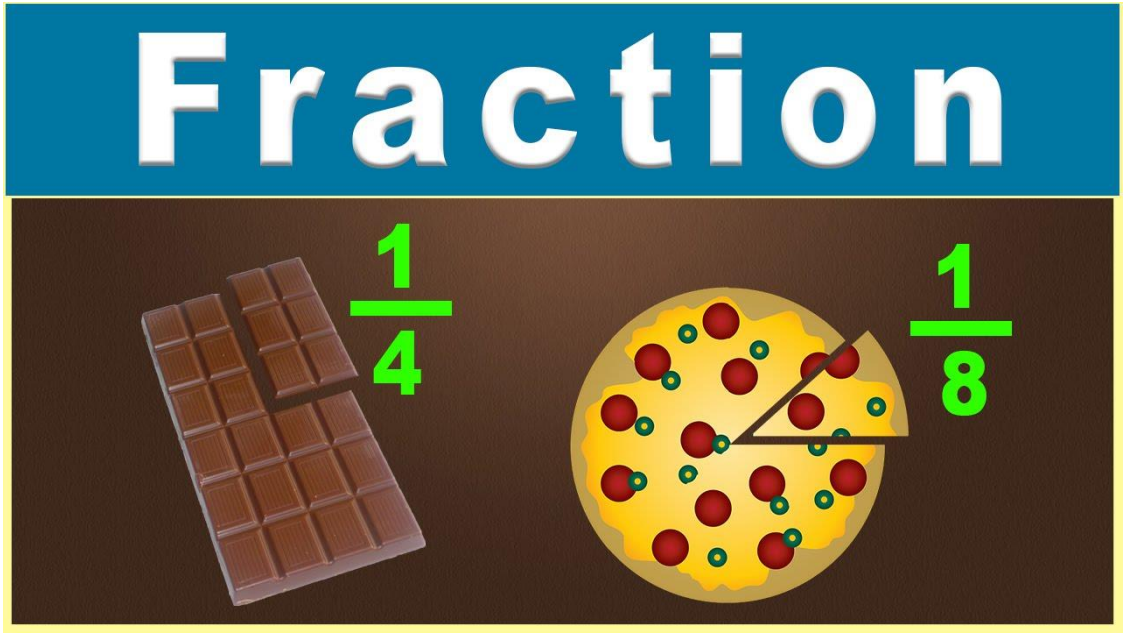
Chương 9
THIẾT KẾ LỚP

1. TS. Nguyễn Tấn Trần Minh Khang
2. ThS. Võ Duy Nguyên
3. ThS. Nguyễn Hoàng Ngân
4. Hồ Thái Ngọc – Source code.

1. ĐẶT VẤN ĐỀ

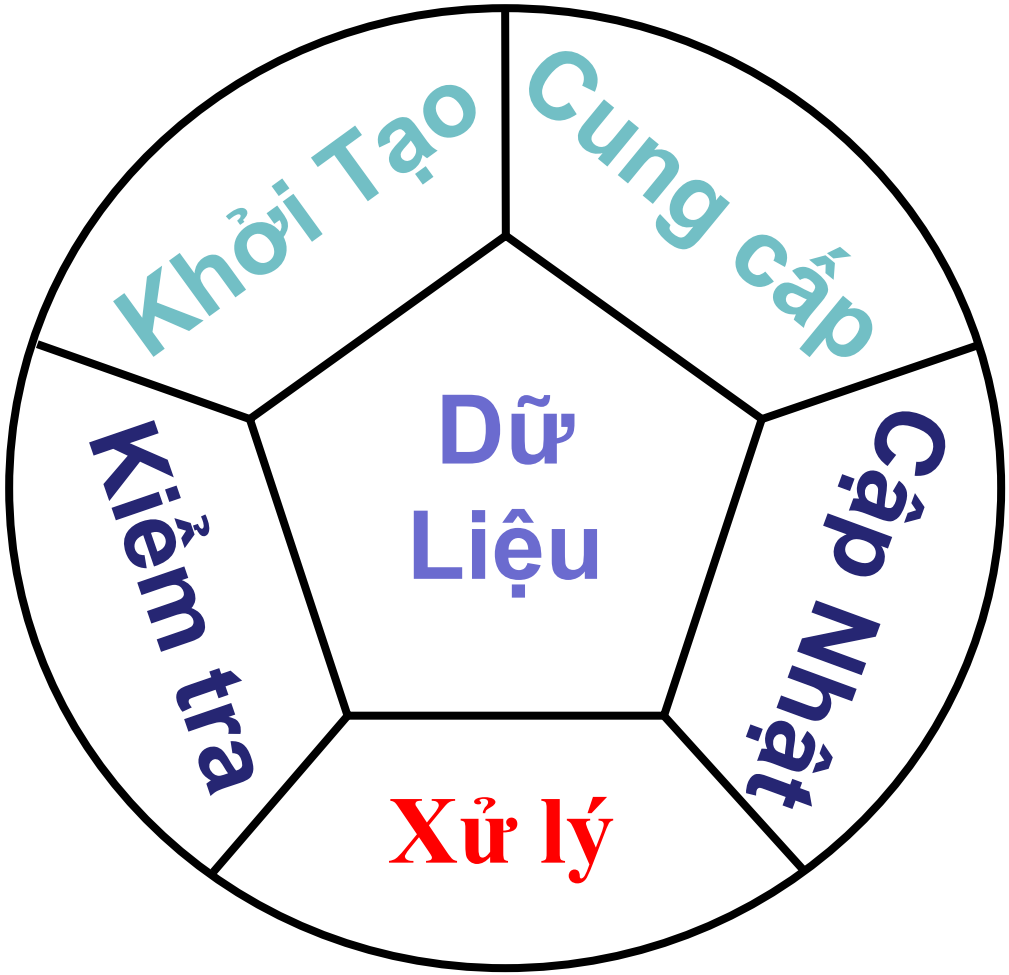
1. Đặt vấn đề

- Hãy thiết kế và xây dựng lớp CPhanSo trong toán học.



2. PHÂN LOẠI PHƯƠNG THỨC

2. Phân loại phương thức



2. Phân loại phương thức

– Các phương thức (method) của một lớp đối tượng (class) được chia thành 5 loại như sau:

+ Nhóm các phương thức khởi tạo.
+ Nhóm các phương thức cung cấp thông tin.
+ Nhóm các phương thức cập nhật thông tin.
+ Nhóm các phương thức kiểm tra.
+ Nhóm các phương thức xử lý.

2. Phân loại phương thức

- Nhóm các phương thức khởi tạo
  - + Khởi tạo (init) thông tin cho đối tượng.
  - + Các phương thức thiết lập (constructors) thuộc nhóm các phương thức khởi tạo.
  - + Các toán tử nhập (operator >>), toán tử xuất (operator <<) thuộc nhóm các phương thức khởi tạo.
  - + Các loại phương thức khởi tạo.
    - Phương thức khởi tạo mặc định.
    - Phương thức khởi tạo dựa vào đối tượng khác cùng thuộc về lớp.
    - Phương thức khởi tạo khi biết đầy đủ thông tin.

2. Phân loại phương thức

- Nhóm các phương thức cung cấp thông tin
  - + Các phương thức cung cấp thông tin của một lớp đối tượng (class) giữ nhiệm vụ cung cấp các thông tin (attributes, properties, information) của những đối tượng thuộc về lớp cho thế giới bên ngoài.
  - + Trong một lớp có bao nhiêu thuộc tính (attributes, properties) thì sẽ có ít nhất bấy nhiêu phương thức cung cấp thông tin.
  - + Thông thường các phương thức cung cấp thông tin bắt đầu bằng cụm từ `get`.

2. Phân loại phương thức

- Nhóm các phương thức cập nhật thông tin
  + Các phương thức cập nhật thông tin của một lớp đối tượng (class) giữ nhiệm vụ cập nhật các thông tin (attributes, properties, information) của những đối tượng thuộc về lớp khi có sự thay đổi (sự hiệu chỉnh, sự biến động).
  + Trong một lớp có bao nhiêu thuộc tính (attributes, properties) thì sẽ có ít nhất bấy nhiêu phương thức cập nhật thông tin.
  + Thông thường các phương thức cập nhật thông tin bắt đầu bằng cụm từ `set`.

