شرح پروژه:

يك شركت بازاريابي در نظر دارد برخي از فعاليت هاي روزمره خود را مكانيزه نمايد تا بتواند با سرعت و دقت بيشتري آنها را مديريت كنيد .از اين رو سعي كرده ايم تا در اين پروژه برخي از اين نيازمنديها را پياده سازي نمائيم كه برخي از آنها در زير درج گرديده است:

تعريف بخش:

هر شركت ممكن است داراي بخش هاي مختلفي مانند سخت افزار و نرم افزار و .... باشد.بنابراين بايد اين امكان در برنامه وجود داشته باشد تا بتوانيم بخش هاي مختلف را در آن تعريف نماييم

تعريف پرسنل:

هر شركت داراي تعدادي پرسنل شاغل است كه در بخش هاي مختلف مشغول به كارند بنابراين اين امكان نيز بايد وجود داشته باشد تا پرسنل جديد را در سيستم تعريف كنيم و سپس بخش هايي كه بايد در آن كار كنند را مشخص نماييم

اختصاص پرسنل به بخش:

همانطور كه در بالا به آن اشاره كرديم اين امكان بايد وجو داشته باشد تا پرسنل هايي را كه تعريف نموده ايم به بخش هاي مختلف اختصاص دهيم لازم به ذكر است كه هر پرسنل فقط مي تواند در يك بخش شروع به فعاليت نمايد.

امنيت:

جهت بالا بردن امنيت فقط كاربران خاصي بايد از برنامه استفاده كنند ،بنابراين بايد قسمتي را براي اين امر اختصاص داد.

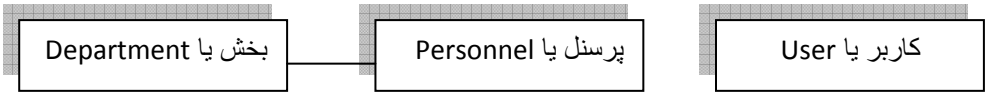
پياده سازي:

جهت كدنويسي اين پروژه از زبان C# و ويژوال استوديو 2010 نسخه Ultimate استفاده شده است و جهت سهولت در نوشتن كدها از بانك اطلاعاتي جهت ذخيره اطلاعات استفاده نشده است بلكه از يك كلاس كمكي جهت ذخيره اطلاعات استفاده شده است كه در ادامه به تفصيل توضيح داده خواهد شد.

براي كد نويسي قسمت هاي مختلف قصد داريم تا از تكنيك هاي شي گرايي استفاده نماييم زيرا درك برنامه را راحت تر مي كند و همچنين اين امكان را به ما مي دهد تا با سرعت بيشتري كد نويسي را انجام دهيم .

تجزيه و تحليل :

در اين مرحله با استفاده از نيازمندي هاي پروژه اشياء مورد نظر را مشخص و ارتباط بين آنها را مشخص نموده ايم كه در تصوير زير مي توانيد آن را مشاهده نماييد



ارتباط بين اشياء:

هر شي مي تواند با شي يا اشياء ديگري در ارتباط باشد که اين ارتباط ها مي تواند به شکل هاي زير باشد:

يك به يك: هر شي فقط و فقط با يك شي در ارتباط است براي مثال هر اتاق(شي اتاق) ميتواند فقط يك تخته(شي تخته) داشته باشد.و بر عکس آن هم نيز صحيح است

يك به چند: هر شي مي تواند با چند شي مثل هم در ارتباط باشد براي مثال هر استاد ميتواند چند دانشجو داشته باشد و هر دانشجو فقط ميتواند فقط يك استاد راهنما داشته باشد.

چند به چند: چند شي مي توانند با چند شي ديگر در ارتباط باشد براي مثال هر دانشگاه(شي دانشگاه) مي تواند چند استاد داشته باشد و هر استاد مي تواند در چندين دانشگاه تدريس نمايد.

با توجه به توضيحات بالا ما براي ارتباط بين شي بخش و پرسنل يك ارتباط يك به چند ايجاد کرده ايم به اين معني که هر بخش مي تواند چند پرسنل داشته باشد و هر پرسنل فقط در يك بخش مي تواند شاغل باشد.

