn (magnitude) của số phức a + bi là $\sqrt{a^2 + b^2}$

- Phương thức Equals dùng để kiểm tra hai số phức có bằng nhau hay không.



Hãy thực hiện các yêu cầu sau:

- Cài đặt lớp SoPhuc theo mô tả trong sơ đồ lớp
- Tạo ra hai số phức z = 3.5 – 4.2i, t = 1.75 + 2.3i
- Xuất hai số phức z, t ra màn hình
- Tính và xuất mô-đun của 2 số phức
- Tìm và xuất số phức liên hợp của z, t
- Gọi các hàm tính toán trên hai số phức z, t
- Tìm và xuất các số phức liên hợp của tổng, hiệu, tích, thương của hai số phức z, t
- Gọi các hàm LaSoKhong, LaSoThuanAo, LaSoThuc trên 2 số phức z, t để kiểm tra.
- Kiểm tra hai số phức z và t có bằng nhau?
- Viết chương trình cho phép người dùng nhập vào một mảng các số phức.
- Xuất mảng số phức ra màn hình
- Tính tổng tất cả các số phức trong mảng
- Tạo ra một mảng các số phức liên hợp của các số phức trong mảng đã nhập.

7. Đa thức một biến bậc n có dạng: $F(x) = c_n x^n + c_{n-1} x^{n-1} + \dots + c_2 x^2 + c_1 x + c_0$. Trong đó, $c_i x^i$ được gọi là một đơn thức. Ví dụ:

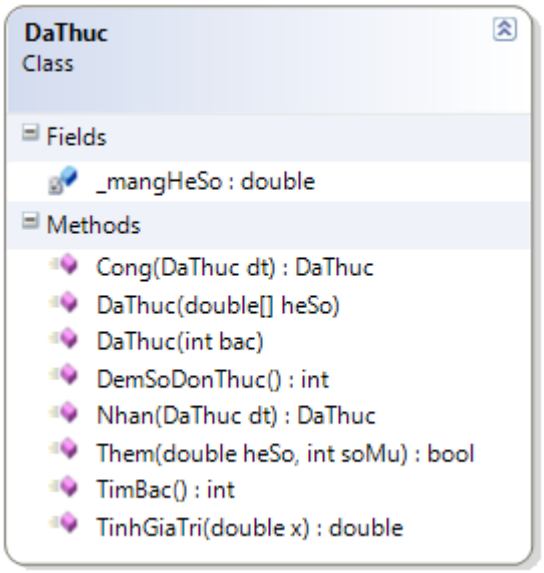
   $F(x) = -3x^6 + 6x^4 + 1.2x^3 - 10.55x^2 - 3.14x + 9 = 9x^0 - 3.14x^1 - 10.55x^2 + 1.2x^3 + 6x^4 – 3x^6$

   Một đa thức như vậy có thể được biểu diễn bởi một mảng một chiều (gồm n+1 phần tử). Trong đó, mỗi phần tử của mảng chứa một hệ số của đa thức, chỉ số của phần tử tương ứng với số mũ. Chẳng hạn, với ví dụ trên, ta có mảng sau:

Chỉ số	0	1	2	3	4	5	6
Hệ số	9	-3.14	-10.55	1.2	6	0	-3

   Bậc của đa thức chính là số mũ lớn nhất trong đa thức đó. Cho x = 2, ta có thể tính giá trị của $F(x) = F(2) = 9*2^0 - 3.14*2^1 - 10.55*2^2 + 1.2*2^3 + 6*2^4 – 3*2^6 = 9 – 6.28 – 42.2 + 9.6 + 96 – 192 = -125.88$.

