مفاهیم و اصول شی گرا(Object Oriented)

-مقدمه:

روش های زیادی وجود دارند که می توان برای حل مسئله در نظر گرفت.یکی از روش هایی که به صورت گسترده در نرم افزار استفاده می شود دیدگاه شی گرایی است.در این حالت دامنه مسئله به صورت تعدادی از اشیائ دیده می شود که این اشیائ هر کدام دارای صفات و رفتار خاص خود هستند .این اشیاء توسط مجموعه ای از توابع(متد، عمکرد،سرویس)دستکاری می شوند و از طریق یک پروتکل پیام رسانی با یکدیگر ارتباط برقرار می سازند.
اشیاء در دسته بندی هایی با نام کلاس و ابر کلاس قرار می گیرند.
عمل تعریف کلاسها شامل تعریف کردن صفات،رفتار،عملیات و پیام ها می شود که این کار توسط مهندس نرم افزار انجام می گیرد.یک شی هم داده و هم پردازشی را که باید بر روی داده انجام شود را پنهان(کپسوله)می کند.
این خاصیت مهم منجر به ساخت اشیاء کلاسهایی می گردد که می توان از آنها استفاده مجدد نمود.
آنچه که در طی فرآیند مهندسی نرم افزار شی گرا تولید می شود مجموعه ای از مدل های شی گرا است که نیازمندی ها،طراحی،کد نویسی و فرآیند تست محصول را تشریح می کنند

از جمله مزایای استفاده از تکنولوژی اشیائ عبارتند از :

1)استفاده مجدد:افزایش سرعت توسعه و کیفیت برنامه تولید شده
2)نگهداری آسان:مجزا بودن ساختار نرم افزار یه صورت ذاتی، کم بودن تاثیرات جانبی هنگام تغییرات یک بخش
3)قابلیت وفق دهی و گسترش آسان:سیستم های حجیم را می توان به راحتی وبا سر هم کردن زیر یسیتم های از پیش آماده شده تهیه کرد.

-کپسوله سازی،وراثت و چند ریختی
(Encapsulation,Inheritance,Polymorphism)

الف)کپسوله سازی:

کپسوله سازی خواص و مزایایی زیر را ایجاد می کند.

1)جزئیات پیاده سازی داخل داده ها وروال ها از دنیای خارج پوشیده می شود.(پنخان سازی اطلاعات)
این عمل از منتشر شدن تاثیرات جانبی به هنگام تغییرات جلوگیری می کند.
2)ساختار داده ها و عملیاتی که آنها را دستکاری می کنند تحت یک نام یا نهادی واحد ترکیب می گردند.

نکته:رابطه بین اشیاء توسط سیستم پیام رسانی انجام می گردد.

ب)وراثت:

وراثت یکی از جنبه های تفاوت مهم سیستم های سنتی و شی گرا می باشد.تمامی صفات در کلاس و عملیات آن توسط زیر کلاس ها به ارث برده می شود.در هر سلسله مراتب کلاس ها ، صفات و عملیات جدیدی به کلاس افزوده می شود.

ج)چند شکلی:

چند شکلی خصوصیتی است که نیاز به توسعه و گسترش سیستم های شی گرای موجود را تا حد زیادی کاهش می دهد.
در این حالت، یک ابر کلاس داریم که سایر کلاس ها از آن ارث می برند.ابر کلاس دارای سرویس های پایه ای است

که کلاس های فرزند از آن ها ارث می برند و خصوصی سازی می کنند. این عمل را بار گذاری اضافی گویند. حال تمامی این کلاس ها دارای یکسری توابع هستند که از نظر نام و نحوه فراخوانی یکسان هستند اما در پیاده سازی در کلاس فرزند متفاوت هستند.بنابراین نیازی به تعریف های دیگر و مجدد نیست.