در دنياي واقعي هر چيزي داراي يك سري خصوصيات ميباشد براي مثال يك اتومبيل را اگر در نظر بگيريم داراي خوصيات مهمي چون رنگ و مدل و تعداد در و نوع اتومبيل(سواري يا وانت)و.... و به همين نحو دانشجو داراي خصوصيات مهمي چون شماره دانشجويي ، نام ، نام خانوادگي و.... ميباشد . بنابراين با اين تفاصيل برخي از خاصيت هاي مهم ،اشياء بخش و پرسنل و همچنين کاربر را در زير مشخص نموده ايم تا در زمان کد نويسي بتوانيم از آنها استفاده نماييم.:

بخش: شماره بخش ، نام ، حداکثر تعداد پرسنل ، تلفن ، آدرس(ممکن است بخش ها از لحاظ مکان جغرافيايي در يك مکان نباشند)

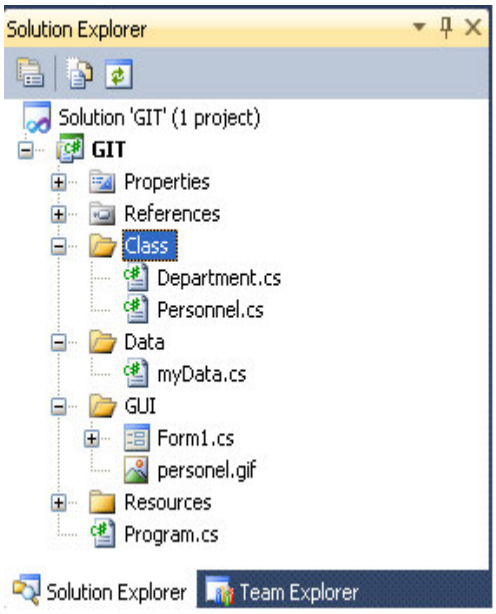
پرسنل: شماره پرسنلي، نام ، نام خانوادگي ، تلفن ، آدرس

کاربر: نام کاربري ، کلمه عبور

براي پياده سازي اشياء به صورت نرم افزاري از يك مفهوم انتزاعي به نام کلاس استفاده مي کنيم بدين صورت که ميبايستي براي هر شي يك کلاس تعريف کنيم و بعد از آن هر جا خواستيم از اشياء استفاده کنيم ، کافي است از کلاس مربوطه استفاده کرد.

بخش اول: پياده سازي کلاس ها

براي اين منظور ابتدا يك پروژه ويندوزي ايجاد کرده ايم و سپس براي اينکه قسمت هاي مختلف را از يگديگر متمايز کنيم چند فولدر به Solution اضافه نموده ايم تا آيتم هاي مختلف مثل فرم ها و کلاس ها را در آن بريزيم.که تصوير آن را در زير مشاهده مي نماييد.



در C# كلاس ها با پسوند cs. نمايش داده مي شود كه در شكل بالا مشاهده مي نماييد .

از آنجايي كه ساختار C# خود بر پايه شي گرايي بنا نهاده شده است بنابراين كليه اصول شي گرا در رابطه با ابزارهايي كه ويژوال استوديو در اختيار ما قرار ميدهد صدق مي كند و مي توانيم از آنها استفاده نماييم.

براي تعريف كلاس بخش يا Department روي پوشه Class كليك راست كرده و گزينه Add New Item... را انتخاب مي كنيم و در پنجره باز شده Class را انتخاب نموده و نام آن را Department مي گذاريم و سپس بر روي دكمه ADD كليك مي كنيم تا مانند شكل بالا به پروژه اضافه گردد .سپس بر روي آن دوبار كليك مي كنيم تا بتوانيم خاصيت هايي را قبلا مشخص نموديم وارد اين كلاس كنيم.هر كلاس داراي يك سطح دسترسي ميباشد كه در ابتداي تعريف هر كلاس و به صورت Publicو Private مشخص مي شود. اگر كلاسي با private شروع شود به اين معني است كه نميتوان از درون كلاس هاي ديگر به اين كلاس دسترسي پيدا كرد و از آنجايي كه قصد داريم بين كلاس بخش و پرسنل ارتباط برقرار كنيم بنابريان بهتر است كلاس را از نوع Public تعريف كنيم.

در زير نحوه پياده سازي كلاس Department مشخص گرديده است:

```
public class Department
{
    public string id;
    public string name;
    public string tel;
    public int maxperson;//ياب بازار پرسنل تعداد حداكثر
    public string address;
    public List<Personnel> personnels;
}
```

همانطور كه در خط آخر مشاهده مينماييد ليستي از پرسنل ها تعريف شده است كه با اين عمل در واقع رابطه بين كلاس بخش و كلاس پرسنل را ايجاد كرده ايم . List يك ساختمان داده جديد است كه مي تواند اشياء را در خود قرار دهد و طرز كار آن دقيقا مشابه يك آرايه است .

كه وظيفه اين ليست نگهداري پرسنل هاي مربوط به بخش است.