2. Phân loại phương thức

- Nhóm các phương thức kiểm tra
  - + Các phương thức kiểm tra của một lớp đối tượng (class) giữ nhiệm vụ kiểm tra (test, check) tính hợp lệ của những thông tin (attributes, properties, information) thuộc về các đối tượng, kiểm tra một tính chất nào đó của đối tượng.
  - + Các phương thức toán tử so sánh (relational and comparison operators) thuộc nhóm các phương thức kiểm tra.
  - + Thông thường các phương thức kiểm tra bắt đầu bằng cụm từ `is`.

2. Phân loại phương thức

- Nhóm các phương thức xử lý.
  - + Các phương thức xử lý của một lớp đối tượng (class) giữ nhiệm vụ thực hiện các xử lý, tính toán cho lớp đối tượng.
  - + Các toán tử số học (arithmetic operators), toán tử gán (operator =) thuộc nhóm các phương thức xử lý.
  - + Phương thức phá hủy (destructor) thuộc nhóm các phương thức xử lý.

3. THIẾT KẾ LỚP PHÂN SỐ

3. Thiết kế lớp phân số

- Thiết kế các thuộc tính của lớp phân số.
- Lớp `CPhanSo` có hai thuộc tính là tử số (`tu`) và mẫu số (`mau`) với kiểu dữ liệu là số nguyên (`int`).

```
11.class CPhanSo
12.{
13.    private:
14.        int tu;
15.        int mau;
16.    …
17.};
```

3. Thiết kế lớp phân số

– Thiết kế các phương thức cung cấp thông tin.

```
11.class CPhanSo
12.{
13.|    …
14.|    public:
15.|        // Nhóm các phương thức
16.|        // cung cấp thông tin
17.|        int getTu();
18.|        int getMau();
19.|        float getGiaTri();
20.|    …
21.};
```

3. Thiết kế lớp phân số

- Định nghĩa các phương thức cung cấp thông tin.
- Cách 01.

```
11.int CPhanSo::getTu()
12.{
13.|    return tu;
14.}
```

- Cách 02.

```
11.int CPhanSo::getTu()
12.{
13.|    return this->tu;
14.}
```

Bên trong thân phương thức của một lớp đối tượng, this là một con trỏ đối tượng thuộc về lớp mà phương thức đó thuộc về, con trỏ đối tượng this giữ địa chỉ của đối tượng đang gọi thực hiện phương thức. Hơn nữa, *this chính là đối tượng đang gọi thực hiện phương thức.

3. Thiết kế lớp phân số

- Định nghĩa các phương thức cung cấp thông tin.
- Cách 01.

```
11.int CPhanSo::getMau()
12.{
13.|    return mau;
14.}
```

- Cách 02.

```
11.int CPhanSo::getMau()
12.{
13.|    return this->mau;
14.}
```

Bên trong thân phương thức của một lớp đối tượng, this là một con trỏ đối tượng thuộc về lớp mà phương thức đó thuộc về, con trỏ đối tượng this giữ địa chỉ của đối tượng đang gọi thực hiện phương thức. Hơn nữa, *this chính là đối tượng đang gọi thực hiện phương thức.

3. Thiết kế lớp phân số

- Định nghĩa các phương thức cung cấp thông tin.
- Cách 01.

```
11.float CPhanSo::getGiaTri()
12.{
13.|    return (float)tu/mau;
14.}
```

- Cách 02.

```
11.float CPhanSo::getGiaTri()
12.{
13.|    return (float)this->tu/this->mau;
14.}
```

3. Thiết kế lớp phân số

– Thiết kế các phương thức cập nhật thông tin.

```
11.class CPhanSo
12.{
13.|    …
14.|    public:
15.|        // Nhóm các phương thức
16.|        // cập nhật thông tin
17.|        void setTu(int);
18.|        void setMau(int);
19.|    …
20.};
```