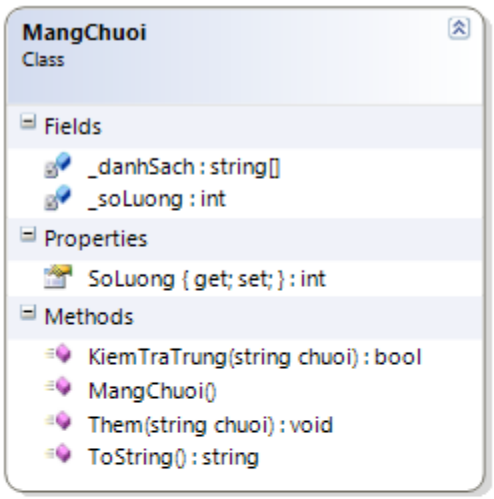
   Hãy cài đặt lớp DaThuc (đa thức) và các phương thức tính toán trên đa thức theo mô tả trên và sơ đồ lớp sau:

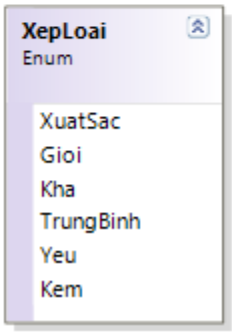


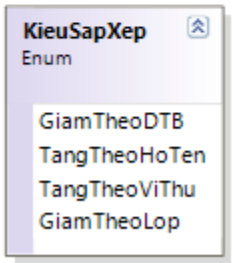
Thực hiện tiếp các yêu cầu sau:

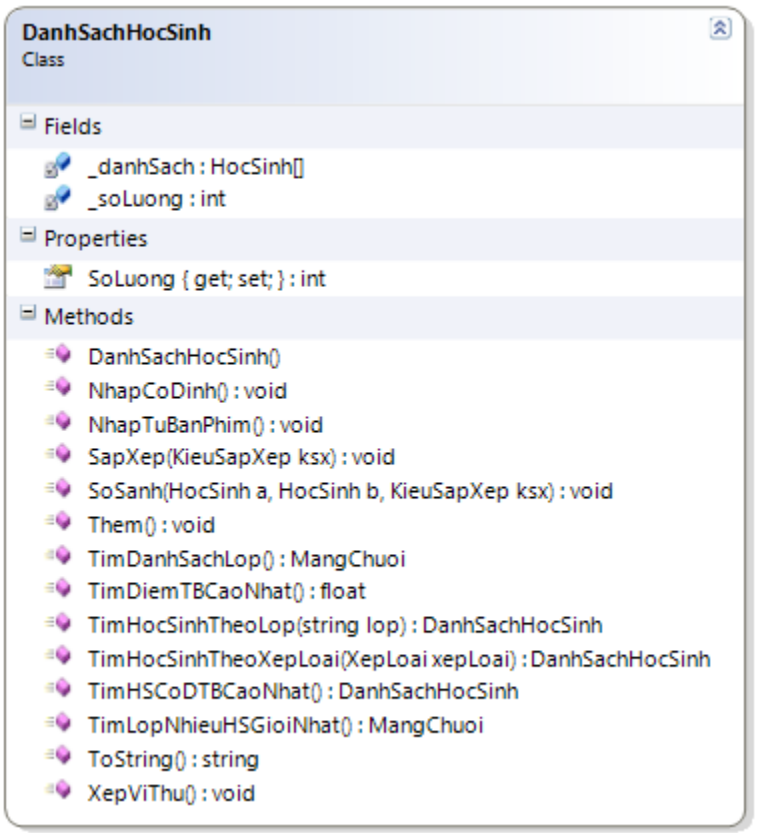
- Tạo hai đa thức fx và gx bằng 2 cách khởi tạo khác nhau. Trong đó, fx là đa thức được nhập cố định như trong ví dụ trên. Gx được nhập từ bàn phím.
- Xuất hai đa thức fx, gx ra màn hình (dùng dấu ^ để thể hiện lũy thừa)
- Tìm và xuất bậc của hai đa thức fx, gx
- Thực hiện các phép toán cộng, nhân hai đa thức fx, gx và xuất ra kết quả.
- Yêu cầu người dùng nhập giá trị x rồi tính và xuất ra giá trị của hai đa thức fx, gx.
- Cho biết số đơn thức trong fx có hệ số khác 0.