ایجاد کلاس در #C:

```
[modifier] class    classname
{
   Class Member
}
```

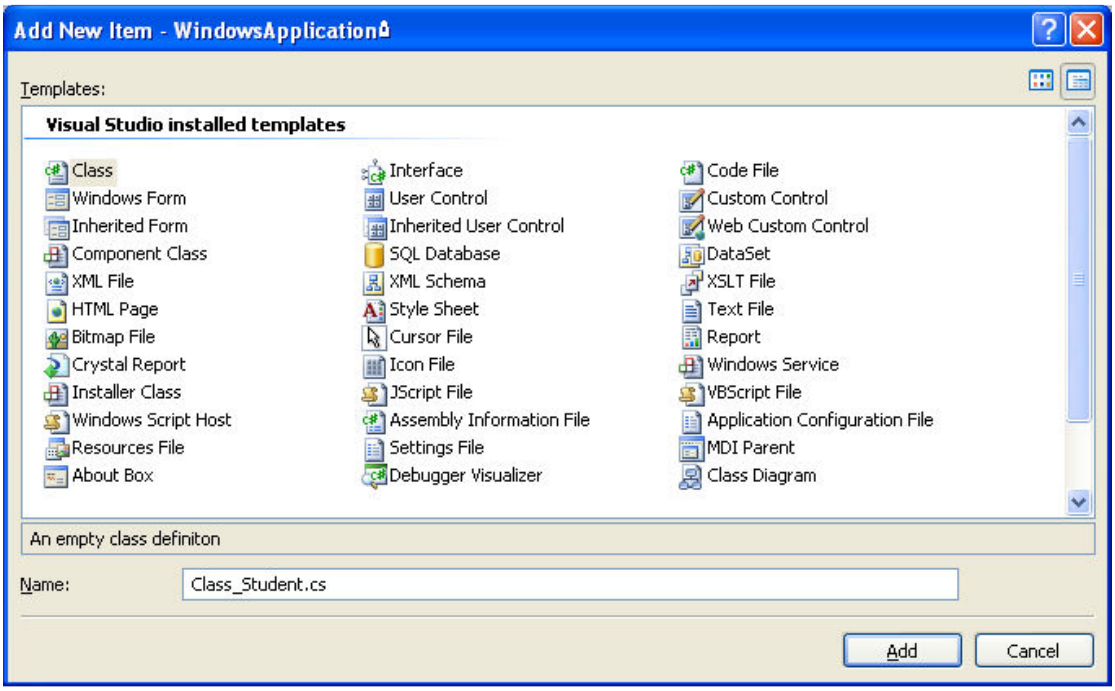
مثال:

```
Public  class test
{
     Private int x;
     Private int y;
     Public int sum()
     {
          Return x+y
      }
 }
```

Modifier:سطح دستیابی به کلاس را مشخص می کند.دو سطح دستیابی که در اینجا برای کلاس معرفی می کنیم، public و internal است.وقتی از سطح دستیابی (عمومی)public استفاده می کنیم به معنایی این است که این کلاس خارج از فضای نامی که در ان تعریف شده است قابل استفاده می باشد ولی سطح دستیابی internal(داخلی) مشخص می کند که کلاس فقط در همان فضای نامی که تعریف شده است قابل استفاده می باشد.

نکته: اگر در هنگام تعریف کلاس، سطح دستیابی به کلاس را مشخص نکنید، پیش فرض آن internal می باشد.

برای ساختن کلاس می توان از منوی project گزینه Add class را انتخاب کرد تا پنجره زیـر بـاز گردد.



در قسمت Name نام کلاس مورد نظر را وارد کرده و دکمه Add را کلیک کنید تا کلاس جدید ایجاد گردد.

نمونه سازی از کلاس:

پس از این که کلاس تعریف شد، برای استفاده از آن باید نمونه ای از آن ایجاد شود.ایجاد نمونه از کلاس را نمونه سازی(Instantiation)و نمونه های ایجاد شده از کلاس را شی(Object)می نامند.

شکل کلی نمونه سازی به صورت زیر می باشد:

```
ClassName  Obj= new ClassName();
```

مثال:ایجاد یک شی از کلاس test که در مثال قبل تعریف شده است.

```
test   ObjTest= new test()
```

با توجه به این نکته می توان تمامی شی ها ی موجود در زبان برنامه نویسی شی گرا را در زمان اجرا نیز ایجاد کرد. روش کار به صورت زیر می باشد.

در این مثال یک Button در زمان اجرا ایجاد و کنترل خواهیم کرد.

ایجاد یک شی از کلاس Button

```
Button  MyButton= new Button();
```

اضافه کردن شی ایجاد شده به لیست کنترل ها.درغیر اینصورت شی بر روی فرم نشان داده نخواهد شد.

```
Controls.Add(MyButton);
```

تنظیم متن داخل Button ایجاد شده

```
MyButton.Test="New Button";
```

تنظیم محل قرار گیری Button ایجاد شده بر روی فرم

```
MyButton.Location= new Point(۱۰۰,۱۰۰);
```

تنظیم اندازه Button ایجاد شده

```
MyButton.Size= new size(۱۵,۲۵);
```

برای اینکه شی ایجاد شده از رویداد ها پشتیبانی کند و بتوان از رویداد ها استفاد کرد باید از روش زیر استفاده شود.

```
MyButton.Click += new
System.EventHandler(this.Component_Click);
```

که در واقع Component_Click یک تابع می باشد که به صورت زیر تعریف شده و دستورات مورد نظری که باید در زمان رخ دادن رویداد اجرا شوند را در داخل آن می نویسیم.

```
  private void Component_Click(object sender,
EventArgs e)
   {
    this.BackColor = System.Drawing.Color.Yellow;
   }
```

با کلیک کردن بر روی دکمه ایجاد شده رنگ فرم به زرد تبدیل می شود.

نکته: this اشاره گری است که همیشه به کلاس پایه ای که در آن قرار داریم اشاره می کند.

برای نشان دادن فرم های دیگر برنامه نیز کافی است که یک شی از فرم طراحی شده به صورت زیر ایجاد کرد و آن را نشان داد.

```
Form frm_Customer = new Form۲();
frm_Customer.ShowDialog();
```

مثال:

-با استفاده از امکانات شی گرایی برنامه ای را بنویسید که تمامی TextBox های موجود بر روی فرم را بدون توجه به نام آنها خالی کند.