سازنده یا Constructor

اگر بخواهيم از كلاس Department يك شي جديد ايجاد كنيم بايد به شيوه زير عمل كنيم:

در محلي كه مي خواهيم شي را تعريف كنيم از كدهاي زير استفاده مي كنيم:

```
Department dep = new Department();
   dep.id = "10";
   dep.name = "افزار نرم ";
   dep.maxperson = 10;
```

همانطور كه مشاهده مي كنيد اين روش مقداري زمان گير است و همچنين فضاي كد نويسي بيشتري را به خود اختصاص ميدهد بنابراين روش ديگر استفاده از Constructor يا سازنده است .

ميتوان گفت سازنده يك متد است كه همنام با نام كلاس است و مي تواند آن را Overload كرد يعني چند سازنده براي يك كلاس تعريف كرده كه هر كدام پارامترهاي ورودي متعددي داشته باشند كه در زير سازنده كلاس Department را مشاهده مي نماييد.

```
public Department(string _id,string _name,string _maxperson,string _tel,string
_adres)
{
    id       = _id;
    name    = _name;
    tel = _tel;
    maxperson =int.Parse ( _maxperson);
    address = _adres;
    personnels = new List<Personnel>();
}
```

با توجه به توضيحات براي مثالي ميتوان سازنده اي ساخن كه فقط پارامتر ID و Name را داشته باشد.

با اين روش هر زمان كه نياز به تعريف بخش داشته باشيم مي توانيم از روش زير استفاده كنيم

```
Department DP1 = new Department("100", "افزار نرم بازاريابي","3","125","home1");
```

يعني مقادير را به سازنده پاس ميدهيم و خود سازنده وظيفه ايجاد شي را بر عهده مي گيرد.

براي ايجاد كلاس Personel نيز به همين صورت عمل مي كنيم .

از آنجايي كه در شي گرايي بهتر است وظايف هر كلاس را به خود كلاس واگذار نماييم و نه به كلاس هاي ديگر بنابراين توابع مربوط به هر كلاس را در خود كلاس تعريف مي كنيم براي مثال تابع زير چك مي كند كه آيا ظرفيت بخش تكميل شده است يا خير ؟ براي اين منظور يك مقدار صحيح گرفته و انرا با مقدار ظرفيت بخش كه در هنگام تعريف بخش وارد شده است چك مي كند و در صورتي كه برابر باشد تابع مقدار True را بر مي گرداند كه نشان ميدهد ظرفيت اين بخش كامل است و در غير اينصورت مقدار Falseرا بر مي گرداند.

```
public bool Check_department_full(int n)
{
    if (n == maxperson)
    {
        return false;
    }
    return true;
}
```

كلاس MyData

همانطور كه قبلا توضيح داده شد در اين برنامه از بانك اطلاعاتي استفاده نشده است براي اين منظور از يك كلاس به نام Mydata استفاده شده است كه اين كلاس در پوشه Data در Solution برنامه قرار دارد .اين كلاس به صورت زير تعريف شده است :

```
public static class myData
```

همانطور كه مشاهده مي كنيد با كلاس هاي ديگر كه قبلا تعريف نموديم يك تفاوت دارد و آنهم استفاده از كلمه كليدي Static است .زماني كه قصد داريد از يك كلاس استفاده كنيد لازم است يك شي جديد از آن ايجاد كنيد كه اينكار با كلمه كليدي New انجام ميشود و مثال هايي از آن را در صفحه هاي قبل مشاهده نموده ايد ولي كلاس هايي كه به صورت Static تعريف مي شوند نيازي نيست تا از آنها يك شي جديد ايجاد كنيم و تنها كافي است نام كلاس را نوشته و سپس بعد از آن نام تابع يا خاصيت هايي كه در درون كلاس تعريف كرده ايم را درج كنيم براي مثال تابع Show از كلاس MessageBox كه براي نمايش پيغام استفاده مي شود يك كلاس Static است زيرا نيازي نيست يك شي جديد از آن تعريف نماييم.

