کلاس و اشیاء

Class
Object

design by ahmmad mohammad zade

کلاس

کلاس نوع جدیدی است که آن را برای حل مسئله های واقعی تعریف می کنند.

کلاس شامل موارد زیر است:

۱) داده ها
۲) تعریف عملیات

فیلد یا صفت : داده های عضو کلاس را صفت یا فیلد می گویند.
متد یا تابع : عملیات موجود درکلاس را متد یا تابع می گویند .

متدها بر روی صفات اجرا می شوند

design by ahmmad mohammad zade

کلاس و شی ء

کلاس :
مجموعه ای از شیاء که ویژگیهای مشتر ک دارند و رفتار یکسانی از خود نشان می دهند

شیء : نمونه ای خاص از کلاس

تعریف کلاس

Class نام کلاس

{

دادها و توابع اختصاصی

Public:

داده ها و توابع عمومی

Private:

داده ها و توابع اختصاصی

Protect:

داده ها ی محافظت شده

} اشیایی از کلاس;

design by ahmmad mohammad zade

مثال :یک کلاس ساده

```
class myclass {
        public:
                int a;
                int b;
        private:
                int c;
                int d;
};
```

```
myclass  w;
```

تعریف شی ء از نوع کلاس :

```
 w.a=12;
 w.b=13;
 w.c=5;
```

خطا چون داده ای اختصای است

مثال : یک کلاس دارای متد

```
#include <conio.h>
using namespace std;
class myclass   {
            public:
                    int a, b;
                    void claculation()
                        {
                            c=a+b;
                            d=a*b;
                        }
                     void display()
                        {
                             cout<< c <<"\t"<< d;
                        }
            private:
                        int c;
                        int d;
};
 void main()
{
         myclass  w;
        w.a=3;
        w.b=5;
        w.claculation();
        w.display();
       _getch();
}
```

خروجی :

```
8        15
```

تابع سازنده و مخرب :

```
class myclass {
        public:
                int a;
                int b;
                myclass(){a=0;b=1;};
                myclass(int x, int y ){a=x;b=y;};
                void claculation()
                {
                        c=a+b;
                        d=a*b;
                }
                void display()
                {
                        cout<<c<<"\t"<<d;
                }
                ~myclass(){};
        private:
                int c;
                int d;
};
```

تابع سازنده بدون آرگومان

تابع سازنده با آرگومان

تابع مخرب

تعریف شی ء هنگامیکه کلاس دارای تابع سازنده است :

```
myclass  w;
w.claculation();
w.display();
```

یا

```
myclass   r   =   myclass(7,8);
r.claculation();
r.display();
```

:

نکته : اگر سازنده دارای یک پارامتر باشد به صورت زیر تعریف می شود

```
myclass  object   =   2;
```

اشاره گر ها و کلاس

```
myclass  w,*p;
  w.a=3;
  w.b=5;
  w.claculation();
  w.display();
  p=&w;
  cout<<p->a ;
  (*p).a=20;
  Cout<<(*p).a;
```

نکته : حتما از پرانتز استفاده شود زیرا . بر * تقدم دارد

اشاره گر This

هر شی ء از طریق اشاره گری به نام this به آدرس خود دسترسی دارد .
این اشاره گر عضوی از شی ء نیست .
این اشاره گر به طوری ضمنی برای مراجعه به عضو یک شیء استفاده می شود .

```
# include "stdafx.h"
#include <conio.h>
using namespace std;
class myclass
{   public :
int x;
                int y;
                myclass()
                {  this ->x=10;
                   this ->y=20;
                   this ->a =12;
                   this ->b=13;
                }
                void print()  {cout <<"x="<<x<<"\ty="<<y<<"\ta="<<a<<"\tb="<<b;}
    private :
                 int a;
                 int b;
};
void main()
{    myclass p;
     p.print() ;
    _getch();
}
```

تابع دوست کلاس :

اعضایی از کلاس که به شکل خصوصی (private) اعلان می‌شوند فقط از داخل همان کلاس قابل دستیابی‌اند و از بیرون کلاس (درون بدنهٔ اصلی) امکان دسترسی به آن‌ها نیست.
اما یک استثنا وجود دارد.

تابع دوست:

تابعی است که عضو یک کلاس نیست اما اجازه دارد به اعضای خصوصی آن دسترسی داشته باشد.

```
#include "stdafx.h"
#include <conio.h>
using namespace std;
class myclass
{
public :
        friend   int   test(myclass);
        int a;
        int b;
        myclass(int x ,int y ){a=x;b=y;c=x+y;d=x*y;}
private :
         int c;
         int d;
};

int test(myclass r)
{
   return   r.c;
}

void main()
{
  myclass p(2,3);
  cout<< test(p);
 _getch();
}
```

design by ahmmad mohammad zade

وراثت

اغلب اوقات برای ایجاد یک کلاس جدید، نیازی نیست که همه چیز از اول طراحی شود.
می‌توانیم برای ایجاد کلاس مورد نظر، از تعاریف کلاس‌هایی که قبلا ساخته‌ایم، استفاده نماییم.
این کار باعث صرفه‌جویی در وقت و استحکام منطق برنامه می‌شود.