روش اول:

```
foreach(Control   Ctrl   in   this.Controls)
{
    if (Ctrl is TextBox)
            {
                 Ctrl.Text=String.Empty;
             }
}
```

روش دوم:

```
foreach (Control Ctrl in this.Controls)
     {
         if (Ctrl.GetType() == typeof(TextBox))
                {
                    Ctrl.Text = String.Empty;
                }
      }
```

روش سوم:

```
for (int i = ۰; i < this.Controls.Count - ۱; i++)
 {
    if (this.Controls[i] is TextBox)
       {
         (this.Controls[i] as TextBox).Text =
string.Empty;
       }
 }
```

تمرین:

1) هر سه روش فوق را اجرا کرده و تحلیل کنید.تفاوت آن را با روش سنتی ذکر کنید.
2) سعی کنید روش فوق را در زبان VB.net نیز به کار ببرید
3) با استفاده از روش ایجاد شی در زمان اجرا و روش تابعی بنویسید که در موقع اجرا برای تمامی TextBox های موجود بر روی فرم سایه ایجاد کند.

سطوح دستیابی به اعضای کلاس:

1) **public** : محدودیتی در دستیابی به آن وجود ندارد و در خارج از کلاس قابل استفاده می باشد معمولا متد ها و خواص را با این سطح دستیابی تعریف می کنند تا در خارج کلاس قابل دستیابی باشند.

2) **private** : اعضای با این سطح دستیابی، فقط در همان کلاس قابل استفاده و شناخته شده اند. معمولا فیلد ها و ثوابت با این ظطح دستیابی معرفی می شوند.

3) protected:سطح متوسطی بین public و private می باشد.اعضای protected کلاس پایه در وراثت، فقط در آن کلاس پایه یا هر کلاسی که از آن مشتق شده است ، قابل دستیابی اند.

4) **Static**: همان طور که دیدید از یک کلاس می توان چندین شی ایجاد کرد و هر شی دارای یک کپی ازمتغیر های خودش می باشد اما گاهی لازم است یک عضو بین تمامی شی ها مشترک باشد که برای این منظور عضو مورد نظر باید به صورت static تعریف شود.

نکته : اگر سطح دستیابی عضوی از کلاس تعیین نشود، private منظور می گردد.

-خواص (properties):

برای مقدار دهی به اعضای private داخل کلاس و گرفتن مقادیر از آنها باید property تعریف کرد که روش کار آن به صورت زیر می باشد:

برای مقدار دهی به خاصیت از واژه set و برا ی ارسال مقدار به خارج کلاس از get استفاده می شود.

```
private Boolean uCanFree;
        public Boolean CanFree
        {

            get
            {
                return uCanFree;
            }
            set
            {
                uCanFree = value;
            }
        }
```

نکته:اگر خاصیتی فاقد بخش set باشد فقط خواندنی و اگز فاقد بخش get باشد فقط نوشتنی می شود.

نکته:با استفاده از خواص می توان مقادیر ورودی و یا خروجی به متغیر private موجود در کلاس را کنترل کرد.

```
public  enum ActiveColor
        {
             Red=۰,
             Yellow=۱

        }

        private ActiveColor uInfo;
        public ActiveColor getInfo
        {

            get
            {
                return uInfo;
            }
            set
            {
                uInfo = value;
                if (uInfo == ActiveColor.Yellow)
  this.BackColor =    System.Drawing.Color.Yellow;
                 if (uInfo == ActiveColor.Red)
    this.BackColor = System.Drawing.Color.Red ;
            }

        }
```

همان طور که در قسمت بالا مشاهد می کنید. خواص می توانند به صورت داده های نوع شمارشی نیز تعریف شوند .

وراثت:

در وراثت، کلاس،رفتار یا صفاتی را از کلاس دیگر به ارث می بریم.

کلاسی که موجود است را کلاس پایه یا مافوق و کلاسی که از کلاس پایه ایجلد می شود، کلاس مشتق یا فرعی گویند.

چگونگی ارث بری:

```
Class derivedClass : baseClass
```

کلاس پایه:کلاس مشتق Class

مثال:

```
Class EmployeeService:Employee
{

}
```

کلاس های Sealed:

وقتی که کلاسی از این نوع تعریف شود دیگر اجازه ارث بری را ندارد و از وراثت آن جلوگیری می شود واگر متدی از این نوع تعریف شود دیکر اجازه تعریف مجدد در کلاس مشتق را ندارد.و دیگر Override نمی شود.متد ها یی که Static و یا private تعریف مـی شـوند بـه طـور خودکـارSealed هستند.

نکته: کلاس های که Sealed هستند نمی توانند کلاس پایه باشند.

نکته:واژه Sealed به کامپایلر اجازه بهینه سازی می دهد چـون کامپـایلر ایـن متـدد هـا را بـه صـورت inlining پیاده سازی می کند

نکته برنامه نویسی:

با استفاده از این کد می توان تعریف کرد که کاربر فقط عدد وارد نماید و داده ورودی را چک کرد.

```
if (!(char.IsNumber(e.KeyChar)))

 {
    e.KeyChar = Convert.ToChar(۰);
 }
```