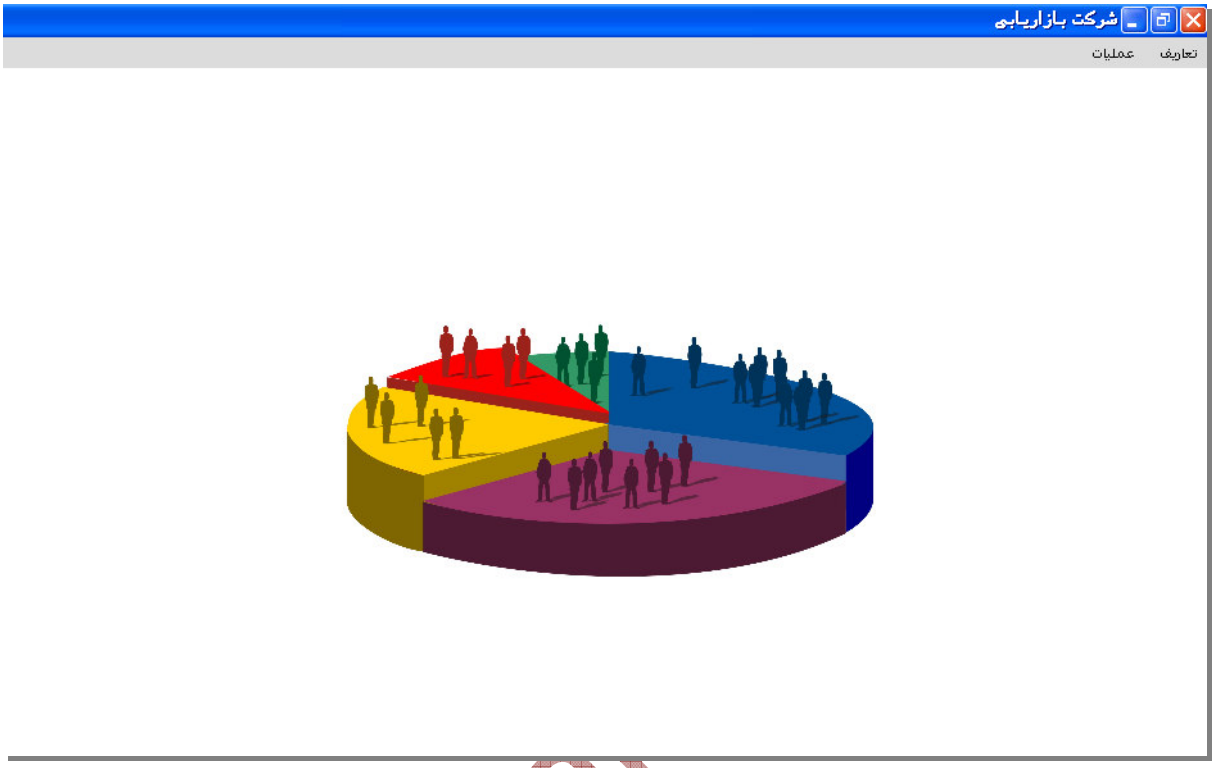
در كلاس Mydata يك تابع به نام BuildMyadata() وجود دارد كه وظيفه آن ايجاد چند شي نمونه از كلاس بخش و پرسنل و .... است كه برنامه با اين داده هاي مجازي شروع به كار نمايد و بعد از آن هر تعريفي كه در برنامه انجام دهيم درون ليستي كه در اين كلاس تعبيه شده است ذخيره مي شود. در زير تعريف دو تابع و ليست هاي ذخيره را مشاهده مي نماييد:

```
        public static List<Department> departments;
        public static List<Personnel> personnels;

        public static void BuildmyData()
        {
            departments = new List<Department>();
            personnels = new List<Personnel>();
            Department DP1 = new Department("100", "افزار نرم بازاريابي","3","125","home1");
            Department DP2 = new Department("101", "سخت بازاريابي
افزار","2","126","home2");
            Department DP3 = new Department("102", "محصولات بازاريابي
شبكه","1","127","home3");
            departments.Add(DP1);
            departments.Add(DP2);
            departments.Add(DP3);


            Personnel Prs1 = new Personnel("1","علی","حمیدی","20","شیراز");
            Personnel Prs2 = new Personnel("2", "سعید", "مجیدی", "15", "اصفهان");
            Personnel Prs3 = new Personnel("3", "مینا", "حسینی", "17", "تهران");
            personnels.Add(Prs1);
            personnels.Add(Prs2);
            personnels.Add(Prs3);



        }
```

در زمان شروع به كار برنامه يا اجراي برنامه اين تابع اجرا مي شود و اشياء مورد نظر را در دورن حافظه ايجاد مي كند و آنها را درون ليست هاي مربوط به هر شي قرار ميدهد تا در صورت لزوم از آنها استفاده شود.

فرم اصلي برنامه

زماني كه كاربر برنامه را اجرا مي كند فرم زير اجرا مي شود كه اين فرم درواقع فرم اصلي برنامه است و از آن ميتوان به كليه قسمت هاي برنامه دسترسي پيدا كرد.



در قسمت بعدي سعي داريم تا فرمي را براي تعريف بخش ايجاد نماييم و دسترسي آن را از طريق فرم اصلي مهيا سازيم.

پايان قسمت اول