3. Thiết kế lớp phân số

– Định nghĩa các phương thức cập nhật thông tin.

```
11.void CPhanSo::setTu(int tutu)
12.{
13.|    tu = tutu;
14.}

15.void CPhanSo::setMau(int maumau)
16.{
17.|    mau = maumau;
18.}
```

3. Thiết kế lớp phân số

– Thiết kế các phương thức kiểm tra.

```
11.class CPhanSo
12.{
13.|        …
14.|        // Nhóm các phương thức kiểm tra
15.|        int isCoNghia();
16.|        int isToiGian();
17.|        int isKhong();
18.|        int isDuong();
19.|        int isAm()
20.|        …
21.};
```

3. Thiết kế lớp phân số

– Thiết kế các phương thức kiểm tra.

```
11.class CPhanSo
12.{
13.        // Nhóm các phương thức kiểm tra
14.        bool operator>(CPhanSo&);
15.        bool operator<(CPhanSo&);
16.        bool operator>=(CPhanSo&);
17.        bool operator<=(CPhanSo&);
18.        bool operator==(CPhanSo&);
19.        bool operator!=(CPhanSo&);
20.        …
21.};
```

3. Thiết kế lớp phân số

– Định nghĩa các phương thức kiểm tra.

```
11.int CPhanSo::isCoNghia()
12.{
13.|    if(mau!=0)
14.|        return 1;
15.|    return 0;
16.}
```

3. Thiết kế lớp phân số

– Định nghĩa các phương thức kiểm tra.

```
11.int CPhanSo::isToiGian()
12.{
13.|    if(ucln(tu,mau)==1)
14.|        return 1;
15.|    return 0;
16.}
```

3. Thiết kế lớp phân số

– Định nghĩa các phương thức kiểm tra.

```
11.int CPhanSo::isKhong()
12.{
13.|    if(tu==0)
14.|        return 1;
15.|    return 0;
16.}
```

3. Thiết kế lớp phân số

– Định nghĩa các phương thức kiểm tra.

```
11.int CPhanSo::isDuong()
12.{
13.|    if(tu*mau>0)
14.|        return 1;
15.|    return 0;
16.}
```

3. Thiết kế lớp phân số

– Định nghĩa các phương thức kiểm tra.

```
11.int CPhanSo::isAm()
12.{
13.|    if(tu*mau<0)
14.|        return 1;
15.|    return 0;
16.}
```

3. Thiết kế lớp phân số

– Thiết kế các phương thức khởi tạo.

```
11.class CPhanSo
12.{
13.        // Nhóm các phương thức khởi tạo
14.        void KhoiTao();
15.        void KhoiTao(int);
16.        void KhoiTao(int,int);
17.        void KhoiTao(CPhanSo&);
18.        CPhanSo();
19.        CPhanSo(int);
20.        CPhanSo(int,int);
21.        CPhanSo(CPhanSo&);
22.};
```

3. Thiết kế lớp phân số

– Thiết kế các phương thức khởi tạo.

```
11.class CPhanSo
12.{
13.        // Nhóm các phương thức khởi tạo
14.        void KhoiTao();
15.        void KhoiTao(int);
16.        void KhoiTao(int,int);
17.        void KhoiTao(CPhanSo&);
18.        CPhanSo();
19.        CPhanSo(int);
20.        CPhanSo(int,int);
21.        CPhanSo(CPhanSo&);
22.};
```

3. Thiết kế lớp phân số

- Định nghĩa các phương thức khởi tạo.

```
11.void CPhanSo::KhoiTao()
12.{
13.|    tu = 0;
14.|    mau = 1;
15.}
```

- Phương thức khởi tạo mặc định, không nhận tham số đầu vào, các thông tin ban đầu của đối tượng được thiết lập mặc định như sau: tử lấy giá trị 0, mẫu lấy giá trị 1.