8. Xây dựng ứng dụng quản lý điểm của học sinh cấp 3 theo mô tả sau:



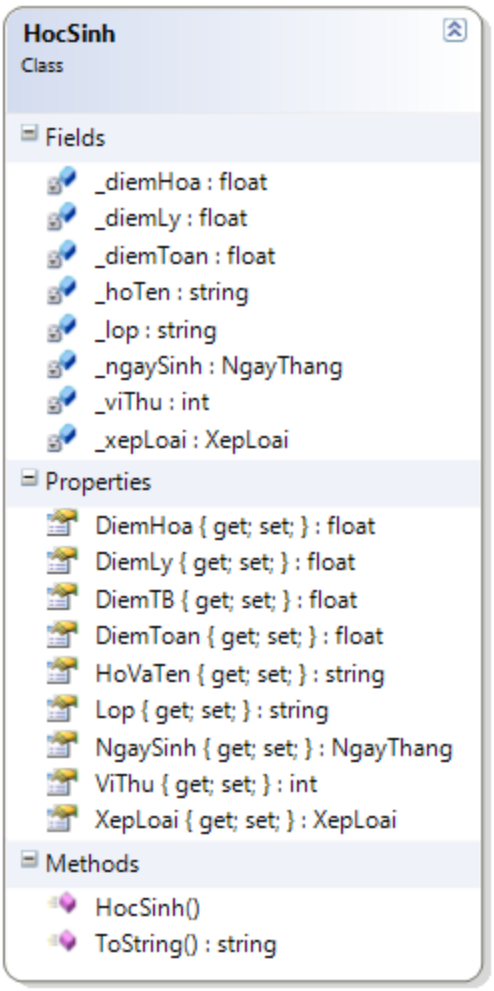






Trình tự cài đặt các lớp/enum: XepLoai, KieuSapXep, MangChuoi, HocSinh, DanhSachHocSinh.

Điểm trung bình = (Toán * 2 + Lý + Hóa) / 4. Điểm tối thiểu để không phải thi lại là 4.



Cài đặt và thử nghiệm thêm các chức năng sau:

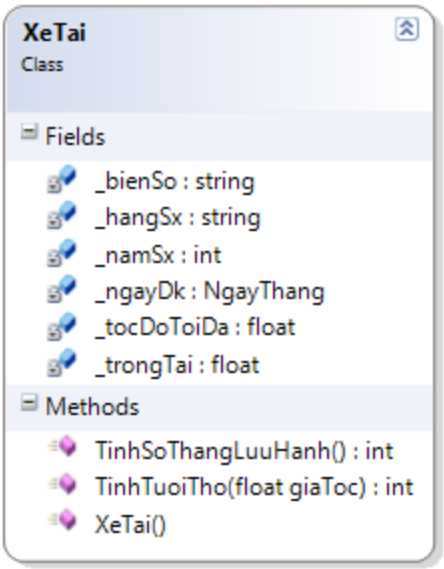
- Tìm danh sách học sinh thi lại môn toán theo lớp.
- Tìm tất cả học sinh không thi lại môn nào.
- Tìm những học sinh có họ Nguyễn và đạt loại Khá.
- Tính điểm trung bình chung của từng lớp.
- Xóa những học sinh có điểm trung bình dưới 2.
- Thống kê số lượng học sinh Xuất sắc, Giỏi, Khá, ... theo từng lớp
- Tìm những học sinh lớn hơn 15 tuổi.
- Tìm học sinh có tên dài nhất
- Tìm những học sinh xếp vị thứ 1 của mỗi lớp.
- Tìm những học sinh có xếp loại trung bình trở xuống.
- Cộng thêm 1.5 điểm cho tất cả những học sinh không rớt môn nào.
- Xóa những học sinh có họ Trần