وراثت روشی برای ایجاد کلاس جدید از روی کلاس قبلی است. گاهی به وراثت « اشتقاق » نیز می‌گویند.

کلاس پایه	کلاس مشتق شده
شکل	دایره مثلث مستطیل
حساب	حساب جاری حساب پس انداز حساب کوتاه مدت

نحوه ایجاد کلاس بر اساس کلاس دیگر:

```
Class     نام کلاس مشتق شده   :   نوع دستیابی     نام کلاس پایه
{
    دستورات مربوط به کلاس
}
```

نوع دستیابی:

Public :

تمامی اعضای عمومی کلاس پایه اعضای عمومی کلاس مشتق شده خواهد بود
و تمامی اعضای محافظت شده کلاس پایه به عنوان اعضای محافظت شده کلاس مشتق شده است .
توجه داشته باشید که توابع دوست به ارث برده نمی وشوند

Private:

تمامی اعضای عمومی و محافظت شده کلاس پایه به عنوان اعضای اختصاصی کلاس مشتق شده خواهد بود

Protected:

اعضای کلاس پایه و دوست کلاس پایه و اعضای کلاس مشتق شده به انها دسترسی دارند.

مثال: نحوی ایجاد کلاس مشتق شده و بررسی وراثت

```
#include "stdafx.h"
#include <conio.h>
using namespace std;
class class1
{  int a;
  int b;
public :
                class1(){a=0;b=0;}
                class1(int x,int y){a=x;b=y;}
                void print(){cout<<a<<"\t"<<b<<endl;}
};
class class2:public class1
{   int c;
   public:
                class2(){c=1;}
                class2(int x){c=x;}
                void print(){cout<<"*******"<<endl;}
                void show(){cout<<c<<endl;}
};
void main()
{   class1 ob1(33,44);
    class2 ob2(55);
   ob1.print();
   ob2.print();
   ob2.show();
   _getch();
}
```

ساختمان

مجموعه ای از عناصر است که می توانند همنوع نباشند .
همانند کلاس است با این تفاوت اعضای ان به صورت پیش فرض عمومی هستند

نحوه ی تعریف ساختمان:

```
struct   نام ساختمان  {
                عناصر عمومی ساختمان
            Private:
                عناصر اختصاصی ساختمان
        };
```

مثال

```
struct  student
{
        void  input();
        void  print();
        private :
          int    code;
          char name [20];
} st1,st2;
student st3;
```

تعریف نوع با typedef

typedef نوع داده نام نوع جدید ;

مثال

typedef int integer;

مثال: در برنامه زیر کاربرد typedef ملاحظه می شود .

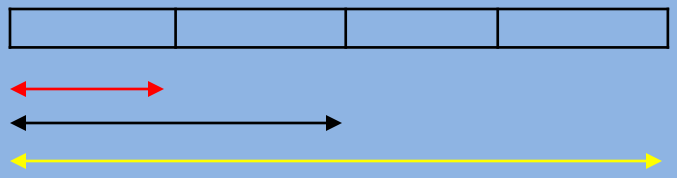
```
#include "stdafx.h“
 #include <conio.h>
using namespace std;
typedef int integer;
void main()
{
    integer x=12;
    cout<<x;
   _getch();
}
```

یونیون:

ساختمان داده ای است که تمام اعضای آن از یک محل ؛ از حافظه شروع می شود . اعضای آن در حالت عادی عمومی هستند

مثال:

```
union   test{
                int     x;
               float   y;
               char   ch;
} p;
```



فیلدهای بیتی:

هر یک از فیلدهای کلاس یا ساختمان می توانند بیتی باشند.
معمولا از نوع int یا unsigned هستند .
استفاده ی بهتر از حافظه را فراهم می کنند .
فیلدهای بیتی برای تشخیص و نگهداری وضعیت بایت های دستگاها بسیار مفید است .

مثال:

```
#include "stdafx.h“
 #include <conio.h>
using namespace std;
struct   bits{
                    unsigned    t:4;
                   unsigned    color:1;
                    unsigned     d:2;
};

void main()
{
        bits x;
        x.d=3;
        cout<<x.d<<endl;
        x.d=7;
        cout<<x.d ;
        _getch();
}
```

خروجی

```
3
3
```

داده های شمارشی :

با استفاده از این نوع می توان اعضای یک مجموعه متناهی را نامگذاری و شماره گذاری نموده مانند مجموعه رنگ ها ی سبز ،آبی ، قرمز و .....

مثال:

```
enum   color {
                        red ,
                        blue ,
                        green,
                        yellow
                };

enum color  x1 , x2;
```