3. Thiết kế lớp phân số

– Định nghĩa các phương thức khởi tạo.

```
11.void CPhanSo::KhoiTao(int tutu)
12.{
13.|    tu = tutu;
14.|    mau = 1;
15.}
```

– Phương thức khởi tạo khi biết tử, nhận một tham số đầu vào là `tutu`, các thông tin ban đầu của đối tượng được thiết lập như sau: tử (`tu`) lấy giá trị `tutu`, mẫu (`mau`) lấy giá trị mặc định là 1.

3. Thiết kế lớp phân số

– Định nghĩa các phương thức khởi tạo.

```
11.void CPhanSo::KhoiTao(int tutu,int maumau)
12.{
13.|    tu = tutu;
14.|    mau = maumau;
15.}
```

– Phương thức khởi tạo khi biết đầy đủ thông tin, nhận hai tham số đầu vào là `tutu`, `maumau` các thông tin ban đầu của đối tượng được thiết lập như sau: tử (`tu`) lấy giá trị `tutu`, mẫu (`mau`) lấy giá trị `maumau`.

3. Thiết kế lớp phân số

– Định nghĩa các phương thức khởi tạo.

```
11.void CPhanSo::KhoiTao(CPhanSo &x)
12.{
13.|    tu = x.tu;
14.|    mau = x.mau;
15.}
```

– Phương thức khởi tạo dựa vào đối tượng khác cùng thuộc về lớp, nhận một tham số đầu vào là `x` là đối tượng thuộc lớp `CPhanSo`, các thông tin ban đầu của đối tượng được thiết lập như sau: tử (`tu`) lấy giá trị `x.tu`, mẫu (`tu`) lấy giá trị `x.mau`.

3. Thiết kế lớp phân số

– Định nghĩa các phương thức khởi tạo.

```
11.CPhanSo::CPhanSo()
12.{
13.|    tu = 0;
14.|    mau = 1;
15.}
```

– Phương thức thiết lập mặc định, không nhận tham số đầu vào, các thông tin ban đầu của đối tượng được thiết lập mặc định như sau: tử (`tu`) lấy giá trị 0, mẫu (`mau`) lấy giá trị 1.

3. Thiết kế lớp phân số

- Định nghĩa các phương thức khởi tạo.

```
11.CPhanSo::CPhanSo(int tutu)
12.{
13.|    tu = tutu;
14.|    mau = 1;
15.}
```

- Phương thức thiết lập khi biết tử, nhận một tham số đầu vào là `tutu`, các thông tin ban đầu của đối tượng được thiết lập như sau: tử (`tu`) lấy giá trị `tutu`, mẫu (`mau`) lấy giá trị mặc định là 1.

3. Thiết kế lớp phân số

- Định nghĩa các phương thức khởi tạo.

```
11.CPhanSo::CPhanSo(int tutu,int maumau)
12.{
13.|    tu = tutu;
14.|    mau = maumau;
15.}
```

- Phương thức thiết lập khi biết đầy đủ thông tin, nhận hai tham số đầu vào là `tutu`, `maumau` các thông tin ban đầu của đối tượng được thiết lập như sau: tử (`tu`) lấy giá trị `tutu`, mẫu (`mau`) lấy giá trị `maumau`.

3. Thiết kế lớp phân số

– Định nghĩa các phương thức khởi tạo.

```
11.CPhanSo::CPhanSo(CPhanSo &x)
12.{
13.|    tu = x.tu;
14.|    mau = x.mau;
15.}
```

– Phương thức thiết lập sao chép, nhận một tham số đầu vào là `x` là đối tượng thuộc lớp `CPhanSo`, các thông tin ban đầu của đối tượng được thiết lập như sau: tử (`tu`) lấy giá trị `x.tu`, mẫu (`mau`) lấy giá trị `x.mau`.