Với các câu sau, xuất nội dung có định dạng & canh lề

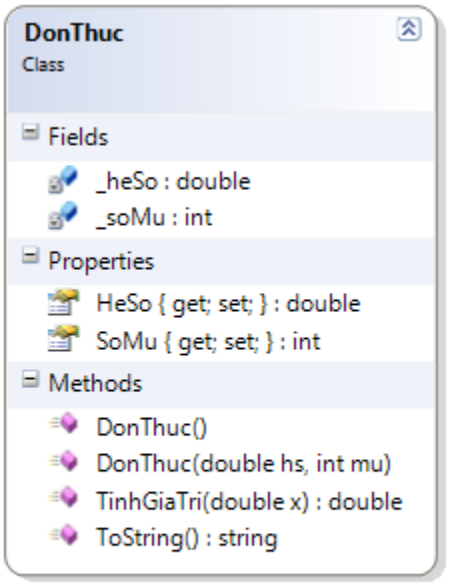
- Xuất danh sách học sinh theo từng lớp
- Xuất danh sách học sinh theo xếp loại
- Xuất danh sách học sinh theo từng năm sinh (sử dụng lớp MangSoNguyen ở bài Lab trước).

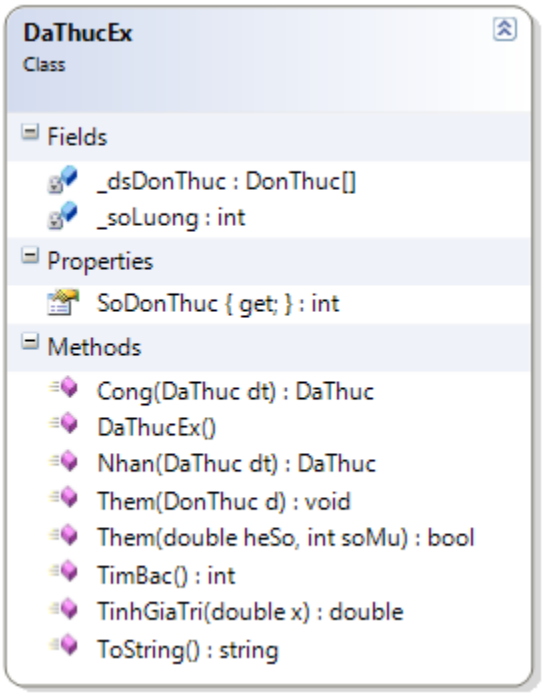
E. Bài tập làm thêm

1. Sửa đổi lớp Sach (sách) trong bài tập 5 ở trên để lưu nhiều tác giả.
2. Định nghĩa lớp XeTai (xe tải) có các thuộc tính Biển số, Trọng tải (theo tấn), Hãng sản xuất, Tốc độ tối đa (km/h), Năm sản xuất, Ngày đăng kiểm (sử dụng lớp NgayThang đã tạo trong bài tập 4 ở trên) và các hàm tính tuổi thọ của xe (gợi ý: dùng DateTime.Now.Year để lấy năm hiện thời), tính số thời gian xe đã lưu hành (kể từ ngày đăng kiểm) theo tháng.
   Sau đó, tạo 3 thể hiện của lớp XeTai, đặt tên là hyundai, suzuki, benz. Viết lời gọi hàm để kiểm tra hoạt động của các phương thức.

