类

struct 与 class 区别：

使用struct没有什么访问控制，则默认为public，而class为private；因此若希望所有成员为public用struct ，反之用class。

```
struct Boy{
  private:
    string name;
    mutable int age;
  public:
    //创建默认无参构造函数
    Boy() = default;
    Boy(string nm,int agenum): name(nm),age(agenum){}
    Boy(string nm):Boy(nm,1){}  // 构造函数可以调用构造函数进行初始化


    //声明为 const 成员函数，不能改变成员变量值，除非变量声明为mutable
    void printName() const{
      cout << this->name << endl;
      this->name = "change";     //报错，不能改变所有成员
    }

    string getName(){
      return this->name;
    }

    string setName(string name);     //声明，外部定义
    // 声明为友元就可以了，不管public和private，一般放最前面或者最后面
    friend void sout(Boy& boy);

    // 返回this的成员函数注意事项
    Boy &getAge(){       //如果不加引用，则返回的是Boy的副本，再对原来的值进行更改，只是更改副本
      cout << this->age  << endl;
      return *this;
    }
    Boy &setAge(int age){    // 如果声明成const方法，则不能返回this了，否则报错。
      this->age = age;
      return *this;
    }



};


string Boy::setName(string name){    //函数外部定义
    this->name = name;
}
```

```
45
46  //  定义类相关，非成员函数，放在同一个文件里就好了，可以使用友元
47  void pring(Boy boy){
48    cout << boy.getName() << endl;
49  }
50
51  // 因为在Boy里为友元函数，可以访问类私有成员
52  void sout(Boy boy){
53    cout << boy.name << endl;
54  }
55
```

对于类中this的理解

目前理解：this 存在于类的方法中，应该在调用方法的时候创建的，this本身是对象的常量指针，根据函数是什么类型来确定指针指向是常量还是非常量，因此你会看到常量函数返回const classA *const this这样的指针，或者非常量函数的 classA* const this这样的指针。

类的拷贝、赋值和析构

objectA = objectB; // 只是简单的把objectB赋值给objectA;

更详细见12、13章

关于类的定义及相互调用

```
1  class A;     // 这仅仅是声明A,如果是调用A里面一些方法或者变量，则会报错，因为此处仅仅告诉编译器，我有这个类，并没有定义。
2  解决办法：将类的定义与实现分离。
3  A a; // 这也使用了类的定义，初始化；因此用A* a;类指针可以；
```

类的构造函数（最好保证有默认构造函数，以防以后默认初始化出错）

```
1  classA(int a1,int b1):a(a1),b(b1){  }   // 对成员进行初始化，也就可以初始化常量
2  classA(int a1,int b1){a = a1; b = b1;}  // 对成员进行赋值操作，不能初始化常量
3  类成员初始化顺序按定义顺序来。
4  void classA::combine(classA a){ }   // 隐式的类类型转换，通过构造函数匹配初始化，创建了一个临时对象。 但通过在构造函数前加 explicit 关键字，便可以抑制构造函数定义的隐式转换。
```

聚合类

满足一下要求的类都是聚合类：

- 所有成员都是public的。
- 没有定义任何构造函数。
- 没有类内初始值。
- 没有基类，也没有virtual函数。

ps:

```
struct Data{
  int age;
  string name;
};
Data data1 = {22,"nier"};   // 可以这样初始化
```

字面常量类p267

类中的静态成员

注意：定义静态变量的时候注意静态变量必须初始化。

```
class A{
  static int a;
};
int A::a = 33;  //方法一：在外初始化
```

```
class A{
  static constexpr int tmp = 11;     // 这样初始化，只能用于在编译时初始化其他变量
  double array[tmp];     // 初始化array时用到的
}
```

```
class A{
    public:
        void seta(char c = bkground);    // 区别
    private:
        static const char bkground;      // 静态成员可以作为默认实参
}
```

注：c++中的this为指向本身的指针（常量且指向常量的指针），而Java不是