3. Thiết kế lớp phân số

– Thiết kế các phương thức khởi tạo.

```
11.class CPhanSo
12.{
13.        …
14.        // Nhóm các phương thức khởi tạo
15.        void Nhap();
16.        void Xuat();
17.        friend ostream& operator<<
18.                        (ostream&, CPhanSo&);
19.        friend istream& operator>>
20.                        (istream&, CPhanSo&);
21.        …
22.};
```

3. Thiết kế lớp phân số

– Định nghĩa các phương thức khởi tạo.

```
11.void CPhanSo::Nhap()
12.{
13.|    cout<<"Nhap tu: ";
14.|    cin>>tu;
15.|    cout<<"Nhap mau: ";
16.|    cin>>mau;
17.}
```

3. Thiết kế lớp phân số

– Định nghĩa các phương thức khởi tạo.

```
11.void CPhanSo::Xuat()
12.{
13.|    cout<<tu<<"/"<<mau;
14.}
```

3. Thiết kế lớp phân số

– Định nghĩa các phương thức khởi tạo.

```
11.istream& operator>>(istream&is,CPhanSo&x)
12.{
13.|    cout<<"Nhap tu: ";
14.|    is>>tu;
15.|    cout<<"Nhap mau: ";
16.|    is>>mau;
17.|    return is;
18.}
```

3. Thiết kế lớp phân số

– Định nghĩa các phương thức khởi tạo.

```
11.ostream& operator<<(ostream&os,CPhanSo&x)
12.{
13.|    os<<"\nTu: "<<tu;
14.|    os<<"\nMau: "<<mau;
15.|    return os;
16.}
```

3. Thiết kế lớp phân số

– Thiết kế các phương thức xử lý.

```
11.class CPhanSo
12.{
13.        // Nhóm các phương thức xử lý
14.        ~PhanSo();
15.        CPhanSo& operator=(CPhanSo&);
16.        void RutGon();
17.        CPhanSo Tong(CPhanSo&);
18.        CPhanSo Hieu(CPhanSo&);
19.        CPhanSo Tich(CPhanSo&);
20.        CPhanSo Thuong(CPhanSo&);
21.        …
22.};
```

3. Thiết kế lớp phân số

– Thiết kế các phương thức xử lý.

```
11.class CPhanSo
12.{
13.        …
14.        // Nhóm các phương thức xử lý
15.        CPhanSo operator+(CPhanSo&);
16.        CPhanSo operator-(CPhanSo&);
17.        CPhanSo operator*(CPhanSo&);
18.        CPhanSo operator/(CPhanSo&);
19.        …
20.};
```

3. Thiết kế lớp phân số

– Thiết kế các phương thức xử lý.

```
11.class CPhanSo
12.{
13.        // Nhóm các phương thức xử lý
14.        CPhanSo& operator++();
15.        CPhanSo& operator--();
16.        CPhanSo operator++(int);
17.        CPhanSo operator--(int);
18.        CPhanSo& operator+=(CPhanSo);
19.        CPhanSo& operator-=(CPhanSo);
20.        CPhanSo& operator*=(CPhanSo);
21.        CPhanSo& operator/=(CPhanSo);
22.};
```

3. Thiết kế lớp phân số

– Thiết kế các phương thức xử lý.

```
11.class CPhanSo
12.{
13.|        // Nhóm các phương thức xử lý
14.|        CPhanSo operator^(int);
15.|        CPhanSo operator^=(int);
16.};
```