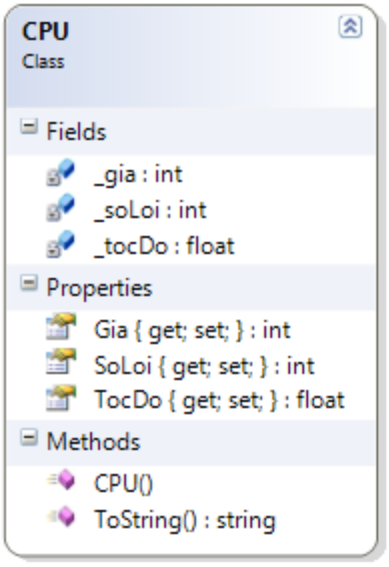


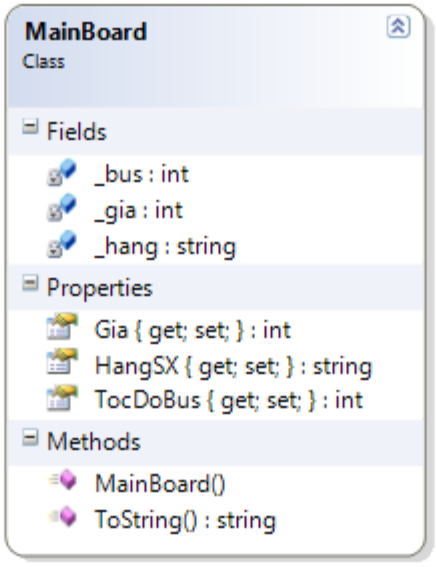
3. Cách biểu diễn đa thức trong bài tập 7 ở trên có một hạn chế là khi bậc của đa thức lớn nhưng số đơn thức ít, sẽ có nhiều phần tử của mảng mang giá trị 0. Các thao tác trên đa thức vì vậy cũng tốn thời gian vì luôn phải duyệt hết mảng. Để khắc phục nhược điểm này, ta phân tách đa thức thành nhiều đơn thức và lưu danh sách các đơn thức này. Mỗi đơn thức có hai thành phần là hệ số và số mũ. Hãy cài đặt lại lớp DaThuc và các thao tác trên đa thức theo mô tả sau:

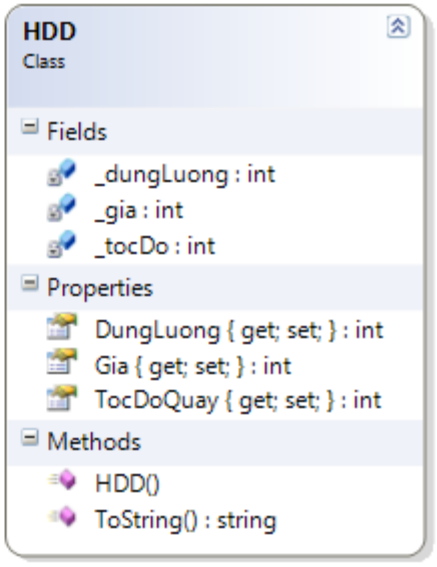


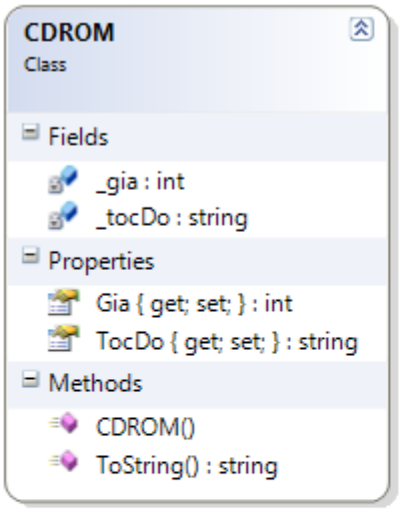


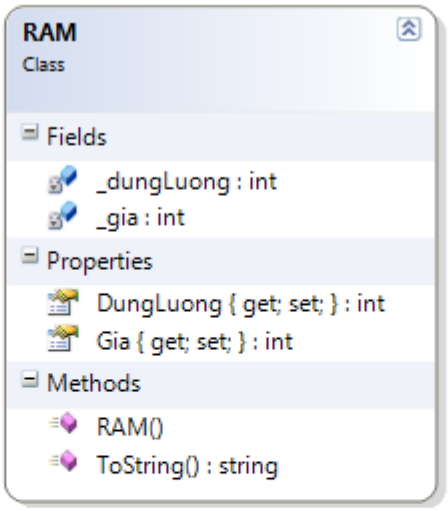
4. Cài đặt các lớp theo sơ đồ lớp sau

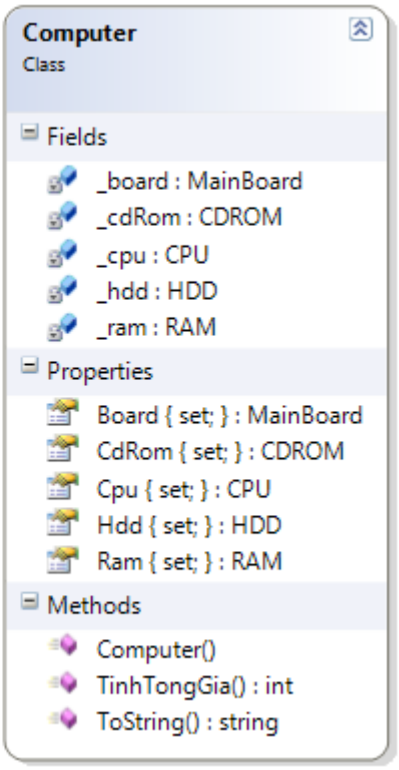






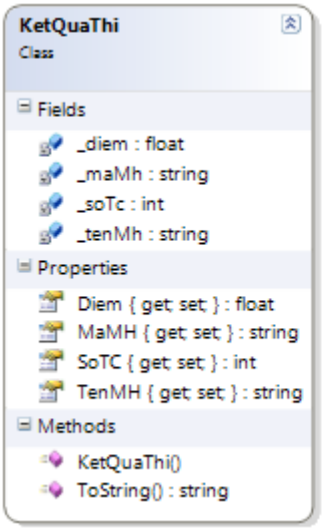


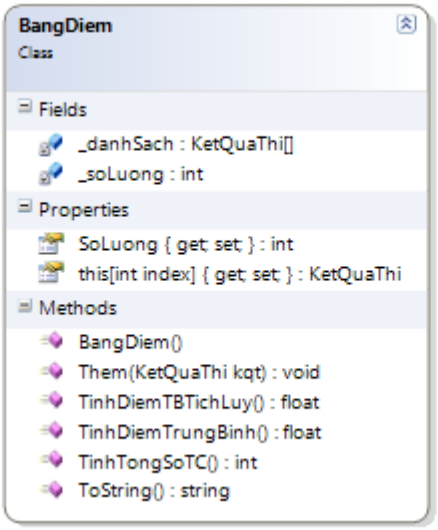


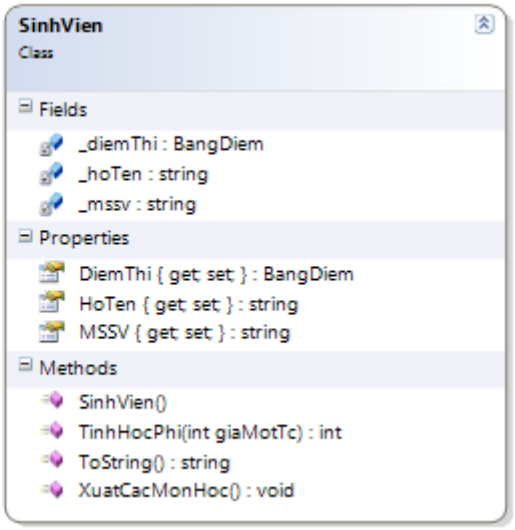


- Tạo 3 thể hiện của lớp Computer, đặt tên là acer, vaio, dell.
- Xuất thông tin của 3 đối tượng đã tạo (có định dạng, canh lề)
- Tính tổng giá thành mỗi máy

5. Cài đặt các lớp theo sơ đồ sau







Nhập thông tin 3 sinh viên và gọi các phương thức để kiểm tra kết quả.