Sinh viên tự định nghĩa các phương thức thuộc nhóm phương thức xử lý

4. REVIEW

```
11.class CPhanSo
12.{
13.    private:
14.        int tu;
15.        int mau;
16.    public:
17.        // Nhóm khởi tạo
18.        void KhoiTao();
19.        void KhoiTao(int);
20.        void KhoiTao(int,int);
21.        void KhoiTao(CPhanSo&);
22.        CPhanSo();
23.        CPhanSo(int);
24.        CPhanSo(int,int);
25.        CPhanSo(CPhanSo&);
26.};
```

```
11.class CPhanSo
12.{
13.    private:
14.        int tu;
15.        int mau;
16.    public:
17.        …
18.        // Nhóm cung cấp thông tin
19.        int getTu();
20.        int getMau();
21.        float getGiaTri();
22.        // Nhóm cập nhật thông tin
23.        void setTu(int);
24.        void setMau(int);
25.        …
26.};
```

```
11.class CPhanSo
12.{
13.        …
14.        // Nhóm kiểm tra
15.        int isCoNghia();
16.        int isToiGian();
17.        int isKhong();
18.        int isDuong();
19.        int isAm();
20.        bool operator>(CPhanSo&);
21.        bool operator<(CPhanSo&);
22.        bool operator>=(CPhanSo&);
23.        bool operator<=(CPhanSo&);
24.        bool operator==(CPhanSo&);
25.        bool operator!=(CPhanSo&);
26.};
```

```
11.class CPhanSo
12.{
13.        …
14.        // Nhóm xử lý
15.        ~PhanSo();
16.        CPhanSo& operator=(CPhanSo&);
17.        void RutGon();
18.        CPhanSo Tong(CPhanSo&);
19.        CPhanSo Hieu(CPhanSo&);
20.        CPhanSo Tich(CPhanSo&);
21.        CPhanSo Thuong(CPhanSo&);
22.        CPhanSo operator+(CPhanSo&);
23.        CPhanSo operator-(CPhanSo&);
24.        CPhanSo operator*(CPhanSo&);
25.        CPhanSo operator/(CPhanSo&);
26.};
```

```
11.class CPhanSo
12.{
13.    public:
14.        …
15.        // Nhóm xử lý
16.        CPhanSo& operator++();
17.        CPhanSo& operator--();
18.        CPhanSo operator++(int);
19.        CPhanSo operator--(int);
20.        CPhanSo& operator+=(CPhanSo);
21.        CPhanSo& operator-=(CPhanSo);
22.        CPhanSo& operator*=(CPhanSo);
23.        CPhanSo& operator/=(CPhanSo);
24.        CPhanSo operator^(int);
25.        CPhanSo operator^=(int);
26.};
```

Class Quality

5. CLASS QUALITY

Class Quality

- Booch proposes five metrics to measure the quality of classes:
  + Coupling
  + Cohesion
  + Sufficiency
  + Completeness
  + Primitiveness

Coupling

- How closely do classes rely on each other?
- Inheritance makes for strong coupling (generally a bad thing) but takes advantage of the re-use of an abstraction (generally a good thing).

Cohesion

- How tied together are the state and behavior of a class? Does the abstraction model a cohesive, related, integrated thing in the problem space?

Sufficiency

- Does the class capture enough of the details of the thing being modeled to be useful?

Completeness

- Does the class capture all of the useful behavior of the thing being modeled to be re-usable? What about future users (reusers) of the class?
- How much more time does it take to provide completeness?
- Is it worth it?

Primitiveness

- Do all the behaviors of the class satisfy the condition that they can only be implemented by accessing the state of the class directly? If a method can be written strictly in terms of other methods in the class, it isn't primitive.
- Primitive classes are smaller, easier to understand, with less coupling between methods, and are more likely to be reused. If you try to do too much in the class, then you're likely to make it difficult to use for other designers.

Primitiveness

- Sometimes issues of efficiency or interface ease-of-use will suggest you violate the general recommendation of making a class primitive. For example, you might provide a general method with many arguments to cover all possible uses, and a simplified method without arguments for the common case.
- The truism "Make the common case fast" holds, but in this case 'fast 'means "easy". Easy may violate primitive.

6. BÀI TẬP VỀ NHÀ

6. BÀI TẬP VỀ NHÀ

– Hãy thiết kế lớp cho các lớp đối tượng (class) sau:

1. Lớp điểm (CDiem).
2. Lớp điểm không gian (CDiemKhongGian).
3. Lớp phân số (CPhanSo).
4. Lớp hỗn số (CHonSo).
5. Lớp số phức (CSoPhuc).
6. Lớp ngày (CNgay).
7. Lớp thời gian (CThoiGian).
8. Lớp đơn thức (CDonThuc).

6. BÀI TẬP VỀ NHÀ

– Hãy thiết kế lớp cho các lớp đối tượng (class) sau:

9. Lớp đường thẳng (CDuongThang) trong mặt phẳng Oxy.
10. Lớp đường tròn (CDuongTron) trong mặt phẳng Oxy.
11. Lớp lớp tam giác (CTamGiac) trong mặt phẳng Oxy.
12. Lớp hình cầu (CHinhCau) trong không gian Oxyz.